FROM SEEING TO SOLVING: BENCHMARKING AND DIAGNOSING SYMBOLIC REASONING IN LARGE VISION-LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large vision-language models (LVLMs) remain unreliable on symbolic reasoning tasks, yet final-answer accuracy alone provides limited insight into whether failures arise from incorrect problem recovery or downstream symbolic solving. We curate the Multimodal Constraint Satisfaction Problem Benchmark (MMCSP-Bench), which provides matched structured-text, standard-image, and out-of-distribution (OOD) image views of Sudoku, Nonogram, and Graph Coloring, with all candidate solutions verified against the original constraints. We further propose the Declarative Constraint Satisfaction Problem Interface (DeCSP), a controlled symbolic interface that replaces unconstrained solver-program generation with predefined constraint-operator selection and deterministic execution. Across seven LVLMs, DeCSP achieves 96.24% mean valid-solution accuracy on structured text, compared with 8.19% for direct answering and 43.43% for PAL. However, DeCSP accuracy decreases to 42.57% on standard images and 28.19% on OOD images, revealing substantial challenges in multimodal symbolic reasoning. Controlled evaluations with fixed solvers, correct structures, and shared transcriptions disentangle visual-recovery failures from symbolic-generation failures, showing that recovery quality and final validity measure distinct dimensions of reasoning. Our stage-targeted training further shows that recovery improvements do not consistently transfer to direct answering, while program-training effects vary across models. Code and data are available at `https://anonymous.4open.science/r/Seeing2Solving-37532`.

1 INTRODUCTION

Despite substantial progress in mathematical reasoning and code generation (Wei et al., 2022; Guo et al., 2025), large language models (LLMs) and large vision-language models (LVLMs) remain unreliable on symbolic reasoning tasks that require satisfying multiple interacting constraints. Recent evaluations on Sudoku variants and logic-grid puzzles reveal persistent difficulties in constraint-based reasoning (Seely et al., 2025; Lin et al., 2025), highlighting the challenge of reliably transforming structured problems into valid solutions.

Program-Aided Language Models (PAL) address part of this challenge by shifting deterministic computation from the LLM to an external interpreter through program generation (Gao et al., 2023). We adapt PAL to generate complete Python solvers for Sudoku, Nonogram, and Graph Coloring and evaluate this route across seven LVLMs. This formulation still requires the model to specify both the constraints and search procedure, making symbolic generation an additional source of failure.

Prior studies have investigated visual parsing, evidence utilization, and multimodal reasoning behavior (Leng et al., 2025; Liu et al., 2026), yet the internal sources of reasoning failures remain insufficiently understood. Final-answer accuracy is only an outcome-level metric and cannot reveal which stage of the reasoning pipeline failed. An LVLM may fail during problem recovery, symbolic generation, or solution execution while producing the same final outcome. This motivates distinguishing recovery errors from downstream symbolic reasoning errors and raises three questions: how visual recovery affects solving performance, what symbolic challenges remain after correct recovery, and whether improving one stage leads to better end-to-end solving.

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.

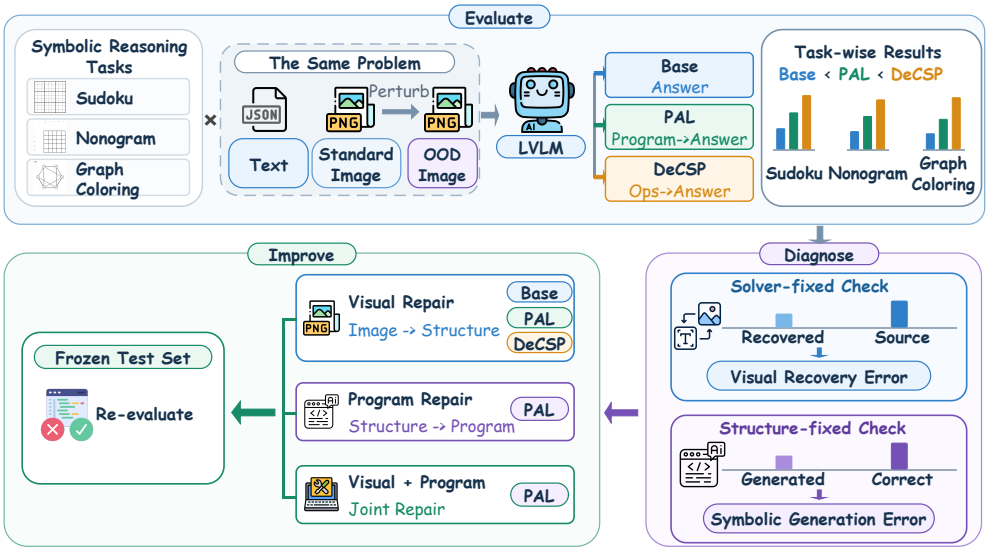

Figure 1: **Overview of Seeing2Solving.** Base generates a candidate answer, PAL generates a solver program, and DeCSP selects predefined constraint operators. Controlled comparisons fix either the solver or the supplied structure to identify failures in visual recovery and symbolic generation, guiding stage-targeted repair. Every candidate is verified against the frozen original problem.

To make these questions measurable, we present **Seeing2Solving** (Figure 1), a framework for evaluating, diagnosing, and improving symbolic reasoning in LVLMs. We introduce the **Multi-modal Constraint Satisfaction Problem Benchmark (MMCSP-Bench)**, which provides matched structured-text, standard-image, and out-of-distribution (OOD) image views of the same constraint problems. Each instance retains its original problem structure, allowing recovered representations to be compared with the source structure and final candidates to be verified against the original constraints. This design makes problem recovery and solution validity separately measurable across input views.

Moreover, we present the **Declarative Constraint Satisfaction Problem Interface (DeCSP)**. Instead of requiring an LVLM to generate a complete solver, DeCSP asks the model to select operators from a predefined constraint library while delegating constraint construction and search to a deterministic backend. This controlled interface separates symbolic selection from solver implementation, allowing the model's operator choices to be examined separately from deterministic constraint construction and search. When applied to images, however, DeCSP still relies on accurately recovering the underlying problem structure before selecting constraints, so visual recovery remains a separate source of error.

We use this structure to disentangle **visual recovery** from downstream **symbolic generation**. Specifically, we evaluate recovered structures with the same fixed solver, provide correct structures to isolate symbolic generation, and compare declarative interfaces under shared transcriptions. These controls show that recovery quality and final validity need not coincide: non-exact recovery may still yield valid solutions, while exact recovery may be followed by downstream failure.

Finally, we study diagnosis-guided interventions for different solving stages and test whether improvements transfer across routes. Our contributions are threefold:

1. **A diagnostic benchmark.** MMCSP-Bench provides matched multimodal views and original-problem verification for stage-level diagnosis.
2. **A controlled symbolic reasoning interface.** DeCSP isolates operator selection from deterministic constraint construction and search.
3. **A framework for stage-targeted improvement.** Seeing2Solving evaluates targeted visual and program interventions and tests whether they transfer to end-to-end solving.

Table 1: Comparison with representative work on visual reasoning and diagnosis.

Work	Inputs	Diagnostic focus	Intervention
NS-VQA (Yi et al., 2018)	Image	Scene structures; programs	Scene and program training
VGRP-Bench (Ren et al., 2025)	Text, image	Grid perception; rule adherence	Solution or reasoning fine-tuning
CrossWordBench (Leng et al., 2025)	Text, image	Grid parsing; answer consistency	Interactive solving; self-reflection
Seeing but Not Believing (Liu et al., 2026)	Image	Attention to evidence regions	Attention-guided highlighting
VisuRiddles (Yan et al., 2026)	Image, description	Fine-grained perception	Perception then reasoning training
MMCSP-Bench + DeCSP (ours)	Text, image, OOD image	Exact recovery; rule/program errors	Visual transfer; separate PAL training

2 Related Work

Table 1 summarizes representative efforts in visual reasoning, symbolic solving, and multimodal diagnosis. We review these directions and discuss how existing approaches evaluate symbolic reasoning, introduce external execution, and analyze intermediate failures.

2.1 Symbolic and Visual Reasoning Benchmarks

Symbolic reasoning benchmarks such as Sudoku-Bench and ZebraLogic evaluate whether models can satisfy interacting constraints (Seely et al., 2025; Lin et al., 2025). MathVista and VisuLogic extend this evaluation to mathematical and logical reasoning with visual inputs (Lu et al., 2024; Xu et al., 2026). Recent puzzle benchmarks such as VGRP-Bench and CrossWordBench further introduce controllable generation and text/image comparisons to study the relationship between perception and solving (Ren et al., 2025; Leng et al., 2025). Our focus is on distinguishing errors in problem recovery from downstream symbolic-generation errors through controlled comparisons.

MMCSP-Bench retains the source structure for each set of matched views. Recovered structures are compared with this reference, while candidate solutions are verified against the original constraints, enabling separate evaluation of recovery quality and final validity.

2.2 Reasoning with Programs and Symbolic Solvers

PAL and Program of Thoughts generate executable programs and delegate computation to external interpreters (Gao et al., 2023; Chen et al., 2022). Solver-based approaches such as SatLM and Logic-LM translate natural language into symbolic formulations for downstream reasoning engines (Ye et al., 2023; Pan et al., 2023). Similar ideas have also been explored in visual reasoning through tool composition frameworks such as VISPROG and ViperGPT (Gupta & Kembhavi, 2023; Surís et al., 2023). These approaches reduce the burden of direct answer generation by introducing intermediate symbolic representations or external execution.

2.3 Diagnosing Multimodal Failures and Targeted Training

Early neural-symbolic systems introduced explicit intermediate representations to separate visual perception from symbolic execution (Yi et al., 2018; Amizadeh et al., 2020). Recent studies further investigate visual evidence utilization, input interventions, and staged perception-reasoning training (Liu et al., 2026; Yan et al., 2026). Constrained decoding studies address another aspect of reliability by enforcing output-format validity (Scholak et al., 2021; Geng et al., 2023). These studies motivate evaluating intermediate states of multimodal reasoning in addition to final predictions.

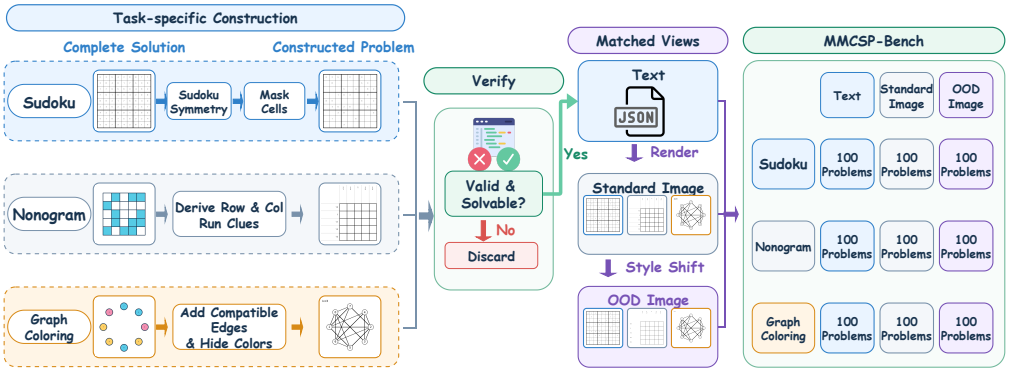

Figure 2: **Construction of MMCSP-Bench.** Task-specific generators turn complete solutions into problems, which are checked for validity and solvability before inclusion. Each retained problem is provided as structured text, a standard image, and an OOD image without exposing the construction solution. The three views share the same constraints and represent the same 100 problems per task.

3 MMCSP-BENCH

Figure 2 illustrates the construction pipeline of MMCSP-Bench. MMCSP-Bench is designed around three principles: (1) matched multimodal views of the same problem, (2) preservation of the original problem structure, and (3) verification against original constraints rather than recovered instances. Each problem instance is rendered into structured-text, standard-image, and OOD-image views, while the original structure is retained as the reference for evaluation. This design enables separate evaluation of problem recovery and final solution validity.

3.1 TASK-SPECIFIC CSP CONSTRUCTION

MMCSP-Bench is constructed from 300 source problems, with 100 problems in each of three complementary CSP families: Sudoku emphasizes local consistency constraints, Nonogram imposes row and column run-length constraints, and Graph Coloring evaluates relational constraints over graph structures.

Sudoku. We generate valid 9×9 solution grids, apply rule-preserving transformations, and remove entries to create partially observed puzzles. The model receives only the incomplete grid. A valid solution preserves the givens and satisfies row, column, and 3×3 box constraints.

Nonogram. We generate binary grids and derive row and column clue sequences from consecutive filled-cell runs. The model receives only the grid dimensions and clues. A valid solution must reproduce all clue sequences.

Graph Coloring. We generate graphs with hidden color assignments and remove the assignments from the input, leaving only nodes, edges, and the color budget. A valid solution assigns colors such that adjacent nodes receive different colors within the allowed budget.

We retain only valid and solvable instances, and exclude all construction-time solutions from model inputs to prevent information leakage.

3.2 MATCHED INPUT VIEWS

Each problem is represented by a canonical structure x without its construction solution. From the same structure, we generate three matched views: structured text, standard image, and OOD image. The structured-text view exposes the problem representation directly, whereas image views introduce visual variation while preserving all constraints. OOD is restricted to presentation changes and does not alter task rules or problem structures; task-specific rendering settings are detailed in Appendix A.1.

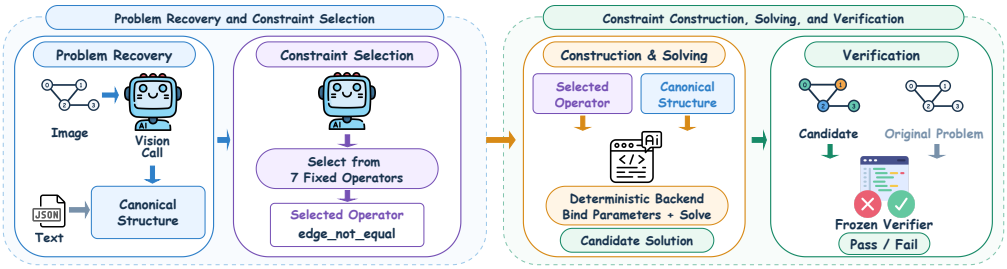

Figure 3: **DeCSP workflow.** The LVLM selects predefined constraint operators from a recovered or structured input representation. A deterministic backend instantiates the operators, searches for a solution, and verifies the candidate against the original constraints.

The primary evaluation therefore contains 900 rendered problem instances: for each of the three tasks, the same 100 source problems are provided under each of the three matched input views. Auxiliary instances used by the separately reported training controls are described with their corresponding protocols in Appendix C.

3.3 Original-Problem Verification

To ensure that models are evaluated on the source problem rather than a recovered surrogate, all candidates are verified against the original problem structure x. For a candidate solution y, the verifier returns $V(x, y) = 1$ iff y satisfies the required value domain and all constraints in x. We report valid-solution accuracy as the fraction of evaluated problem instances with $V(x, y) = 1$. Any satisfying solution is accepted, even if it differs from the construction solution, and the verifier never substitutes a model-recovered structure $\hat{x}$ for x.

4 DeCSP: Declarative Constraint Satisfaction Problem Interface

We introduce DeCSP as a symbolic interface that separates constraint selection from constraint construction and search. Unlike PAL, where the LVLM generates an executable solver, DeCSP restricts the model output to selecting predefined constraint operators. The deterministic backend then instantiates these operators, solves the resulting CSP, and verifies the candidate solution.

As shown in Figure 3, the DeCSP pipeline contains three stages: recovering the problem structure when needed, selecting symbolic constraint operators, and constructing and solving the resulting CSP with a deterministic backend. This design changes the allocation of symbolic responsibilities: the LVLM performs high-level symbolic selection, while deterministic components handle constraint implementation and search. Such separation enables analysis of whether failures originate from symbolic decisions or downstream execution.

4.1 Problem Recovery and Constraint Selection

For each instance, let x denote the original public constraint structure. Structured text supplies x directly, whereas an image I is transcribed into a recovered structure $\hat{x} = T_\theta(I)$. Thus, DeCSP receives $\tilde{x} = x$ for text inputs and $\tilde{x} = \hat{x}$ for image inputs.

The LVLM receives the supplied representation $\tilde{x}$ together with task information and selects a set of predefined constraint operators. The output contains only operator identifiers; constraint implementations and search procedures are provided by the deterministic backend.

For the fixed primary benchmark, DeCSP uses a closed operator vocabulary to measure reliable operator selection under a stable task interface. We separately evaluate language-conditioned selection on unseen rule descriptions in Appendix E.

Table 2: Models used in our experiments.

Model Name	Type
Primary evaluation panel (seven models)	
`Qwen3.5-4B` (Qwen Team, 2026a)	LVLM
`Qwen3.5-9B` (Qwen Team, 2026a)	LVLM
`Qwen3-VL-32B-Instruct` (Bai et al., 2025)	LVLM
`Qwen3.8-27B` (Qwen Team, 2026b)	LVLM
`gemma-3-27b-it` (Gemma Team et al., 2025)	LVLM
`gemma-4-12B-it` (Gemma Team et al., 2026)	LVLM
`Mistral-Small-3.2-24B-Instruct-2506`[1]	LVLM
Auxiliary text-only study (one model)	
`Mistral-Small-24B-Instruct-2501`[2]	Text-only LLM

4.2 Constraint Execution and Verification

Given the selected operators and supplied structure, the backend constructs the corresponding CSP and searches for a candidate solution. For image inputs, construction and search operate only on $\hat{x}$, while the original structure x is used solely for verification. This design supports controlled comparisons that distinguish incorrect symbolic selection from errors in problem recovery.

5 Experiments

We organize our experiments around three objectives: evaluating end-to-end solving performance, diagnosing failures across intermediate stages, and testing stage-targeted improvements.

5.1 Models and Inference Settings

As shown in Table 2, we use seven LVLMs in the fixed primary evaluation panel and one additional text-only LLM in a separate variable-rule feedback study. Detailed inference settings, output contracts, and execution environments are provided in Appendix B.

5.2 Evaluation and Diagnostic Protocols

We evaluate Base, PAL, and DeCSP on 100 source problems per task across structured-text, standard-image, and OOD-image views. Structured-text routes receive the source problem structure directly. For images, Base generates a candidate from the rendered input, whereas PAL and DeCSP first recover a public problem structure before generating a solver or selecting constraint operators. All candidates are verified against the original problem constraints.

We use four controls for stage-level diagnosis. Known-Program fixes the downstream solver to isolate visual recovery; the correct-structure comparison removes visual uncertainty to compare symbolic interfaces; the shared-transcription comparison gives identical recovered structures to DeCSP and CDT, a declarative interface that generates reusable rules; and the unseen-wording study tests operator selection beyond the fixed benchmark wording. We additionally report transcription parseability, indicating schema-conformant output, and exact recovery, which compares task-normalized recovered and source structures. Detailed protocols, first-failure definitions, and statistical procedures are provided in Appendix B.

5.3 Stage-Targeted Training Setup

We further evaluated targeted interventions on different stages of the solving pipeline. For visual recovery, we trained image-to-structure components using public problem-JSON targets and evaluated the trained components within the Base and DeCSP solving routes. For program generation,

[1] `https://huggingface.co/mistralai/Mistral-Small-3.2-24B-Instruct-2506`
[2] `https://huggingface.co/mistralai/Mistral-Small-24B-Instruct-2501`

Table 3: Main benchmark valid-solution accuracy (%). Panel (a) averages seven LVLMs within each task; panel (b) averages the three tasks within each model. Both panels compare Base, PAL, and DeCSP across matched structured-text, standard-image, and OOD-image views.

(a) Task-level accuracy

	Structured text			Standard image			OOD image		
Task	Base	PAL	DeCSP	Base	PAL	DeCSP	Base	PAL	DeCSP
Sudoku	0.29	86.71	100.00	0.14	57.86	58.14	0.00	35.00	35.29
Nonogram	0.71	1.00	88.71	0.00	0.86	45.00	0.00	0.71	26.29
Graph Coloring	23.57	42.57	100.00	6.29	9.00	24.57	6.14	8.43	23.00
Mean	**8.19**	**43.43**	**96.24**	**2.14**	**22.57**	**42.57**	**2.05**	**14.71**	**28.19**

(b) Accuracy by LVLM and input view

	Structured text			Standard image			OOD image		
LVLM	Base	PAL	DeCSP	Base	PAL	DeCSP	Base	PAL	DeCSP
Qwen3.5-4B	3.33	3.00	100.00	0.00	0.00	17.33	0.00	0.00	11.00
Qwen3.5-9B	5.67	49.00	74.67	1.00	39.00	50.33	1.67	34.00	42.67
Qwen3-VL-32B	4.67	67.67	100.00	0.67	34.33	64.67	0.33	11.33	29.00
Qwen3.8-27B	6.00	34.33	100.00	1.67	34.00	81.67	1.33	30.00	72.33
Gemma3-27B	5.33	66.67	99.00	2.33	10.67	26.67	2.33	2.33	3.67
Gemma4-12B	31.00	49.67	100.00	8.00	36.00	44.00	7.00	25.00	32.33
Mistral-3.2-24B	1.33	33.67	100.00	1.33	4.00	13.33	1.67	0.33	6.33
Mean	**8.19**	**43.43**	**96.24**	**2.14**	**22.57**	**42.57**	**2.05**	**14.71**	**28.19**

we trained PAL program adapters using task-generic solver targets and evaluated free program generation with correct structures. Additionally, we compared four PAL conditions: None, Vision, Program, and Joint, with Joint combining separately trained visual components and program adapters. Training configurations and evaluation procedures are provided in Appendix C.

5.4 Results on MMCSP-Bench

The MMCSP-Bench reveals gaps across both solving routes and input views. On structured text, DeCSP achieves 96.24% mean valid-solution accuracy across seven LVLMs, compared with 8.19% for direct answering (Base) and 43.43% for PAL (Table 3(a)). However, this advantage does not fully transfer to visual inputs: DeCSP falls to 42.57% on standard images and 28.19% on OOD images. Since the matched views preserve the same source problems and constraints, reliable symbolic solving with a supplied structure does not ensure reliable solving when that structure must first be recovered from an image.

The drop from structured-text to image-input accuracy also varies across models. Qwen3.8-27B and Mistral-3.2-24B both reach 100.00% structured-text DeCSP accuracy, but only 81.67% and 13.33%, respectively, on standard images (Table 3(b)). Similar performance on supplied structures can therefore conceal large differences under visual inputs. These endpoint comparisons establish the performance gaps but do not identify whether failures arise during visual recovery or subsequent symbolic generation.

6 Analysis

The results on MMCSP-Bench reveal substantial performance gaps across input views, but endpoint accuracy alone does not identify where these failures arise. We therefore analyze visual recovery, symbolic generation, and stage-targeted interventions through controlled comparisons.

6.1 Recovery Quality and Final Validity Are Distinct

The Known-Program control tests the contribution of visual recovery by fixing the task-correct solver and original-problem verifier. Under this controlled setting, replacing source structures with standard-image transcriptions reduces valid-solution accuracy from 100.00% to 46.10%. This drop

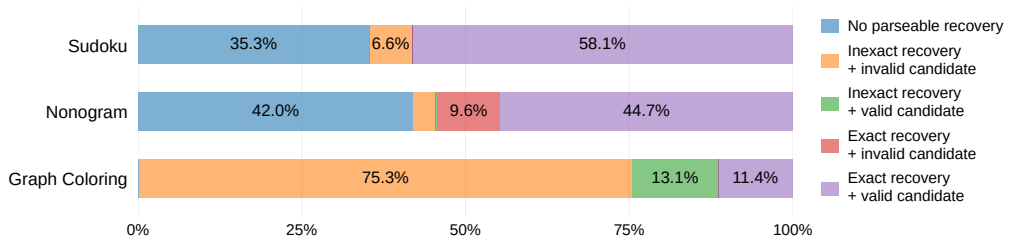

Figure 4: Standard-image DeCSP outcomes by task. Each bar contains 700 records from seven LVLMs and 100 source problems. Parseable recoveries are classified jointly by exact structure recovery and final validity against the original problem; unparseable recoveries are shown separately.

Table 4: Correct-structure valid-solution accuracy (%). Each task value averages the seven-LVLM panel. Extended full specification uses a 4,096-token budget.

Route	**Sudoku**	**Nonogram**	**Graph Coloring**	**Mean**
Canonical	100.00	100.00	100.00	**100.00**
DeCSP	100.00	88.71	100.00	**96.24**
CDT	71.43	83.86	94.71	**83.33**
PAL	86.71	1.00	42.57	**43.43**
Extended full specification	0.00	4.57	14.29	**6.29**

measures the effect of recovered inputs under fixed downstream solving: task-correct solver code alone does not ensure valid solutions to the original problems.

To further understand the relationship between recovery quality and final solving success, we analyze recovery correctness jointly with solution validity (Figure 4). The results show that exact recovery is neither necessary nor sufficient for achieving valid solutions in these evaluations. Among 2,100 standard-image DeCSP evaluations, 94 yield valid solutions after parseable but non-exact recovery, indicating that final success can conceal structural errors. Conversely, 67 recover the source structure exactly but still fail downstream, demonstrating that correct recovery supplies the right input without ensuring successful symbolic solving.

6.2 Symbolic Generation under Controlled Inputs

Removing visual uncertainty does not eliminate symbolic-generation failures. With the same original public structures, Canonical achieves 100.00% accuracy without model-generated specifications, compared with 96.24% for DeCSP, 83.33% for CDT, 43.43% for PAL, and 6.29% for Full Instance Specification (Table 4).

All five routes use external execution. PAL leaves both constraint implementation and search logic to the model. CDT and Full Instance Specification delegate search but still require generated declarative rules or concrete instance-level constraints. DeCSP also delegates constraint construction, restricting the model to predefined constraint-operator selection. External execution does not by itself ensure that the generated constraints or solver logic correctly represent the problem. Reliability also depends on how symbolic responsibilities are divided between the model and the backend.

To test whether the DeCSP–CDT ordering carries over to recovered inputs, we give both interfaces identical archived transcriptions and use the same declarative backend. Across 4,200 paired records covering standard and OOD images, CDT and DeCSP achieve 35.64% and 35.38% valid-solution accuracy, respectively, with a paired CDT-minus-DeCSP difference of 0.26 percentage points (95% interval: $[-0.40, 0.93]$). Model-level differences nevertheless point in opposite directions: CDT leads by 17.50 points on Qwen3.5-9B, whereas DeCSP leads by 11.67 points on Qwen3.5-4B. These opposing differences partially cancel in the aggregate, so the overall near-tie masks substantial model-dependent interface effects under shared recovered inputs (Appendix D.3).

Table 5: Stage-targeted training and transfer. (a) Visual-repair transfer and a PAL four-arm study (1,536-token program budget), each with its own untrained baseline; (b) correct-structure PAL training. Joint combines trained visual and program components; Exact denotes exact DeCSP structure recovery. Accuracies are percentages; changes and 95% bootstrap intervals are percentage points.

(a) Visual-repair transfer and PAL four-arm results

	Visual-repair transfer (Δ, pp)			PAL four-arm accuracy (%)			
LVLM	Base	DeCSP	Exact	None	Vision	Program	Joint
Qwen3.5-4B	+0.00	+75.11	+74.89	0.00	2.17	14.17	**72.61**
Qwen3.5-9B	−17.50	+27.22	+34.50	36.50	39.50	64.83	**77.67**
Qwen3-VL-32B	+1.39	+38.56	+36.28	22.83	34.83	46.67	**63.28**
Qwen3.8-27B	−2.78	+10.11	+16.50	33.00	28.44	**76.83**	75.44
Gemma3-27B	−4.89	+30.83	+25.17	6.67	17.56	16.00	**38.61**
Gemma4-12B	−0.44	+4.17	+5.39	30.50	**31.44**	12.56	13.06
Mistral-3.2-24B	+1.83	+64.39	+57.44	2.17	18.11	10.33	**50.50**
Mean	−3.20	+35.77	+35.74	**18.81**	**24.58**	**34.48**	**55.88**

(b) Correct-structure program training: separate PAL control

LVLM	Untrained PAL	Trained PAL	Change	95% interval
Qwen3-VL-32B	67.67	100.00	+32.33	$[27.00, 37.67]$
Gemma4-12B	52.33	35.78	−16.56	$[-24.00, -9.11]$

6.3 Stage-Targeted Training Transfers Selectively

We next investigate whether training targeted at recovery and symbolic generation improves final solving. In the visual-repair transfer study (Appendix C.2), exact structure recovery increases by 35.74 percentage points and DeCSP valid-solution accuracy by 35.77 points on average, while Base changes by -3.20 points (Table 5(a)). Because DeCSP explicitly uses the recovered structure, it improves across all seven LVLMs, whereas Base shows both improvements and degradations. These results indicate that optimizing an explicit recovery objective can substantially improve structure-dependent solving, but the improvements do not necessarily transfer to direct answer generation.

We then examine whether jointly training visual recovery and program generation provides more consistent improvements. In the PAL four-arm study, joint combines separately trained visual components and program adapters, raising mean accuracy from 18.81% under None to 55.88% and exceeding both single-stage conditions on average (Table 5(a)). Gemma4-12B, however, falls from 30.50% to 13.06%. This suggests that joint learning does not uniformly benefit all LVLMs.

Finally, to determine whether model-dependent training effects also occur without visual recovery, we evaluate PAL program training with correct source structures (Table 5(b)). Accuracy increases from 67.67% to 100.00% for Qwen3-VL-32B, but decreases from 52.33% to 35.78% for Gemma4-12B. These results show that model-dependent effects of program training persist even with correct input structures.

7 Conclusion

We have presented MMCSP-Bench and DeCSP to study how LVLMs transform visual inputs into symbolic solutions. First, MMCSP-Bench pairs structured text, standard images, and OOD images for the same Sudoku, Nonogram, and Graph Coloring problems, while verifying every candidate against the frozen original problem. This design enables separate evaluation of recovery quality and final validity. Second, DeCSP provides a controlled symbolic interface that disentangles operator selection from instance-specific constraint binding and search, allowing failures in perception and reasoning to be analyzed independently. Third, our controlled analyses and stage-targeted repairs show that failures arise across visual recovery and symbolic generation: a valid answer need not establish faithful problem recovery, and improvements to one component need not transfer across routes or models. Together, these findings highlight the importance of stage-aware evaluation, explicit intermediate representations, and original-problem verification for more reliable multimodal symbolic reasoning.

AI USE STATEMENT

We used generative AI tools for writing assistance, including improving manuscript clarity and organization, and for literature retrieval during related work preparation. We did not use AI tools to generate experimental results, construct benchmark data, perform evaluations, or draw scientific conclusions. All experiments, analyses, and interpretations were conducted and verified by the authors. We reviewed all AI-assisted content and take responsibility for the final manuscript.

ETHICS STATEMENT

This work does not involve human subjects, private user data, or sensitive personal information. MMCSP-Bench consists of synthetic constraint satisfaction problems, and all evaluations use automatically generated instances with deterministic verification. We study multimodal reasoning reliability in controlled symbolic environments and do not make claims about high-stakes deployment.

REPRODUCIBILITY STATEMENT

Anonymous code, benchmark data, and supplementary materials are available at `https://anonymous.4open.science/r/Seeing2Solving-37532`. The release includes MMCSP-BENCH construction and rendering scripts, matched structured-text, standard-image, and OOD-image views, evaluation code with frozen original-problem verification, and implementations of the Base, PAL, and DeCSP routes. The supplementary material documents model identifiers, prompting and decoding settings, diagnostic controls, training configurations, and statistical procedures. All reported results are derived from frozen source structures and checked by deterministic verifiers; intermediate records support reproduction of the stage-level analyses.

REFERENCES

Saeed Amizadeh, Hamid Palangi, Oleksandr Polozov, Yichen Huang, and Kazuhito Koishida. Neuro-symbolic visual reasoning: disentangling "visual" from "reasoning". In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL `https://arxiv.org/abs/2511.21631`.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Trans. Mach. Learn. Res.*, 2023, 2022. URL `https://api.semanticscholar.org/CorpusID:253801709`.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe

Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, Mayank Chaturvedi, Aditya Chawla, Victor Cotruta, Alice Coucke, Phil Culliton, Robert Dadashi, Lucas Dixon, Mohamed Elhawaty, Utku Evci, Clément Farabet, Johan Ferret, Filippo Galgani, Sertan Girgin, Jean-Bastien Grill, Maarten Grootendorst, Jiaxian Guo, Cassidy Hardin, Yanzhang He, Steven M. Hernandez, Omri Homburger, Léonard Hussenot, Juyeong Ji, Armand Joulin, Aishwarya Kamath, Parnian Kassraie, Olivier Lacombe, Preethi Lahoti, Gaël Liu, Gus Martins, Luciano Martins, Tatiana Matejovicova, Ramona Merhej, Nikola Momchev, Sneha Mondal, Ryan Mullins, Sindhu Raghuram Panyam, Shreya Pathak, Sarah Perrin, André Susano Pinto, Etienne Pot, Angéline Pouget, Alexandre Ramé, Sabela Ramos, Douglas Reid, David Rim, Morgane Rivière, Karsten Roth, Louis Rouillard, Omar Sanseviero, Pier Giuseppe Sessa, Shane Settle, Danila Sinopalnikov, Sara Smoot, Piotr Stanczyk, Andreas Steiner, Lawrence Stewart, Ilya Tolstikhin, Michael Tschannen, Anton Tsitsulin, Nino Vieillard, Renjie Wu, Pingmei Xu, Haichuan Yang, Edouard Yvinec, Biao Zhang, Li Zhang, Joe Zou, Nicolas Aagnes, Abdelrahman Abdelhamed, Jakub Adamek, Shivani Agrawal, Shubham Agrawal, Ibrahim Alabdulmohsin, Jean Baptiste Alayrac, Uri Alon, Chandramouli Amarnath, Ankesh Anand, Chrysovalantis Anastasiou, Setareh Ariafar, François-Xavier Aubet, Kyriakos Axiotis, Federico Barbero, Joelle Barral, Alexei Bendebury, Urs Bergmann, Stanley Bileschi, Kat Black, Mathieu Blondel, Sebastian Borgeaud, Arthur Bražinskas, Ryan Burnell, Robert Busa-Fekete, Mu Cai, Daniele Calandriello, Glenn Cameron, Charlotte Caucheteux, Rahma Chaabouni, Garima Chadha, Jetha Chan, Blake Jianhang Chen, Jesse Chen, Lin Chen, Xu Chen, Derek Cheng, Tzu hsiang Chien, Nikolai Chinaev, Yi Chou, Zhaohui Chu, Benjamin Coleman, Pooja Consul, Sam Conway-Rahman, Scott Crowell, Dylan Cutler, Vivek Dani, Samira Daruki, Anil Das, Daniel Deutsch, Nishanth Dikkala, Li Ding, Qiuhan Ding, Shenil Dodhia, Konstantin Donhauser, Tulsee Doshi, Anca Dragan, Alex Druinsky, Sahil Dua, Zoltan Egyed, Danielle Eisenbud, Daniel Eppens, Cindy Fan, Bahare Fatemi, Yassir Fathullah, Vlad Feinberg, Milen Ferev, Sebastian Flennerhag, Takumi Fujimoto, João Gabriel Oliveira, Isaac Galatzer-Levy, João Gante, Simon Geisler, Soham Ghosal, Antonious M. Girgis, Tamara von Glehn, Alec Go, Alhaad Gokhale, Alex Grills, Yiming Gu, Mayank Gupta, Pramod Gupta, Guru Guruganesh, Raia Hadsell, Hamza Harkous, Jitendra Harlalka, Demis Hassabis, Anja Hauth, Joe

Heyward, Arian Hosseini, Chih-Yang Hsia, I-Hung Hsu, Xiaopeng Huang, Yangsibo Huang, Kevin Hui, Adrian Hutter, Te I, Fotis Iliopoulos, Advait Jain, Ganesh Jawahar, Ziwei Ji, Qilin Jin, Melvin Johnson, Kandarp Joshi, Arun Kandoor, Wang-Cheng Kang, Koray Kavukcuoglu, Mehran Kazemi, Kathleen Kenealy, Amr Khalifa, Phoebe Kirk, Ivan Korotkov, Suraj Kothawade, Vitaly Kovalev, Neel Kovelamudi, Adam Kraft, Ravin Kumar, Vivek Kumar, Harish Kuppam, Justin Lannin, Chen-Yu Lee, Seungji Lee, Dmitry Lepikhin, Alon Levkovitch, Dongdong Li, Qiujia Li, Valentin Liévin, Ethan Lin, Ziqian Lin, Casper Liu, Tianlin Liu, Tianqi Liu, Xin Liu, Ivan Lobov, Mayank Lunayach, Min Ma, Gagan Madan, Andrii Maksai, Eric Malmi, Michal Matuszak, Daniel McDuff, Gaurav Menghani, Maciej Mikuła, Daniil Mirylenka, Karolis Misiunas, Vedant Misra, Andreea Mitran, Kareem Mohamed, Maksim Mukha, Eric Noland, James O'Donnell, Brendan O'Donoghue, Kate Olszewska, Bernett Orlando, Wanqiong Pan, Rina Panigrahy, Unnati Parekh, Nicolas Perez-Nieves, Chunjong Park, Eric Paskie, Liqian Peng, Bryce Petrini, Slav Petrov, Jonas Pfeiffer, Bilal Piot, Martyna Plomecka, Siim Poder, Octavio Ponce, Arijit Pramanik, David Racz, Anish Rajan, Michelle Ramanovich, Anand Rao, Marvin Ritter, Vitor Rodrigues, Evan Rosen, Mikołaj Rybiński, Noveen Sachdeva, Michaël E. Sander, Rohit Sathyanarayana, Sagar Savla, Samuel Schmidgall, Tal Schuster, George Scrivener, Benoit Seguin, Andrew Sellergren, Aliaksei Severyn, Izhak Shafran, Dhruv Shah, Bobak Shahriari, Yuan Shangguan, Ashish Shenoy, Pradeep Shenoy, Rakesh Shivanna, Pauline Sho, Lucas Spangher, Wojciech Stokowiec, Tim Strother, Yao Su, Yinghao Sun, Mukund Sundararajan, Andrea Tacchetti, Mor Hazan Taege, Pouya Tafti, Jean Tarbouriech, Chetan Tekur, Shantanu Thakoor, Rahul Thapa, Madeleine Traverse, Lenart Treven, Tao Tu, Chien Te Tung, Çağlar Ünlü, Petar Veličković, Malini Pooni Venkat, Sagar Gubbi Venkatesh, Vidya Venkiteswaran, Francesco Visin, Alex Vitvitskyi, Kiran Vodrahalli, Weiyi Wang, Xin Wang, Tris Warkentin, Jan Wassenberg, John Wieting, Cindy Wu, Lechao Xiao, Hao Xu, Yuhui Xu, Fuzhao Xue, Arun Yadav, Jun Yan, Antoine Yang, Lin Yang, Ming-Hsuan Yang, Ziyu Ying, Jae Hyeon Yoo, Morteza Zadimoghaddam, Sajjad Zafar, Fred Zhang, Jiageng Zhang, Jianyi Zhang, Xiaofan Zhang, Chao Zhao, David Zhou, and Chen Zou. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured NLP tasks without finetuning. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10932–10952, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.674. URL https://aclanthology.org/2023.emnlp-main.674/.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui

Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://doi.org/10.1038/s41586-025-09422-z.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14953–14962, 2023. doi: 10.1109/CVPR52729.2023.01436.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Jixuan Leng, Chengsong Huang, Langlin Huang, Bill Yuchen Lin, William W. Cohen, Haohan Wang, and Jiaxin Huang. Crosswordbench: Evaluating the reasoning capabilities of llms and lvlms with controllable puzzle generation, 2025. URL https://arxiv.org/abs/2504.00043.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. Zebralogic: on the scaling limits of llms for logical reasoning. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025.

Zhining Liu, Ziyi Chen, Hui Liu, Chen Luo, Xianfeng Tang, Suhang Wang, Jingying Zeng, Zhenwei Dai, Zhan Shi, Tianxin Wei, Hanqing Lu, Benoit Dumoulin, and Hanghang Tong. Seeing but not believing: Probing the disconnect between visual attention and answer correctness in vlms. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 72849–72876, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/76818d8d85e05e45ce3a16a8468619d1-Paper-Conference.pdf.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=KUNzEQMWU7.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3806–3824, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.248. URL https://aclanthology.org/2023.findings-emnlp.248/.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.8-max: A new bar for coding and cowork, August 2026b. URL https://qwen.ai/blog?id=qwen3.8.

Yufan Ren, Konstantinos Tertikas, Shalini Maiti, Junlin Han, Tong Zhang, Sabine Süsstrunk, and Filippos Kokkinos. Vgrp-bench: Visual grid reasoning puzzle benchmark for large vision-language models, 2025. URL https://arxiv.org/abs/2503.23064.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 9895–9901, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.779. URL https://aclanthology.org/2021.emnlp-main.779/.

Jeffrey Seely, Yuki Imajuku, Tianyu Zhao, Edoardo Cetin, and Llion Jones. Sudoku-bench: Evaluating creative reasoning with sudoku variants, 2025. URL https://arxiv.org/abs/2505.16135.

Dídac Surís, Sachit Menon, and Carl Vondrick. ViperGPT: Visual Inference via Python Execution for Reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. URL https://arxiv.org/abs/2303.08128.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Weiye Xu, Jiahao Wang, Weiyun Wang, Zhe Chen, Wengang Zhou, Aijun Yang, Lewei Lu, Houqiang Li, Xiaohua Wang, Xizhou Zhu, Wenhai Wang, Jifeng Dai, and Jinguo Zhu. Visulogic: A benchmark for evaluating visual reasoning in multi-modal large language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=mXuzDDVXxi.

Hao Yan, Xingchen Liu, Hao Wang, Zhenbiao Cao, Handong Zheng, Liang Yin, Xinxing Su, Zihao Chen, Jihao Wu, Minghui Liao, CHAO WENG, Wei Chen, Yuliang Liu, and Xiang Bai. Visuriddles: Fine-grained perception is a primary bottleneck for multimodal large language models in abstract visual reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=fGRwRnDVMX.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. Satlm: satisfiability-aided language models using declarative prompting. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B. Tenenbaum. Neural-symbolic vqa: disentangling reasoning from vision and language understanding. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 1039–1050, Red Hook, NY, USA, 2018. Curran Associates Inc.

Table 6: Primary MMCSP-Bench evaluation composition.

Task	Problems	Matched input views	Original-problem verifier
Sudoku	100	Structured text, standard image, OOD image	Givens, rows, columns, and 3×3 boxes
Nonogram	100	Structured text, standard image, OOD image	Row and column run clues
Graph Coloring	100	Structured text, standard image, OOD image	Original edges and color budget

A Additional Benchmark Construction Details

Sudoku instances are generated from valid 9×9 completed grids after digit, row-band, and column-stack permutations. Each puzzle removes 42–51 digits, leaving 30–39 givens. Nonogram instances alternate between 5×5 and 6×6 binary grids, derive ordered row and column run clues, and exclude the completed binary grid from model inputs. Graph Coloring instances use balanced hidden three-color assignments, connected graph structures, and edge densities spanning sparse to dense conditions. Node positions are independent of hidden colors and node labels.

Table 6 summarizes the task-specific composition and original-problem verification criteria of the matched primary evaluation.

A.1 OOD Rendering Details

In the reported image evaluations, the standard-image view uses the `legible` renderer for all three tasks. The OOD-image view uses a task-specific alternate renderer for the same source problem while preserving its public constraint structure.

Sudoku. The `jitter` renderer preserves the givens and deterministically shifts grid-line locations by up to two pixels. It also changes digit placement and font relative to the centered `legible` template.

Nonogram. The `thin` renderer preserves the row and column clues. Compared with `legible`, it uses thinner grid strokes, default-font clue text, and shifted clue offsets. It also omits the guide lines used by the standard template.

Graph Coloring. The `bold` renderer preserves the nodes, edges, circular layout, and color budget. It changes node radius and node-label typography and placement.

Across all three tasks, the OOD view changes only rendering attributes; the source problem, public structure, and original-problem verifier remain unchanged.

B Experimental Protocols

Table 7 organizes the reported experiments by the controlled input, solving route, and diagnostic question. Tables 8 and 10 provide the corresponding generation budgets, execution settings, and training configurations.

B.1 Inference Settings and Output Contracts

All primary evaluations use greedy decoding. Base, PAL, and DeCSP use a maximum of 1,024 generated tokens for each model call in the primary benchmark. For image inputs, Base maps the rendered image directly to a candidate answer. PAL and DeCSP use a transcription call that receives the source image and task-specific public-structure schema and returns structured JSON enclosed in `<INSTANCE>` tags; up to three transcription attempts precede one downstream PAL or DeCSP call. Thinking is disabled for Qwen3.5-4B, Qwen3.5-9B, and Qwen3.8-27B; the remaining checkpoints use their released chat-template settings.

Table 7: Experiment protocol overview. Each study uses the frozen original-problem verifier. Generation budgets, output contracts, and training configurations are reported in Tables 8 and 10.

Study	Models	Input and route	Question addressed
Evaluation			
Main benchmark	7 LVLMs	Matched structured text, standard image, and OOD image; Base, PAL, and DeCSP	How do input view and symbolic interface affect end-to-end validity?
Diagnosis			
Known-Program control	7 LVLMs	Correct source structure x or recovered structure $\hat{x}$; fixed task solver	How much downstream loss follows image-to-structure recovery?
Correct-structure interface comparison	7 LVLMs	Correct public structure; Canonical, DeCSP, CDT, PAL, and Full Instance Specification	How do symbolic output interfaces differ after visual recovery is fixed?
Shared-transcription comparison	7 LVLMs	Identical recovered structure $\hat{x}$; CDT and DeCSP	Does an interface difference remain when both routes receive the same image transcription?
Unseen-wording study	7 LVLMs	Unseen rule descriptions; DeCSP operator selection	Can operator selection transfer to controlled unseen rule wording?
Improvement			
Visual-repair transfer	7 LVLMs	Standard and OOD images; Base and DeCSP with visual repair	Does improved image-to-structure recovery transfer to native solving routes?
PAL four-arm repair	7 LVLMs	Standard and OOD images; PAL with None, Vision, Program, and Joint conditions	What do visual and program repair contribute separately and jointly?
Correct-structure program repair	2 LVLMs	Correct public structure; PAL with program repair	Does program repair change free solver generation when visual recovery is removed?
Auxiliary rule study			
Text-only variable-rule study	1 LLM	Variable-rule text cards; PAL and DeCSP with feedback variants	Can rule selection and feedback-based repair support changing task definitions?

Base receives a problem and a direct-answer output contract. PAL receives the task name, public structure, and a contract requiring a complete Python `solve(instance)` function. Generated programs are parsed, audited, executed in an isolated environment, and then checked against the frozen original problem.

DeCSP receives the task name, public schema, one public instance, and the seven predefined constraint operators listed in Table 9. Its output is a versioned JSON list of selected operators. The deterministic backend binds values from the supplied structure, constructs the selected constraints, performs bounded search, and verifies the resulting candidate against the original problem.

CDT generates a declarative abstract syntax tree containing variable declarations, domains, quantified constraints, and output variables. Its grammar supports general collections and predicates, including rows, columns, boxes, edges, equality, inequality, `all_different`, and `runs`. Full Instance Specification additionally requires concrete instance-level constraints. Canonical inserts the task-correct declarative representation without a model generation step.

B.2 Diagnostic Controls

Known-Program holds the task-correct solver and frozen original-problem verifier fixed while replacing the correct source structure x with the model-recovered structure $\hat{x}$. This comparison measures how replacing the source structure with a recovered structure affects valid-solution accuracy under the fixed solver.

Table 8: Generation and execution settings. Token limits apply to each model call.

Route or control	Tokens	Model-generated output	Execution setting
Base	1,024	Candidate solution	Original-problem verification
Image transcription	1,024	Public structure JSON	Up to three transcription attempts
Primary PAL	1,024	Python solver program	2 CPU seconds; 3 wall seconds; 512 MiB
DeCSP	1,024	Operator names	5 seconds; 500,000 search nodes
CDT	512	Reusable declarative rules	Shared declarative backend
Full specification	1,024	Concrete constraints	Shared declarative backend
Extended full specification	4,096	Concrete constraints	Shared declarative backend
Unseen-wording selection	192	Operator names	Held-out deterministic execution
PAL repair evaluation	1,536	Python solver program	Same PAL execution setting

Table 9: Predefined DeCSP operators. The deterministic backend binds instance values from the supplied public structure.

Task	Operator	Requirement
Sudoku	`givens`	Preserve supplied digits.
Sudoku	`row_all_different`	Require unique digits in every row.
Sudoku	`column_all_different`	Require unique digits in every column.
Sudoku	`box_all_different`	Require unique digits in every 3×3 box.
Nonogram	`row_runs`	Match every row clue.
Nonogram	`column_runs`	Match every column clue.
Graph Coloring	`edge_not_equal`	Require different colors at the endpoints of every edge.

The correct-structure interface comparison supplies the original public structure to every route. Canonical, DeCSP, CDT, and Full Instance Specification share the declarative backend; PAL executes a model-generated solver program. This design compares the symbolic-generation responsibilities imposed by each interface after visual recovery has been removed.

The shared-transcription comparison passes each archived DeCSP recovered structure to CDT without an additional vision call. CDT and DeCSP therefore receive the same image-derived public structure and use the same declarative backend. Failed transcription units remain in the denominator of both routes.

A separate operator-alias replay normalizes `col_runs` to `column_runs` in frozen correct-structure DeCSP Nonogram responses and reruns the same compiler, solver, and verifier. This replay yields 79 failure-to-valid transitions and provides a direct measurement of interface-name sensitivity.

B.3 Statistical Procedures

First-failure records identify the earliest unsuccessful check along a route. Shared-transcription intervals use percentile bootstrap resampling over source-problem clusters while retaining paired interface outcomes and both image views. Visual-transfer intervals use task-stratified paired bootstrap resampling over original problems. Correct-structure program-repair intervals resample original problems while retaining paired untrained and trained outcomes. Unseen-wording intervals resample the 22 target rule sets together with their associated language renderings and model outputs. All reported intervals use 10,000 bootstrap draws.

Table 10: Stage-training configuration. Both procedures use AdamW (Loshchilov & Hutter, 2019).

Setting	Visual training	Program training
Training-pool examples	768	768
Optimization presentations per run	768	256
Optimizer updates	96	64
Micro-batch size	1	1
Gradient accumulation	8	4
Learning rate	2×10^{-5}	2×10^{-4}
Weight decay	0.01	0.00
Maximum sequence length	1,536	2,048
Precision	bfloat16	bfloat16
Learning-rate schedule	Constant	Constant
Updated component	Visual module	LoRA in final eight language layers
LoRA rank / alpha / dropout	—	8 / 16 / 0.05

C Stage-Targeted Training and Transfer

C.1 Training Data and Updated Components

The visual-training pool contains 768 examples, with 256 examples for each task. The program-training pool also contains 768 examples: 256 Sudoku examples and 512 examples pooled across Nonogram and Graph Coloring. Each program-adapter training run samples 256 optimization presentations from this pool. Both pools are generated independently of the primary-test instances.

Visual training maps an image and transcription instruction to canonical public JSON, updating only the architecture-specific visual component. Program training maps public JSON and the PAL instruction to an audited task-generic `solve(instance)` program, updating low-rank adapters in the final eight language layers while keeping the base model frozen (Hu et al., 2022).

The PAL four-arm study combines the original or trained visual component with the original language model or program adapter. The None condition uses the original visual and language states, Vision updates only the visual component, Program activates only the program adapter, and Joint combines both trained components. The program adapter is active only during program generation after image transcription has completed.

C.2 Visual-Repair Transfer

The visual-repair transfer results in Table 5(a) are computed from the dedicated archived record set for the `visual-repair-route-transfer-20260918-v1` protocol. Within this protocol, the unmodified and repaired visual states are paired on the same 100 source instances per task. The Graph Coloring instances use the archived `graph-s20260813-*` identifiers. The transfer baseline is evaluated on these records, rather than imported from the primary benchmark.

For each model, task, view, and route, we compare the unmodified visual component with three independently trained visual components. The source instance, rendered input, route implementation, parser, and original-problem verifier remain fixed within each comparison; only the visual component state changes. Base generates a candidate directly from the image, whereas DeCSP first recovers the public structure and then selects constraint operators. No PAL program adapter is loaded. Generation uses greedy decoding with a 1,024-token limit per model call. DeCSP allows up to three transcription attempts before operator selection.

Standard images use the `legible` renderer, and OOD images use the task-specific renderers in Appendix A.1. During replay, we check record coverage, image hashes, public-structure hashes, and the absence of a program adapter. Candidate solutions are checked against their corresponding original problems.

Table 11: Base validity (%) on the dedicated visual-repair transfer record set. The transfer baseline uses the unmodified visual state on the same instances as the repaired states. Changes are paired repaired-minus-unmodified differences; changes and 95% intervals are in percentage points.

LVLM	Transfer baseline (%)	Visual repair (%)	Change (pp)	95% interval (pp)
Qwen3.5-4B	0.00	0.00	$+0.00$	$[0.00, 0.00]$
Qwen3.5-9B	17.67	0.17	-17.50	$[-20.39, -14.61]$
Qwen3-VL-32B	0.00	1.39	$+1.39$	$[0.94, 1.89]$
Qwen3.8-27B	25.83	23.06	-2.78	$[-5.28, -0.33]$
Gemma3-27B	14.17	9.28	-4.89	$[-7.17, -2.50]$
Gemma4-12B	2.50	2.06	-0.44	$[-1.72, 0.72]$
Mistral-3.2-24B	2.33	4.17	$+1.83$	$[0.33, 3.28]$

Table 12: DeCSP validity and exact recovery (%) on the dedicated visual-repair transfer record set. The transfer baseline uses the unmodified visual state on the same instances as the repaired states. Changes are paired repaired-minus-unmodified differences; changes and 95% intervals are in percentage points.

LVLM	Metric	Transfer baseline (%)	Visual repair (%)	Change (pp)	95% interval (pp)
Qwen3.5-4B	Validity	12.33	87.44	$+75.11$	$[71.94, 78.11]$
	Exact recovery	8.50	83.39	$+74.89$	$[71.94, 77.72]$
Qwen3.5-9B	Validity	41.00	68.22	$+27.22$	$[24.44, 30.00]$
	Exact recovery	53.50	88.00	$+34.50$	$[31.67, 37.39]$
Qwen3-VL-32B	Validity	43.33	81.89	$+38.56$	$[35.22, 41.94]$
	Exact recovery	41.17	77.44	$+36.28$	$[33.06, 39.50]$
Qwen3.8-27B	Validity	78.83	88.94	$+10.11$	$[6.94, 13.33]$
	Exact recovery	70.67	87.17	$+16.50$	$[13.50, 19.56]$
Gemma3-27B	Validity	16.00	46.83	$+30.83$	$[27.56, 34.00]$
	Exact recovery	14.33	39.50	$+25.17$	$[22.33, 28.00]$
Gemma4-12B	Validity	34.67	38.83	$+4.17$	$[2.39, 6.00]$
	Exact recovery	27.00	32.39	$+5.39$	$[3.50, 7.44]$
Mistral-3.2-24B	Validity	8.33	72.72	$+64.39$	$[60.94, 67.78]$
	Exact recovery	8.00	65.44	$+57.44$	$[54.28, 60.50]$

Repair accuracies average the three trained visual components. Reported changes are paired repaired-minus-unmodified differences within this transfer record set, with 95% intervals obtained by task-stratified paired bootstrap resampling over source instances. Tables 11 and 12 report the Base and DeCSP results, respectively.

C.3 Correct-Structure Program Repair

The correct-structure PAL control supplies the correct public structure to Qwen3-VL-32B-Instruct and Gemma4-12B. The original model and three independently trained program-adapter instances freely generate one `solve(instance)` program per problem while program auditing, execution limits, and original-problem verification remain fixed. This control measures program-generation repair after removing image-to-structure recovery. The resulting accuracy changes and paired intervals are reported in Table 5(b).

C.4 PAL Four-Arm Results

Table 13 expands the PAL repair results by independently trained run. Each value averages the three tasks and two image views for one model and one repair arm.

Table 13: PAL accuracy (%) across independently trained runs. SD is the sample standard deviation across the three runs.

LVLM	Arm	Run accuracy (%)			Summary	
		Run 1	Run 2	Run 3	Mean	SD
Qwen3.5-4B	Vision	2.33	2.50	1.67	**2.17**	0.44
	Program	14.17	14.17	14.17	**14.17**	0.00
	Joint	75.83	76.33	65.67	**72.61**	6.02
Qwen3.5-9B	Vision	39.17	41.50	37.83	**39.50**	1.86
	Program	64.83	64.83	64.83	**64.83**	0.00
	Joint	76.67	78.83	77.50	**77.67**	1.09
Qwen3-VL-32B	Vision	36.83	31.83	35.83	**34.83**	2.65
	Program	46.67	46.67	46.67	**46.67**	0.00
	Joint	67.00	60.17	62.67	**63.28**	3.46
Qwen3.8-27B	Vision	27.83	22.67	34.83	**28.44**	6.11
	Program	76.83	76.83	76.83	**76.83**	0.00
	Joint	78.00	62.50	85.83	**75.44**	11.87
Gemma3-27B	Vision	16.67	17.17	18.83	**17.56**	1.13
	Program	16.00	16.00	16.00	**16.00**	0.00
	Joint	43.67	38.33	33.83	**38.61**	4.92
Gemma4-12B	Vision	32.00	30.50	31.83	**31.44**	0.82
	Program	10.50	10.50	16.67	**12.56**	3.56
	Joint	10.50	11.17	17.50	**13.06**	3.86
Mistral-3.2-24B	Vision	16.83	18.67	18.83	**18.11**	1.11
	Program	10.33	10.33	10.33	**10.33**	0.00
	Joint	49.83	52.83	48.83	**50.50**	2.08

Table 14: CDT correct-structure accuracy (%) by LVLM and task.

LVLM	Sudoku	Nonogram	Graph Coloring	Mean
Qwen3.5-4B	0.00	0.00	67.00	**22.33**
Qwen3.5-9B	100.00	100.00	100.00	**100.00**
Qwen3-VL-32B	100.00	100.00	100.00	**100.00**
Qwen3.8-27B	100.00	100.00	100.00	**100.00**
Gemma3-27B	0.00	99.00	100.00	**66.33**
Gemma4-12B	100.00	88.00	100.00	**96.00**
Mistral-3.2-24B	100.00	100.00	96.00	**98.67**
Fixed-panel mean	**71.43**	**83.86**	**94.71**	**83.33**

D Supplementary Diagnostic Results

D.1 Correct-Structure Interface Comparison

Table 14 provides the model- and task-level CDT breakdown for the correct-structure comparison in Section 6.2.

Among unsuccessful CDT outputs, rule-content errors account for 300 cases: Gemma3-27B omits the Sudoku givens rule in 100 cases, and Qwen3.5-4B selects an incorrect task-rule family in 100 Sudoku and 100 Nonogram cases. The remaining 50 outputs contain version or unsupported-declaration errors.

Table 15: Joint recovery and validity outcomes for standard-image DeCSP. Parsed and exact recovery are computed over all records. The final two columns condition on exact and parseable non-exact recovery, respectively.

Task	Recovery (%)		Validity given recovery (%)	
	Parsed	Exact	Exact	Parseable non-exact
Sudoku	64.71	58.14	100.00	0.00
Nonogram	58.00	54.29	82.37	7.69
Graph Coloring	99.86	11.43	100.00	14.86
All tasks	**74.19**	**41.29**	**92.27**	**13.60**

Table 16: Valid outcomes and first-failure categories for standard-image routes. Values are percentages within each route.

Outcome or first unsuccessful check	PAL (%)	DeCSP (%)
Valid candidate	22.57	42.57
Transcription parse	25.81	25.81
Program or interface check	25.29	3.19
Execution	1.38	15.62
Candidate parse	1.67	0.00
Invalid candidate	23.29	12.81

D.2 Recovery, Validity, and First-Failure Records

Table 15 provides the task-level recovery–validity breakdown underlying the aggregate analysis in Section 6.1.

Table 16 summarizes valid outcomes and first-failure categories for the standard-image PAL and DeCSP routes. Both routes have a 25.81% transcription-parse failure rate, but differ in their downstream outcome distributions.

PAL produces truncated solver programs in 176 standard-image calls following a parseable transcription. Among runs with exact transcription and successful execution, PAL still produces invalid candidates in 135 standard-image and 82 OOD-image cases. Program execution therefore provides an observable intermediate stage but does not by itself establish original-problem validity.

D.3 Shared-Transcription Comparison

Tables 17, 18, and 19 report the aggregate, task-level, and model-level results for the shared-transcription control discussed in Section 6.2.

E Rule-Selection Controls and Auxiliary Results

E.1 Fixed-Task Routing and Unseen Wording

The fixed benchmark protocol makes task identity informative for operator selection. We therefore distinguish closed-task routing from language-conditioned rule selection. A deterministic task router selects the task-specific operator set from task metadata and serves as a fixed-task baseline.

The unseen-wording study evaluates a separate capability: selecting operators from rule descriptions whose wording differs from the frozen original cards. It contains six linguistic renderings of each of 22 target rule sets, including 16 Sudoku subsets, four Nonogram subsets, and two Graph Coloring subsets. The 132 descriptions contain no exact overlap with the original cards and no literal machine operator names.

Table 17: Shared-transcription accuracy (%) and paired CDT-minus-DeCSP differences in percentage points.

View	CDT (%)	DeCSP (%)	Difference (pp)	95% interval (pp)
Standard image	42.52	42.57	-0.05	$[-0.90, 0.86]$
OOD image	28.76	28.19	$+0.57$	$[-0.10, 1.24]$
Both views	35.64	35.38	$+0.26$	$[-0.40, 0.93]$

Table 18: Shared-transcription results by task. Recovery and validity rates are percentages; Difference is CDT minus DeCSP in percentage points.

Task	Recovery (%)		Validity (%)		Difference (pp)
	Parsed	Exact	CDT	DeCSP	
Sudoku	55.50	46.50	44.86	46.71	-1.86
Nonogram	50.43	43.14	39.71	35.64	$+4.07$
Graph Coloring	99.57	11.64	22.36	23.79	-1.43

The frozen construction targets define the evaluation labels. Exact-set recovery measures agreement with these target sets, whereas held-out validity measures whether the selected operators yield valid solutions on held-out instances. Each LVLM makes one greedy operator-set selection per card, and each selected set is executed on held-out instances. Minimal-pair consistency evaluates whether changing one target operator produces the corresponding predicted-set change across 222 predefined single-operator minimal pairs.

Table 20 compares the rule-selection baselines with the seven-LVLM panel, which reaches 63.42% exact operator-set recovery with a target-cluster bootstrap interval of $[55.74, 71.21]$. The selected operator sets achieve 90.80% held-out execution validity, showing that held-out solution validity and exact operator-set recovery measure different outcomes. Table 21 reports model-level outcomes, and Table 22 breaks down exact recovery by linguistic phenomenon.

E.2 Information-Matched PAL and Specification Budgets

The information-matched PAL control inserts the global DeCSP operator descriptions immediately before the native PAL output contract. Table 23 reports the paired result. The aggregate PAL validity changes from 43.43% to 41.00%, corresponding to a paired difference of -2.43 percentage points.

With a 4,096-token budget, Full Instance Specification reaches 6.29% aggregate valid-solution accuracy. Its valid outputs occur in Nonogram and Graph Coloring, while no Sudoku instance is solved by the fixed seven-LVLM panel under this interface.

E.3 Separate Text-Only Auxiliary Study

Mistral-Small-24B-Instruct-2501 is used in a separate text-only variable-rule study. It receives no image input and selects operators or generates solver programs from changing rule descriptions. Table 24 summarizes aggregate validity and exact operator-set recovery; Table 25 provides the no-feedback task-level breakdown. On the fixed initial-error subset, the model receives a public counterexample in a compact-witness repair prompt. One additional call recovers the construction-defined target operator set in 81.82% of these cases. The one-shot rate is computed over all rule cards, whereas the compact-witness rate is computed only over the fixed initial-error subset; the two percentages therefore use different denominators.

Table 19: Shared-transcription results by LVLM. Recovery and validity rates are percentages; Difference is CDT minus DeCSP in percentage points. W/T/L denotes paired CDT wins, ties, and losses.

LVLM	Recovery (%)		Validity (%)		Difference (pp)	W/T/L
	Parsed	Exact	CDT	DeCSP		
Qwen3.5-4B	43.83	11.00	2.50	14.17	−11.67	0/530/70
Qwen3.5-9B	90.33	58.50	64.00	46.50	+17.50	105/495/0
Qwen3-VL-32B	75.33	40.50	46.83	46.67	+0.17	1/599/0
Qwen3.8-27B	97.67	72.50	77.00	77.00	+0.00	0/600/0
Gemma3-27B	56.17	13.50	11.17	15.17	−4.00	2/572/26
Gemma4-12B	63.50	31.83	38.17	38.17	+0.00	0/600/0
Mistral-3.2-24B	52.67	8.50	9.83	10.00	−0.17	0/599/1

Table 20: Aggregate unseen-wording rule-selection results. Exact-set recovery is defined against frozen construction targets.

Selector	Exact-set recovery (%)	Selection failure (%)	Minimal-pair consistency (%)
Task router	13.64	0.00	0.00
TF–IDF nearest neighbor	13.64	0.00	1.35
Frozen deterministic parser	0.00	100.00	0.00
Seven LVLMs	63.42	7.25	57.01

Table 21: Unseen-wording operator-selection results. Exact-set recovery compares each selected operator set with its construction-defined target.

LVLM	Exact recovery (%)	Selection failure (%)	Held-out validity (%)	Minimal-pair consistency (%)
Qwen3.5-4B	59.85	0.00	98.48	45.50
Qwen3.5-9B	34.85	38.64	61.36	29.28
Qwen3-VL-32B	59.09	0.00	98.48	48.20
Qwen3.8-27B	90.91	0.00	100.00	91.89
Gemma3-27B	64.39	7.58	89.39	52.25
Gemma4-12B	71.97	3.79	95.45	67.12
Mistral-3.2-24B	62.88	0.76	92.42	64.86

Table 22: Exact operator-set recovery by linguistic phenomenon.

Linguistic phenomenon	Exact recovery (%)
Lexical substitution	58.44
Pairwise quantification	60.39
Contrast and scope	90.91
Negated obligation	49.35
Ordinal anaphora	79.87
Discourse-list reference	41.56

Table 23: Information-matched PAL control. Accuracies are percentages; differences and intervals are percentage points.

LVLM	Original (%)	Augmented (%)	Difference (pp)	95% interval (pp)
Qwen3.5-4B	3.00	16.33	$+13.33$	$[9.67, 17.00]$
Qwen3.5-9B	49.00	33.33	-15.67	$[-19.67, -11.67]$
Qwen3-VL-32B	67.67	39.33	-28.33	$[-31.33, -25.33]$
Qwen3.8-27B	34.33	33.33	-1.00	$[-2.33, 0.00]$
Gemma3-27B	66.67	66.67	$+0.00$	$[0.00, 0.00]$
Gemma4-12B	49.67	49.33	-0.33	$[-4.67, 4.33]$
Mistral-3.2-24B	33.67	48.67	$+15.00$	$[11.67, 18.33]$
Mean	**43.43**	**41.00**	$\mathbf{-2.43}$	$\mathbf{[-3.62, -1.24]}$

Table 24: Aggregate text-only variable-rule results for Mistral-Small-24B-Instruct-2501. One-shot recovery is evaluated over all rule cards; feedback repair is evaluated only over the fixed initial-error subset. All values are percentages.

Metric	Evaluation set	Result (%)
PAL valid-solution accuracy	Held-out instances	27.27
DeCSP valid-solution accuracy	Held-out instances	90.91
One-shot exact operator-set recovery	All rule cards	50.00
Exact operator-set recovery after compact-witness feedback	Fixed initial-error subset	81.82

Table 25: Text-only variable-rule auxiliary results.

Task	Validity (%)		Exact operator-set recovery (%)
	PAL	DeCSP	
Sudoku	37.50	87.50	50.00
Nonogram	0.00	100.00	50.00
Graph Coloring	0.00	100.00	50.00