NEURO-SYMBOLIC GROUNDING FOR LATENT VISUAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Neuro-symbolic learning combines flexible neural perception with structured, executable reasoning. Yet systems that place a predefined executor on the inference path bind their answers to a fixed ontology, limiting integration with open-ended vision–language models. We introduce *neuro-symbolic supervision for latent reasoning*: a framework that trains over continuous reasoning states by grounding them as shared entities and executable predicates. Symbolic losses thereby shape the latent trajectory from which the original VLM backbone generates its answer. This brings compositional supervision into latent reasoning while allowing the decoder to answer beyond the symbolic vocabulary. We instantiate the framework in SYLVA through local symbolic grounding, compositional symbolic grounding, and reinforcement through discrete symbolic execution. The final stage pairs each sampled text with one symbolic scene and optimizes both using answer and execution rewards. The symbolic path is used only during training. With four latent steps, SYLVA reaches a 72.53% mean across six benchmarks, achieving state-of-the-art performance.

1 INTRODUCTION

Neuro-symbolic learning combines adaptive neural perception with explicit entities, relations, and executable rules. These structures provide compositional constraints beyond answer supervision, an opportunity for vision–language models (VLMs) whose multi-step reasoning capabilities have grown while their internal computations remain underspecified (Liu et al., 2023b; Bai et al., 2025). We study how symbolic supervision can shape latent visual reasoning.

Existing systems typically use an explicit program to produce answers: NS-VQA executes over recovered scenes (Yi et al., 2018), NS-CL learns concepts through an executor (Mao et al., 2019), and NeSyCoCo composes differentiable concept scores (Kamali et al., 2025). Their inference depends on a predefined ontology and operator set. We instead use symbolic execution to supervise latent states during training, retaining the original VLM decoder at inference. The symbolic vocabulary then defines the supervision, without restricting the decoder's answer space.

Latent reasoners compute through continuous hidden states before decoding an answer (Li et al., 2025a; Wang et al., 2025a; 2026a). Their trajectories need not preserve entity identity: two locally plausible steps may refer to different objects, making their predicates inconsistent when composed. Localizing individual states is therefore insufficient. Their grounded interpretations must also resolve cross-step references and support execution over a shared scene.

We introduce SYLVA, *Symbolically Grounded Latent Visual Reasoning*. Given training-time entity annotations, answers, and programs, it constructs a typed scene valuation from the backbone's latent trajectory and visual-token states. Entity–concept binding supervises the scene semantics; program execution checks whether the grounded facts support the answer. Both losses train the states used by the neural decoder. At evaluation, this symbolic path is removed, and the decoder answers without programs, scene annotations, or an executor-defined vocabulary.

Training proceeds in three stages. First, **local symbolic grounding** connects each latent step to subject/object visual evidence and a typed relation. Second, **compositional symbolic grounding** groups repeated references across all latent steps into shared entity slots, binds them to annotated concepts, and executes the question's program over their predicted relations. Finally, **reinforcement**

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.

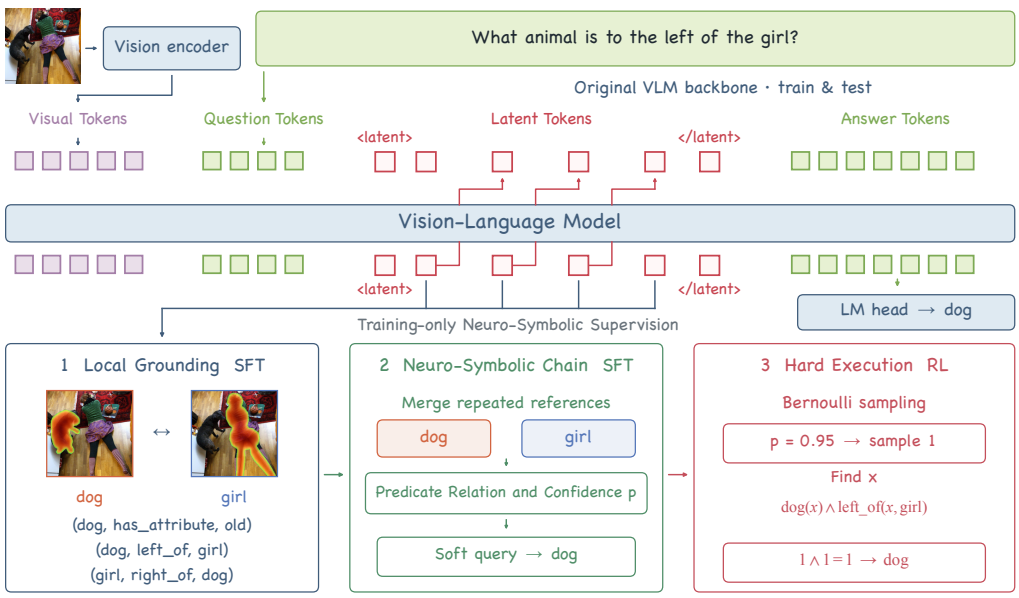

Figure 1: SYLVA overview. Training grounds facts, composes a soft scene, and samples hard facts for execution. Stage 3 executes a typed program on the same scene. Inference uses the original VLM backbone.

through discrete symbolic execution pairs sampled texts with hard scenes. A symbolic-answer reward and an LLM text-answer reward jointly train the decoder and scene probabilities; hard execution reuses each sampled fact and checks exact counts and unique selections.

We make three contributions:

- We introduce *neuro-symbolic supervision for latent reasoning* as a training mechanism: symbolic execution constrains the entities and predicates represented by latent states, as well as the resulting symbolic answers, while the original neural decoder retains an open answer space.
- We develop a three-stage training procedure that grounds local facts, resolves entity references across steps, and optimizes paired text and scene samples using decoder-answer and exact-execution rewards.
- We report results across six benchmarks and all three training stages. With four latent steps, SYLVA reaches a 72.53% mean, 2.33 points above ReGuLaR's reported mean; joint RL improves every evaluated setting over compositional grounding.

2 RELATED WORK

Neuro-symbolic reasoning. Neuro-symbolic reasoning couples learned perception with structured representations and execution. Modular networks and induced programs expose visual reasoning steps (Andreas et al., 2016; Johnson et al., 2017b; Hudson & Manning, 2018; 2019a; Mascharka et al., 2018; Chen et al., 2021a); relational logic, concept learning, and grounded grammars provide stronger symbolic abstractions (Dong et al., 2019; Han et al., 2019; Li et al., 2020; Mao et al., 2021; Chen et al., 2021b). NS-VQA executes over recovered scenes (Yi et al., 2018), NS-CL learns concepts through an executor (Mao et al., 2019), and NeSyCoCo composes distributed predicates (Kamali et al., 2025). Foundation-model systems also generate visual programs or augment concept grounding with logic (Gupta & Kembhavi, 2023; Surís et al., 2023; Hsu et al., 2023a). Unlike these inference-path interfaces, SYLVA uses symbolic execution only as training supervision, preserving the model's ability to address more open-ended problems. Compositional generalization provides an important background for this setting, as it evaluates whether familiar primitives can be systematically recombined (Hupkes et al., 2020; Sinha et al., 2024). Grounded benchmarks isolate novel language–vision combinations (Ruis et al., 2020; Wu et al., 2021; Sikarwar et al., 2022), while struc-

tured, syntax-guided, and coverage-aware methods address these shifts (Kuo et al., 2021; Kamali & Kordjamshidi, 2023; Hsu et al., 2023b; Chen et al., 2020; Sehgal et al., 2024). Yet visual primitives do not necessarily compose reliably, and neural and symbolic systems exhibit different failure modes (Yun et al., 2023; Zhu et al., 2022). SYLVA addresses this limitation through neuro-symbolic grounding: it binds recurring entities across latent steps to shared symbols and constrains their cross-step composition through executable predicates, thereby strengthening compositional reasoning.

Latent multimodal reasoning. Chain-of-thought externalizes reasoning in text (Wei et al., 2022), whereas latent methods use continuous states. LaCoT learns visual rationales (Sun et al., 2025); LVR predicts visual latents (Li et al., 2025a); MONET reasons in visual space (Wang et al., 2025a); and LASER uses latent superposition (Wang et al., 2026a). DMLR, MCOUT, and SLVR respectively optimize, interleave, or semantically supervise latent states (Liu et al., 2026; Pham & Ngo, 2025; Xu et al., 2026). In contrast, SYLVA neuro-symbolically grounds continuous latent states into shared entities and executable predicates. By supervising both local grounding and cross-step symbolic composition, it shapes the latent trajectory toward more structured and reliable reasoning.

Reward-guided visual learning. DeepEyes learns active inspection (Zheng et al., 2026), Vision-R1 applies outcome-based RL (Huang et al., 2025), and perception-aware policy optimization regularizes visual reasoning (Wang et al., 2026b). SYLVA also samples program-relevant scene facts. Its score-function objective (Williams, 1992) effectively combines a symbolic answer reward with an LLM text-answer reward to jointly update the sampled text and Bernoulli fact probabilities.

3 METHOD

3.1 OVERVIEW

Let $x = (I, q)$ contain one or more images and a question, and let a be the answer. A vision–language backbone generates a trajectory $Z = (z_1, \ldots, z_T)$ followed by answer tokens, where T is the number of valid autoregressive latent reasoning steps and $z_t \in \mathbb{R}^d$ is the state at step t:

$$p_\theta(a \mid x) = \prod_{j=1}^{|a|} p_\theta(a_j \mid I, q, Z, a_{<j}). \tag{1}$$

During training, each question has a fixed first-order logic (FOL) program φ_q. In Stage 1, G_ψ combines each latent state with visual tokens to ground a local (subject, object, relation) triplet. Stage 2 clusters repeated endpoint references into shared entity slots and constructs a scene valuation $\mathcal{S}$ containing occupancy, unary, and binary probabilities. $\mathcal{L}_{\text{sem}}$ aligns slots with entity concepts, while $\mathcal{L}_{\text{sym}}$ supervises the answer obtained by executing φ_q over $\mathcal{S}$. Here θ and ψ parameterize the backbone and grounding heads. Stage 3 uses paired text and discrete-scene rewards to reinforce the same latent trajectory. Figure 2 traces this path. These stage-specific objectives update the backbone representations used by the decoder; inference uses only Equation 1.

3.2 STAGE 1: LOCAL SYMBOLIC GROUNDING

Stage 1 represents each latent-step fact as a (subject, object, relation) triplet. Distinct endpoints define a binary fact; identical endpoints define a unary fact whose relation specifies a class or attribute. Let $V = [v_1; \ldots; v_P] \in \mathbb{R}^{P \times d}$ be the visual-token backbone states. NESYFORMER projects latent and visual states to d_g dimensions and cross-attends to form u_t:

$$r_t = z_t W_z, \qquad x_i = v_i W_x, \qquad X = [x_1; \ldots; x_P], \qquad u_t = \mathrm{CrossAttn}(r_t, X). \tag{2}$$

From u_t, separate subject/object projections and shared visual keys give

$$q_t^s = u_t W_q^s, \quad q_t^o = u_t W_q^o, \quad k_i = x_i W_K, \quad K = [k_1; \ldots; k_P]. \tag{3}$$

For role $r \in \{s, o\}$, attention α_t^r localizes the endpoint and pools its visual evidence c_t^r:

$$\alpha_t^r = \mathrm{softmax}(q_t^r K^\top / \sqrt{d_g}), \qquad c_t^r = \sum_{i=1}^{P} \alpha_{t,i}^r x_i, \quad r \in \{s, o\}. \tag{4}$$

We concatenate the latent, subject, and object features in that order to classify the local relation:

$$m_t = [u_t; c_t^s; c_t^o], \qquad \ell_t^{\text{rel}} = \text{RelationHead}(m_t), \qquad p_t^{\text{rel}} = \mathrm{softmax}(\ell_t^{\text{rel}}). \tag{5}$$

Here $m_t \in \mathbb{R}^{3d_g}$; RelationHead denotes the normalized MLP readout.

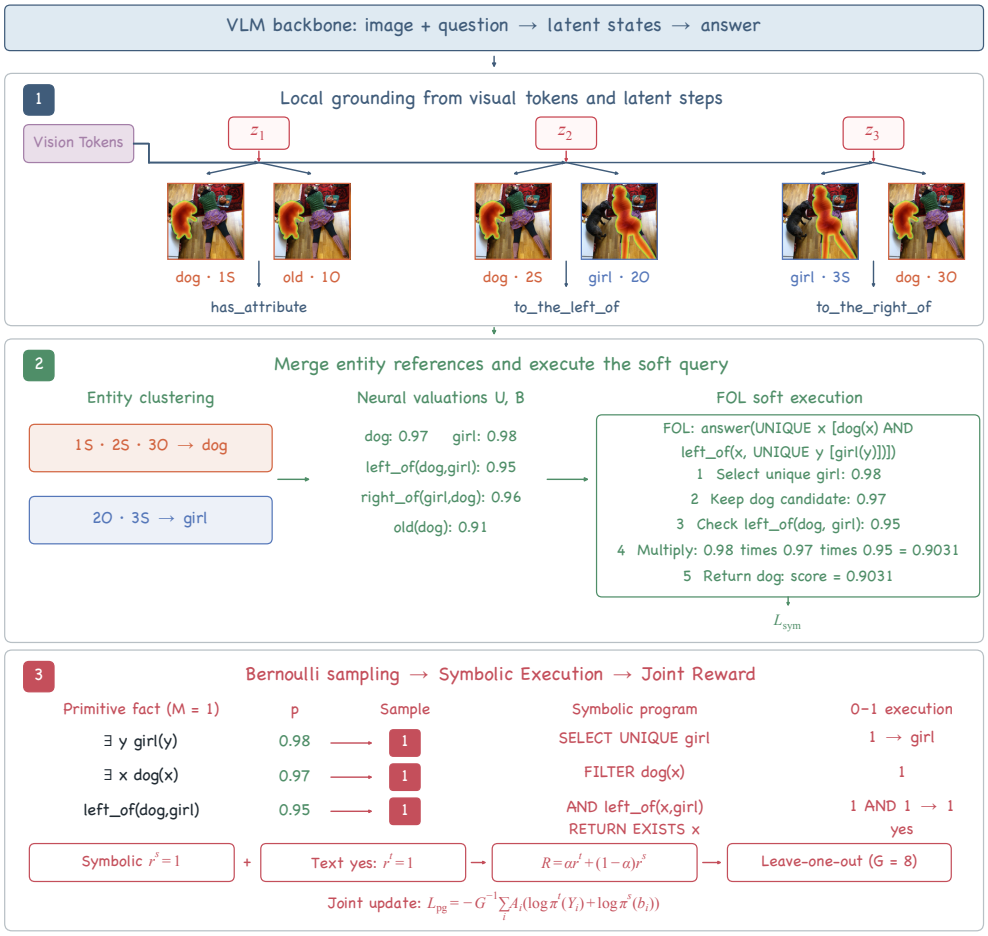

Figure 2: Three-stage training: local grounding, shared-entity composition, and discrete execution reinforcement. S/O denote subject/object references. Stage 3 illustrates one executable query on the same scene: symbolic and text answers jointly supply reward.

Relation labels supervise p_t^{rel} and boxes supervise α_t^r. With answer and latent-boundary supervision, the local objective is

$$\mathcal{L}_{\text{local}} = \lambda_a \mathcal{L}_{\text{ans}} + \lambda_r \mathcal{L}_{\text{rel}} + \lambda_v \mathcal{L}_{\text{att}} + \lambda_b \mathcal{L}_{\text{bound}}. \tag{6}$$

Here $\mathcal{L}_{\text{ans}}$ and $\mathcal{L}_{\text{rel}}$ are answer-token and per-step relation-label cross-entropies. The attention loss is

$$\mathcal{L}_{\text{att}} = \frac{1}{T} \sum_{t,r} D_{\text{KL}}(\beta_t^r \,\|\, \alpha_t^r)\,. \tag{7}$$

It matches valid endpoints to normalized box-overlap targets β_t^r; $\mathcal{L}_{\text{bound}}$ is masked next-token cross-entropy over annotated boundaries and separators, excluding latent placeholders.

3.3 Stage 2: Compositional Symbolic Grounding

Stage 2 turns the step-wise neuro-symbolic predictions from Stage 1 into an executable scene representation. Each latent step proposes a (subject, object, relation) triplet, but references from different steps are initially independent. We cluster and compose triplets that refer to the same entities, organize their unary and binary scores in the format required by the question's FOL program, and execute that program to obtain the final answer. This stage has four modules: shared-entity grouping, typed scene valuation, differentiable FOL execution, and semantic binding supervision.

Grouping triplets by shared entities. The same object may appear as the subject or object of several Stage-1 triplets. We therefore group endpoint attention maps that refer to the same visual

region. For retained questions, $T \leq 16$ valid steps produce at most $2T$ endpoint candidates; unary concept values add no endpoint. Complete-link clustering repeatedly merges the closest groups according to total-variation and spatial-centroid distances (Appendix A.4) until reaching the annotated entity count k, yielding $N = k \leq 16$ shared slots.

For cluster C_i, let α_e denote the attention map of endpoint $e = (t, r)$ from Equation 4. Averaging its member maps produces one shared region, from which we obtain a pooled visual feature and an occupancy score:

$$r_i = \frac{1}{|C_i|} \sum_{e \in C_i} \alpha_e, \qquad c_i = r_i X, \qquad o_i = \mathrm{sigmoid}(w_o^\top c_i + b_o). \tag{8}$$

The cluster assignment is discrete: detached attention maps determine membership, which remains fixed during backpropagation. Feature pooling still uses the original maps, so losses on c_i update both attention and latent states without differentiating through clustering. The annotated count fixes the number of clusters but does not guarantee distinct objects, since identical maps can still produce duplicate regions.

Building a typed scene valuation. After grouping, we describe each shared slot with occupancy, unary concepts, and binary relations. A unary predicate $p(i)$ assigns a class or attribute to slot i, whereas a binary predicate $p(i, j)$ relates an ordered pair of slots. Unary names use `family:value`, such as `class:orange` and `color:orange`, to separate homonyms while retaining properties from different families.

To score whether slot i expresses unary concept p, let e_p be the detached mean of its backbone token embeddings. Learned projections W_v and W_t place the region feature and concept embedding in a shared space:

$$s_{ip} = \cos(W_v c_i,\ W_t e_p) + b_u, \qquad U_{ip} = \mathrm{sigmoid}(s_{ip}). \tag{9}$$

Here s_{ip} is the association logit and U_{ip} the probability that slot i has concept $p \in \mathcal{P}_1$. Only the embedding lookup is detached; the projections and visual/latent branches remain trainable. Because independent scores may recognize *dog* but miss its parent *animal*, a hierarchy built from training data passes the maximum score of each class and its descendants to the executor. Thus *animal* receives at least the *dog* score without repeatedly accumulating correlated evidence.

For a binary relation $p \in \mathcal{P}_2$, we average the valid latent states as $\bar{u} = T^{-1} \sum_{t=1}^{T} u_t$. The Stage-1 RelationHead processes $[\bar{u}; c_i; c_j]$ in the order of Equation 5 and produces B_{ijp}, the score of relation p from slot i to slot j. Together, occupancy o, unary scores U, and the $N \times N \times R$ binary bank B form the typed scene valuation $\mathcal{S} = (o, U, B)$ required by the program.

Executing the FOL program differentiably. Each supported question is converted before training into a fixed FOL program φ_q and checked for predicate types, variable binding, and answer consistency (Appendix B); it is not generated by the model. A compiler maps the program to typed operations over $\mathcal{S} = (o, U, B)$. Atoms read scene scores, connectives combine them, quantifiers aggregate over occupied slots, and selectors identify entities used by later relations. Executing these operations produces a Boolean, count, or attribute answer distribution.

For example, `same_color` compares the color scores of two entities instead of predicting a separate primitive relation. Table 8 lists all supported operations. Their composition is supervised by the annotated answer through

$$\mathcal{L}_{\mathrm{sym}} = -\log \widetilde{P}_{\psi,\theta}(a \mid \varphi_q, \mathcal{S}). \tag{10}$$

Here $\widetilde{P}_{\psi,\theta}$ is the answer score obtained by executing the program over neural valuations; the executor itself has no separately learned parameters. The tilde marks a differentiable surrogate rather than an exact scene probability: product conjunction gives $p \wedge p = p^2$, whereas reusing a Boolean fact gives $b \wedge b = b$. Given fixed cluster assignments, gradients pass through the operations to the valuation heads, pooled regions, attention maps, and latent states (Appendix G). Stage 3 later evaluates the same reasoning under discrete semantics.

Supervising semantic binding. A correct executed answer does not guarantee that the intermediate entities were grounded correctly. We therefore align clustered slots with annotated entities using per-image Hungarian matching between boxes and attention maps, then transfer each matched

entity's class and attribute labels. Let $\mathcal{A}_i$ contain the concepts assigned to slot i and $\mathcal{I}_+$ the labeled slots. We maximize the corresponding unary scores:

$$\mathcal{L}_{\text{sem}} = -\frac{1}{|\mathcal{I}_+|} \sum_{i \in \mathcal{I}_+} \frac{1}{|\mathcal{A}_i|} \sum_{p \in \mathcal{A}_i} \log U_{ip}. \tag{11}$$

This loss trains the projections in Equation 9, while missing labels remain unknown. Detached matching establishes slot–entity correspondence; $\mathcal{L}_{\text{att}}$ separately supervises localization. Appendix C.5 gives the assignment details.

We also retain answer-token, boundary-token, typed-relation, and endpoint-attention supervision (Appendix A.2). Combining them with FOL execution and semantic binding gives the complete Stage-2 objective (all coefficients are reported in Appendix A):

$$\begin{aligned}\mathcal{L}_2 = \lambda_{\text{sym}}\mathcal{L}_{\text{sym}} + \lambda_{\text{sem}}\mathcal{L}_{\text{sem}} + \lambda_{\text{bound}}\mathcal{L}_{\text{bound}} \\ + \lambda_{\text{ans}}\mathcal{L}_{\text{ans}} + \lambda_{\text{rel}}\mathcal{L}_{\text{rel}}^{\text{typed}} + \lambda_{\text{att}}\mathcal{L}_{\text{att}}.\end{aligned} \tag{12}$$

3.4 Stage 3: Reinforcement through discrete symbolic execution

Stage 3 reinforces each latent trajectory through two branches: a *text branch* rewards VLM answers, while a *symbolic branch* rewards discrete execution of the Stage-2 neuro-symbolic scene. Both read the same deterministic latent prefix $Z_\theta(x)$, and their weighted reward trains latent states to support open-ended generation and executable structure.

Sampling the symbolic branch. For every program-relevant occupancy, unary, or binary fact, Stage 2 provides a probability p_ℓ. We clip it to $\hat{p}_\ell = \text{clip}(p_\ell, 0.02, 0.98)$ and draw whether the fact is present ($b_{i\ell} = 1$) or absent ($b_{i\ell} = 0$). Together, these Bernoulli draws convert the soft valuation into a discrete scene:

$$b_{i\ell} \sim \text{Bernoulli}(\hat{p}_\ell), \qquad \pi_\omega^s(b \mid x) = \prod_\ell \hat{p}_\ell^{b_\ell}(1-\hat{p}_\ell)^{1-b_\ell}. \tag{13}$$

where $\omega = (\theta, \psi)$, i indexes rollouts, and ℓ indexes unique facts. Appendix E specifies fact sharing and execution validity.

Sampling the text branch. Following the grouped rollout procedure of GRPO, π_θ^t samples multiple responses Y_i for the same input. From the Stage-2 checkpoint, we initialize the trainable policies and a frozen reference, then draw $G = 8$ text–scene pairs with one independent scene per response ($M = 1$). The branches share $Z_\theta(x)$ but not sampled outputs.

Execution and joint reward. Each pair receives a weighted two-branch reward. The text branch checks the parsed answer; the symbolic branch checks whether hard execution of φ_q on b_i returns the target:

$$\begin{aligned} r_i^t &= \mathbf{1}[\text{Extract}(Y_i) = a], \qquad r_i^s = \mathbf{1}[\text{HardExec}(\varphi_q, b_i) = a], \\ R_i &= \alpha r_i^t + (1-\alpha) r_i^s. \end{aligned} \tag{14}$$

Thus R_i balances answer quality and symbolic executability. Appendix A specifies α, while Appendix E gives the complete typed execution rules; invalid parsing or execution receives zero.

Joint policy optimization. The paired policy factorizes as $\Pi_\omega = \pi_\theta^t \pi_\omega^s$. Its text and scene log-probabilities are

$$\begin{aligned} \ell_i^t &= \sum_u \log p_\theta(y_{iu} \mid x, Z_\theta(x), y_{i,<u}), \\ \ell_i^s &= \sum_\ell [b_{i\ell} \log \hat{p}_\ell + (1 - b_{i\ell}) \log(1 - \hat{p}_\ell)]. \end{aligned} \tag{15}$$

A leave-one-out baseline compares each reward with the other $G-1$ pairs:

$$A_i = R_i - \frac{\sum_{j \neq i} R_j}{G-1}, \qquad \mathcal{L}_{\text{pg}} = -\frac{1}{G}\sum_{i=1}^{G} \text{sg}(A_i)(\ell_i^t + \ell_i^s). \tag{16}$$

Both branches therefore reinforce the shared latent computation with one group-relative signal. Appendix G gives the estimator and stop-gradient contract.

Reference penalties stabilize both policies, while scene entropy supports exploration:

$$\mathcal{L}_3 = \mathcal{L}_{\text{pg}} + \beta_t \mathcal{L}_{\text{KL},t} + \beta_s \mathcal{L}_{\text{KL},s} - \beta_H H_s. \tag{17}$$

Models	V*	HRBench			MMVP	BLINK	SEEDBENCH 2PLUS	Hallusion Bench	Avg.
		Overall	4K	8K					
General-purpose LVLMs									
GPT-4o (OpenAI, 2024)	64.92	57.25	59.00	55.50	68.70	60.02	–	–	–
LLaVA-OneVision (Li et al., 2024a)	72.77	59.56	63.88	55.25	73.33	50.13	61.22	51.28	61.38
Qwen2.5-VL (Bai et al., 2025)	76.44	64.88	68.00	61.75	70.33	57.02	65.31	57.04	65.17
RL-based reasoning LVLMs									
DeepEyes (Zheng et al., 2026)	78.01	65.63	69.25	62.00	71.67	53.02	69.08	61.91	66.55
PAPO (Wang et al., 2026b)	36.13	–	–	–	54.33	52.66	54.11	56.07	–
Vision-R1 (Huang et al., 2025)	78.53	63.44	66.63	60.25	72.00	53.23	68.95	63.06	66.54
Latent reasoning LVLMs									
LVR (Li et al., 2025a)	79.06	50.63	51.13	50.13	70.33	54.97	47.39	65.99	61.40
MONET (Wang et al., 2025a)	79.58	64.13	67.37	60.88	72.33	56.02	65.88	54.91	65.47
LASER (Wang et al., 2026a)	80.10	65.82	70.25	61.38	73.00	58.55	70.05	<u>67.05</u>	69.09
SLVR (Xu et al., 2026)	82.20	<u>66.45</u>	70.40	<u>62.50</u>	73.00	61.23	<u>70.44</u>	66.34	69.94
ReGuLaR (Wang et al., 2026c)	<u>83.25</u>	66.19	<u>70.50</u>	61.88	<u>73.67</u>	**61.81**	70.22	66.08	<u>70.20</u>
SYLVA	**85.86**	**72.01**	**74.88**	**69.13**	**75.33**	<u>61.49</u>	**71.41**	**69.09**	**72.53**

Table 1: Reported scores (%). SLVR uses original-question (Q1) scores. SYLVA uses four latent steps as described in Section 4. Avg. weights six benchmarks equally after averaging HRBench-4K/8K. Bold/underline mark best/second-best available scores; dashes mark unavailable or non-comparable results across the reported evaluation settings.

4 EXPERIMENTS

4.1 EVALUATION SETTING

Training and comparison. We use Qwen2.5-VL-7B (Bai et al., 2025) with a frozen vision tower. Stage 1 trains on RGROUNDING-351K (Wang et al., 2026c) for 1,500 updates. Stage 2 samples 64,000 exposures with replacement from 141,021 eligible examples over 1,000 updates. Stage 3 runs one epoch over 7,838 samples from distinct images in a stricter executable subset. All stages use seed 42. All experiments are conducted on $4\times$ NVIDIA A100 GPUs with 80 GB memory. Appendix B describes preparation, and Appendix A.3 gives the training schedules.

Table 1 groups the reported benchmark results of general-purpose LVLMs, including GPT-4o (OpenAI, 2024), LLaVA-OneVision (Li et al., 2024a), and Qwen2.5-VL (Bai et al., 2025); RL-based reasoning LVLMs, including DeepEyes (Zheng et al., 2026), PAPO (Wang et al., 2026b), and Vision-R1 (Huang et al., 2025); and latent-reasoning LVLMs, including LVR (Li et al., 2025a), MONET (Wang et al., 2025a), LASER (Wang et al., 2026a), SLVR (Xu et al., 2026), ReGuLaR (Wang et al., 2026c), and SYLVA.

Benchmarks and reported scores. We evaluate V* (Wu & Xie, 2024), MMVP (Tong et al., 2024), BLINK (Fu et al., 2024), SEED-Bench-2-Plus (Li et al., 2024b), HallusionBench (Guan et al., 2024), and HRBench (Wang et al., 2025b). These cover detailed evidence, visual discrimination, diverse perception, text-rich understanding, hallucination sensitivity, and high-resolution perception. Table 1 retains HRBench Overall/4K/8K for comparison. Evaluation uses only the Qwen backbone component.

Each checkpoint produces one answer after four *latent reasoning steps*, not four samples. For all six benchmarks, we follow the evaluation protocols used by LVR and ReGuLaR, as detailed in Appendix H.1. We average HRBench-4K/8K once and then compute the mean across all six evaluation benchmarks as the aggregate result.

4.2 MAIN RESULTS

SYLVA has the highest available six-benchmark mean in Table 1: 72.53%, versus 70.20% for ReGuLaR and 69.09% for LASER, and leads on five benchmarks. SYLVA therefore outperforms existing latent-reasoning methods, demonstrating the potential of introducing neuro-symbolic reasoning into latent reasoning.

SEED-Bench-2-Plus and HallusionBench include questions beyond our program language, which lacks general OCR. Older neuro-symbolic systems such as NS-VQA (Yi et al., 2018) and NeSy-CoCo (Kamali et al., 2025) answer through their executors and therefore cannot attempt questions outside their predefined languages. In contrast, SYLVA can answer such questions through its neural decoder and performs better after symbolic training. This suggests that symbolic supervision im-

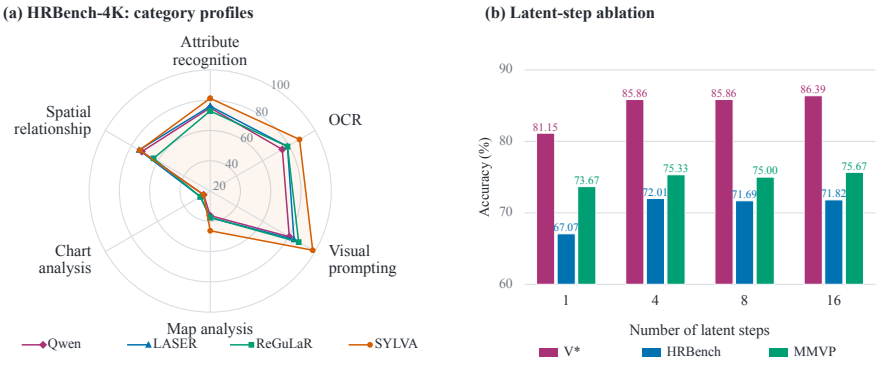

Figure 3: High-resolution analysis and latent-step ablation. (a) Comparison of SYLVA, ReGuLaR, LASER, and the Qwen2.5-VL baseline across six HRBench-4K question categories. (b) Accuracy with 1, 4, 8, or 16 latent steps; HRBench averages 4K/8K.

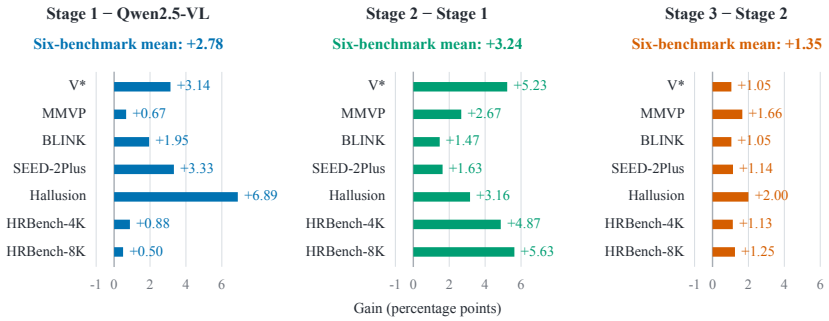

Figure 4: Training-stage gains (percentage points): Stage 1 over published Qwen2.5-VL, corrected Stage 2 over Stage 1, and RL over Stage 2. All checkpoints use four latent steps at evaluation.

proves general capabilities, such as attending to and representing visual evidence, rather than only the specific skills covered by the programs.

4.3 High-resolution perception and latent-step ablation

On HRBench-4K/8K, SYLVA, ReGuLaR, and LASER record 74.88/69.13%, 70.50/61.88%, and 70.25/61.38%. Figure 3a compares the four models across attribute recognition, OCR, visual prompting, map analysis, chart analysis, and spatial relationships. SYLVA achieves the best results on attribute recognition, OCR, visual prompting, and map analysis, remains competitive on spatial relationships, and obtains the highest overall HRBench-4K accuracy, demonstrating stronger high-resolution perception across diverse question types. Figure 3b varies inference length: one step yields 81.15/67.07/73.67% on V*/HRBench/MMVP, four yield 85.86/72.01/75.33%, and eight or sixteen remain within 0.53 points. We use four steps in the remaining evaluations.

4.4 Ablation Study and Visualization

Figure 4 traces the curriculum: local grounding reaches 67.95% (+2.78 over published Qwen2.5-VL); compositional grounding reaches 71.18% (+3.24); joint RL adds 1.35 points, totaling +4.58 over Stage 1. Stage 2 supplies the largest mean increment, especially on HRBench-8K, V*, and HRBench-4K; joint RL improves all seven reported columns. The progression reflects complementary roles: Stage 1 establishes local visual facts, Stage 2 links repeated entities and composes those facts, and Stage 3 reinforces whether the resulting scene supports correct execution. The larger Stage 2 gains on HRBench indicate that entity sharing and relation composition matter most when evidence must be integrated across high-resolution regions, while the consistent Stage 3 gains show that execution feedback improves the composed representation across benchmarks and question families under the same evaluation protocol. Since the stages also differ in data and objectives, the matched controls in Table 2 separately test each supervised objective and the complete reinforcement pathway. This control structure distinguishes cumulative curriculum gains from the contribution of each individual mechanism under matched training conditions.

Table 2: Accuracy (%) on V*, HRBench-4K/8K, and MMVP. "w/o Stage 2" retains Stages 1 and 3; "w/o Stage 1" retains Stages 2 and 3. Triplet-CoT first emits a chain of thought comprising visual triplets, each containing two objects and their relation, and then outputs the answer.

Training condition	V*	HR-4K	HR-8K	MMVP
Stage 1: local grounding				
Ours	79.58	68.88	62.25	71.00
Vanilla SFT (answer only)	73.82	67.88	61.38	71.00
Triplet-CoT SFT	74.87	67.00	60.25	70.33
Qwen2.5-VL-7B baseline	76.44	68.00	61.75	70.33
Stage 2: compositional grounding				
Ours	84.81	73.75	67.88	73.67
No $\mathcal{L}_{\text{sym}}$	81.15	70.13	63.25	71.67
No $\mathcal{L}_{\text{sem}}$	83.25	71.75	65.63	73.33
Stage 3: execution reinforcement				
Ours	85.86	74.88	69.13	75.33
w/o Stage 2	83.25	70.50	64.13	71.33
w/o Stage 1	79.58	68.50	62.75	70.67
Text-only GRPO	81.68	71.63	64.50	72.00

Table 2 reports V*, HRBench-4K, HRBench-8K, and MMVP separately to expose differences across tasks and resolutions. The Stage-1 comparison tests whether local latent grounding improves on answer-only SFT and textual serialization of the same relation annotations (Triplet-CoT SFT). Grounding outperforms both baselines on V* and HRBench; on MMVP it ties Vanilla SFT and exceeds Triplet-CoT. The controls share data, 1,500 updates, and seed. The published backbone row is an external reference.

For Stage 2, removing either objective lowers all four scores, with larger drops for $\mathcal{L}_{\text{sym}}$, especially on HRBench-8K, demonstrating the benefit of constructing the complete symbolic reasoning pipeline. For Stage 3, the full model exceeds text-only GRPO on every column, again most on HRBench-8K. Text-only GRPO uses only decoder-answer reward.

Bypassing either supervised stage before execution reinforcement also lowers every score. These controls support a progressive Stage 1→Stage 2→Stage 3 curriculum rather than three interchangeable objectives. Stage 1 first teaches broad extraction of local object relations. Stage 2 builds on these local facts by binding repeated entities and integrating multiple object relations into executable compositions. Stage 3 then reinforces the logical integration of the resulting scene through discrete execution. The larger degradation without Stage 1 further indicates that broad local grounding provides an important foundation for the later compositional and reinforcement stages.

Figure 5 reveals that latent-step hidden states encode visual region–concept bindings and relations across regions. These neuro-symbolic representations complement the Qwen backbone for more precise reasoning.

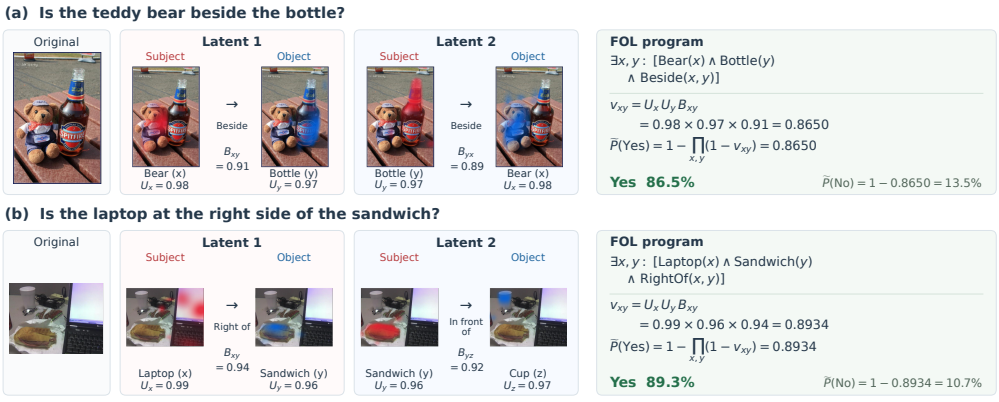

Figure 5: Visualization of the neuro-symbolic information encoded by latent steps for two inference examples. For this analysis, we additionally apply the symbolic branch at evaluation to recover the object attention associated with each latent step and confidence scores for inter-object relations. The FOL program then simulates how these intermediate predictions produce a symbolic answer.

5 Conclusion

We introduce SYLVA, which integrates neuro-symbolic supervision with VLM latent reasoning so learned logical structure can generalize to a broader range of open problems while symbolic knowledge improves latent representations. Its three-stage pipeline grounds local facts, composes shared entities, and applies joint reinforcement. SYLVA achieves state-of-the-art performance across six benchmarks, demonstrating that neuro-symbolic and latent reasoning together offer a promising path toward more precise and general multimodal reasoning.

AI USE STATEMENT

GPT-6 was used offline to generate one candidate FOL program for each Stage-2 sample from its question, canonical answer, and normalized scene annotations. Each program was accepted only after deterministic syntax, type, variable-binding, reference, and answer-consistency checks. The validated programs serve as fixed Stage-2 annotations; GPT-6 is not invoked online during model training or inference. Generative AI was also used to polish the English academic writing and refine the final figures. All AI-assisted outputs were reviewed or validated by the authors.

ETHICS STATEMENT

This work studies model training and evaluation on established vision–language datasets and does not introduce new data collection or direct interaction with human participants.

REPRODUCIBILITY STATEMENT

The main text specifies the training objectives and neural inference procedure. Appendices A–E provide configurations, preprocessing rules, execution semantics, and a worked example. All benchmark evaluations follow the protocols used by LVR and ReGuLaR.

REFERENCES

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to compose neural networks for question answering. In *Proceedings of NAACL-HLT*, 2016.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Wenhu Chen, Zhe Gan, Linjie Li, Yu Cheng, William Wang, and Jingjing Liu. Meta module network for compositional visual reasoning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021a.

Xinyun Chen, Chen Liang, Adams Wei Yu, Dawn Song, and Denny Zhou. Compositional generalization via neural-symbolic stack machines. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1690–1701, 2020.

Zhenfang Chen, Jiayuan Mao, Jiajun Wu, Kwan-Yee Kenneth Wong, Joshua B. Tenenbaum, and Chuang Gan. Grounding physical concepts of objects and events through dynamic visual reasoning. In *International Conference on Learning Representations*, 2021b.

Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. Neural logic machines. In *International Conference on Learning Representations*, 2019.

Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. In *European Conference on Computer Vision*, pp. 148–166. Springer, 2024.

Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, et al. Hallusionbench: an advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14375–14385, 2024.

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

Chi Han, Jiayuan Mao, Chuang Gan, Joshua B. Tenenbaum, and Jiajun Wu. Visual concept-metaconcept learning. In *Advances in Neural Information Processing Systems*, 2019.

Joy Hsu, Jiayuan Mao, Joshua B. Tenenbaum, and Jiajun Wu. What's left? concept grounding with logic-enhanced foundation models. *Advances in Neural Information Processing Systems*, 36:38798–38814, 2023a.

Joy Hsu, Jiayuan Mao, and Jiajun Wu. DisCo: Improving compositional generalization in visual reasoning through distribution coverage. *Transactions on Machine Learning Research*, 2023b.

Wenxuan Huang, Bohan Jia, Zijie Zhai, Shaosheng Cao, Zheyu Ye, Fei Zhao, Zhe Xu, Xu Tang, Yao Hu, and Shaohui Lin. Vision-r1: Incentivizing reasoning capability in multimodal large language models. *arXiv preprint arXiv:2503.06749*, 2025.

Drew Hudson and Christopher D. Manning. Learning by abstraction: The neural state machine. In *Advances in Neural Information Processing Systems*, 2019a.

Drew A. Hudson and Christopher D. Manning. Compositional attention networks for machine reasoning. In *International Conference on Learning Representations*, 2018.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019b.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pp. 5065–5069, 2020. doi: 10.24963/ijcai.2020/708.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2901–2910, 2017a.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017b.

Danial Kamali and Parisa Kordjamshidi. Syntax-guided transformers: Elevating compositional generalization and grounding in multimodal environments. In *Proceedings of the 1st GenBench Workshop on (Benchmarking) Generalisation in NLP*, pp. 130–142, 2023. doi: 10.18653/v1/2023.genbench-1.10.

Danial Kamali, Elham J. Barezi, and Parisa Kordjamshidi. NeSyCoCo: A neuro-symbolic concept composer for compositional generalization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(4):4184–4193, 2025. doi: 10.1609/aaai.v39i4.32439.

Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Shahab Kamali, Matteo Malloci, Jordi Pont-Tuset, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://storage.googleapis.com/openimages/web/index.html*, 2017.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.

Yen-Ling Kuo, Boris Katz, and Andrei Barbu. Compositional networks enable systematic generalization for grounded language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 216–226, 2021. doi: 10.18653/v1/2021.findings-emnlp.21.

Bangzheng Li, Ximeng Sun, Jiang Liu, Ze Wang, Jialian Wu, Xiaodong Yu, Hao Chen, Emad Barsoum, Muhao Chen, and Zicheng Liu. Latent visual reasoning. *arXiv preprint arXiv:2509.24251*, 2025a.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024a.

Bohao Li, Yuying Ge, Yi Chen, Yixiao Ge, Ruimao Zhang, and Ying Shan. Seed-bench-2-plus: Benchmarking multimodal large language models with text-rich visual comprehension. *arXiv preprint arXiv:2404.16790*, 2024b.

Qing Li, Siyuan Huang, Yining Hong, Yixin Chen, Ying Nian Wu, and Song-Chun Zhu. Closed loop neural-symbolic learning via integrating neural perception, grammar parsing, and symbolic reasoning. In *Proceedings of the International Conference on Machine Learning*, 2020.

You Li, Heyu Huang, Chi Chen, Kaiyu Huang, Chao Huang, Zonghao Guo, Zhiyuan Liu, Jinan Xu, Yuhua Li, Ruixuan Li, et al. Migician: Revealing the magic of free-form multi-image grounding in multimodal large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 9845–9867, 2025b.

Chengzhi Liu, Yuzhe Yang, Yue Fan, Qingyue Wei, Sheng Liu, and Xin Eric Wang. Reasoning within the mind: Dynamic multimodal interleaving in latent space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 9225–9236, 2026. URL https://github.com/eric-ai-lab/DMLR.

Fangyu Liu, Guy Emerson, and Nigel Collier. Visual spatial reasoning. *Transactions of the Association for Computational Linguistics*, 11:635–651, 2023a.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023b.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1904.12584.

Jiayuan Mao, Haoyue Shi, Jiajun Wu, Roger P. Levy, and Joshua B. Tenenbaum. Grammar-based grounded lexicon learning. In *Advances in Neural Information Processing Systems*, 2021.

David Mascharka, Philip Tran, Ryan Soklaski, and Arjun Majumdar. Transparency by design: Closing the gap between performance and interpretability in visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

OpenAI. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Tan-Hanh Pham and Chris Ngo. Multimodal chain of continuous thought for latent-space reasoning in vision-language models. *arXiv preprint arXiv:2508.12587*, 2025. doi: 10.48550/arXiv.2508.12587.

Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M. Lake. A benchmark for systematic generalization in grounded language understanding. In *Advances in Neural Information Processing Systems*, 2020.

Atharva Sehgal, Arya Grayeli, Jennifer J. Sun, and Swarat Chaudhuri. Neurosymbolic grounding for compositional world models. In *International Conference on Learning Representations*, 2024.

Ankur Sikarwar, Arkil Patel, and Navin Goyal. When can transformers ground and compose: Insights from compositional generalization benchmarks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 648–669, 2022.

Sania Sinha, Tanawan Premsri, and Parisa Kordjamshidi. A survey on compositional learning of AI models: Theoretical and experimental practices. *Transactions on Machine Learning Research*, 2024.

Guohao Sun, Hang Hua, Jian Wang, Jiebo Luo, Sohail Dianat, Majid Rabbani, Raghuveer Rao, and Zhiqiang Tao. Latent chain-of-thought for visual reasoning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3465.

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.

Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes wide shut? exploring the visual shortcomings of multimodal llms. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9568–9578, 2024.

Qixun Wang, Yang Shi, Yifei Wang, Yuanxing Zhang, Pengfei Wan, Kun Gai, Xianghua Ying, and Yisen Wang. Monet: Reasoning in latent visual space beyond images and language. *arXiv preprint arXiv:2511.21395*, 2025a.

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, Wei Yu, and Dacheng Tao. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(8):7907–7915, 2025b.

Yubo Wang, Juntian Zhang, Yichen Wu, Yankai Lin, Nils Lukas, and Yuhan Liu. Forest before trees: Latent superposition for efficient visual reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11272–11288, 2026a. doi: 10.18653/v1/2026.acl-long.517. URL https://aclanthology.org/2026.acl-long.517/.

Zhenhailong Wang, Xuehang Guo, Sofia Stoica, Haiyang Xu, Hongru Wang, Hyeonjeong Ha, Xiusi Chen, Yangyi Chen, Ming Yan, Fei Huang, et al. Perception-aware policy optimization for multimodal reasoning. In *International Conference on Learning Representations*, 2026b.

Zihu Wang, Karthik Somayaji N.S, and Peng Li. ReGuLaR: Relation-grounded latent reasoning for large vision-language models. *arXiv preprint arXiv:2605.30587*, 2026c.

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

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992. doi: 10.1007/BF00992696.

Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13084–13094, 2024.

Zhengxuan Wu, Elisa Kreiss, Desmond C. Ong, and Christopher Potts. ReaSCAN: Compositional reasoning in language grounding. *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2021.

Tianrun Xu, Yue Sun, Qixun Wang, Jingyi Lu, Yuan Wang, Tianren Zhang, Longteng Guo, Fengyun Rao, Jing Lyu, Feng Chen, and Jing Liu. Semantic-enriched latent visual reasoning. In *International Conference on Machine Learning*, 2026. URL https://github.com/tinnel123666888/slvr.

Jingkang Yang, Yi Zhe Ang, Zujin Guo, Kaiyang Zhou, Wayne Zhang, and Ziwei Liu. Panoptic scene graph generation. In *European conference on computer vision*, pp. 178–196. Springer, 2022.

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 *Advances in Neural Information Processing Systems*, volume 31, 2018.

Tian Yun, Usha Bhalla, Ellie Pavlick, and Chen Sun. Do vision-language pretrained models learn composable primitive concepts? *Transactions on Machine Learning Research*, 2023.

Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. Deepeyes: Incentivizing" thinking with images" via reinforcement learning. In *International Conference on Learning Representations*, 2026.

Wang Zhu, Jesse Thomason, and Robin Jia. Generalization differences between end-to-end and neuro-symbolic vision-language reasoning systems. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 4697–4711, 2022. doi: 10.18653/v1/2022.findings-emnlp.345.

A ADDITIONAL IMPLEMENTATION DETAILS

A.1 NOTATION AND INTERFACES

Table 3: Notation used across the three training stages.

Symbol	Meaning
P	Number of visual tokens; varies with image resolution and image count.
T	Number of valid training latent steps; $T \leq 16$ in Stages 2 and 3.
$k_h,\ N$	Active entities per image and total object slots; $N \leq 16$.
$\mathcal{P}_1, \mathcal{P}_2$	Typed unary- and binary-relation vocabularies.
$G,\ M$	Joint samples per input ($G = 8$); scenes per text ($M = 1$).
$\alpha,\ \omega$	Text reward weight (0.5); trainable parameters (θ, ψ).
R	Number of distinct binary relations required by the current sample.

The 7B backbone has hidden dimension 3,584. Stage 1 uses NESYFORMER to project latent and visual states into dimension 1,024, apply two cross-attention layers with eight heads, and pool subject and object features separately. RelationHead is a two-layer MLP that maps the normalized 3,072-dimensional role-aware representation to 1,024 dimensions with GELU, then to a classifier with 2,111 output classes. The semantic projection rank is 128 in the current Stage-2 implementation. Scene tensors have shapes $C \in \mathbb{R}^{N\times 1024}$, region attention $R \in \mathbb{R}^{N\times P}$, occupancy $o \in [0,1]^N$, unary scores $U \in [0,1]^{N\times|\mathcal{P}_1|}$, and binary scores $B \in [0,1]^{N\times N\times R}$, where R is the number of distinct binary relations required by the current sample.

The symbolic branch receives projected backbone states at visual-token positions. These features come from the backbone rather than a separate detector or ground-truth boxes. Gradients update the trainable language-side representations and grounding projections while the vision tower remains frozen.

A.2 TYPED AUXILIARY RELATION SUPERVISION

The Stage-2 auxiliary relation loss separates targets by arity. Unary steps classify between the retained `is_a` and `has_attribute` rows, while binary steps classify only among verified binary-relation rows. The unary and binary sums share a denominator equal to the total number of valid relation steps. This auxiliary classification preserves the Stage-1 relation interface; concrete class and attribute values are instead learned by the typed association scores in Equation 9. The endpoint attention loss supervises the single valid endpoint of a unary step and both subject and object endpoints of a binary step.

A.3 TRAINING SETTINGS AND PROVENANCE

We follow the training configurations documented for each stage, including random seeds, optimization, and latent-step construction. Stages 1 and 2 use 1,500 and 1,000 optimizer updates, respectively; Stage 3 runs for one epoch over every example in the qualified RL pool. All three stages use training seed 42. Stage 1 uses one latent step per selected relation target. Stage 2 constructs at most 16 latent steps from each example's corrected targets, and Stage 3 likewise uses a latent prefix of at most 16 steps. Standard benchmark evaluation uses four latent steps for every checkpoint and produces one answer per question. The trajectory-length ablation in Figure 3b instead uses 1, 4, 8, and 16 steps. Training updates, epochs, and latent reasoning steps are distinct quantities.

The Stage-2 objective uses $\lambda_{\text{sym}} = 1$, $\lambda_{\text{sem}} = 0.2$, $\lambda_{\text{bound}} = 0.1$, $\lambda_{\text{ans}} = 0.5$, $\lambda_{\text{rel}} = 0.2$, and $\lambda_{\text{att}} = 0.1$. The Stage-3 scores report the paired actor in Equation 16.

The reference is frozen; its fact branch evaluates the detached live region layout to align action indices. For the joint recipe, the initial coefficients are $\beta_t = 0.05$ and $\beta_s = 0.05$. The symbolic KL controller multiplies β_s by 1.5 when mean per-action KL exceeds 0.5, divides by 1.5 below 0.25, and clips it to $[0.001, 1]$. The text coefficient stays fixed. Fact entropy decreases linearly from 0.01 to 0.001 over the single training epoch.

Table 4: Joint RL training configuration.

Item	Value or status
Optimizer / learning rate	AdamW / 10^{-6}
Weight decay / warmup	0.1 / 25 optimizer steps
Batch / accumulation	1 example per device / 16 microbatches
Documented device count	4; global batch 64 under that world size
Training schedule	One epoch over the qualified RL pool
Precision / gradient clipping	bfloat16 / norm 1.0
Joint pairs / scenes per text	$G = 8$ / $M = 1$
Reward weights / baseline	$\alpha = 0.5$ (text) and $1 - \alpha = 0.5$ (symbolic) / leave-one-out
Text sampling / fact clipping	Temperature 1, no top-k/p / $[0.02, 0.98]$
Object-slot cap / latent-step cap	16 / 16
Training seed	42
Maximum sequence length	8,192

Policy evaluation disables dropout and retains gradients. We use no trainable critic.

Joint samples share the deterministic latent prefix and fact probabilities, but text decoding and reference evaluation incur additional computation. Latent boundaries are forced with at most 16 hidden-state feedback steps during RL training; sampled text tokens alone enter the text actor. Truncated responses use the same parser, with malformed answers receiving zero.

A.4 Attention-based entity clustering

The main text presents the single-image case. For a multi-image example, an endpoint map is restricted to the visual-token positions of each image and renormalized before clustering; its original attention mass on that image weights its contribution to the pooled region. After removing padded steps, let α, α' denote two normalized maps over one image and let $\xi_p \in [0,1]^2$ be patch-center coordinates. Complete linkage uses the pairwise dissimilarity

$$d(\alpha, \alpha') = \max\left\{ \frac{\frac{1}{2}\|\alpha - \alpha'\|_1}{0.2}, \frac{\|\sum_p (\alpha_p - \alpha'_p)\xi_p\|_2}{0.08} \right\}. \tag{18}$$

We retain for Stage 2 only questions selecting at most 16 annotated entities and set the object-slot capacity to 16 for tensorized modeling. Examples above this limit are rejected without truncation. For an example with $k \leq 16$ entities, clustering produces k active groups; slots with index $i > k$ are padding slots whose attention weights are set identically to zero and are masked from subsequent valuation and execution. Annotated coordinates, semantic labels, and endpoint identities do not determine cluster membership. For multi-image inputs, the active counts satisfy $k = \sum_h k_h$.

B Data preparation

B.1 Source pool and filtering

Stage 1 adopts RGROUNDING-351K, the relation-grounding dataset constructed by ReGuLaR (Wang et al., 2026c), and follows its reported preprocessing protocol. The final corpus contains 351,139 examples from GQA (Hudson & Manning, 2019b), OpenImages (Krasin et al., 2017), CLEVR (Johnson et al., 2017a), Visual Genome (Krishna et al., 2017), PSG (Yang et al., 2022), multi-image grounding data (Li et al., 2025b), and VSR (Liu et al., 2023a). Table 5 gives the fixed source quotas.

We follow ReGuLaR's source-level overlap controls. Because GQA uses Visual Genome images, the GQA image IDs are removed from the Visual Genome candidate pool before its quota is sampled. VSR and PSG are made mutually exclusive through their shared COCO image IDs, and COCO-derived records are excluded from the multi-image grounding subset. Multiple questions about different facts in the same retained image are preserved because they provide distinct question-conditioned supervision. After source-specific filtering, examples are selected and shuffled with a

Table 5: Stage-1 source composition, following the RGROUNDING-351K preparation reported by ReGuLaR.

Source	Examples	Share
GQA	245,000	69.8%
OpenImages	50,000	14.2%
CLEVR	25,000	7.1%
Visual Genome	15,139	4.3%
PSG	6,000	1.7%
Multi-image grounding	5,000	1.4%
VSR	5,000	1.4%
Total	351,139	100.0%

fixed random seed. When a source supplies an official train/test split, only its training partition contributes Stage-1 examples.

Of the 351,139 examples, 270,000 (76.9%) from GQA and CLEVR already provide the question–answer pair, relation, object identity, and bounding box. The remaining 81,139 examples use generated relation-oriented questions, while their relations, entity identities, and boxes remain source annotations. Before inclusion, deterministic checks validate format, entity references, boxes, answer consistency, and duplicates; these checks reject 4,327 generated candidates. The retained 81,139 generated-question examples pass all checks.

The corpus audit reported by ReGuLaR samples 2,000 unique generated-question examples and inspects the image, question, answer, source relation, entity identities, and rendered boxes. Table 6 records that audit. The eight severe errors are ambiguous generated questions without a unique defensible answer; no audited severe error changes a source relation, entity identity, or box. Following the shared protocol, we apply the resulting uniqueness check to the complete generated-question partition.

Table 6: Quality audit reported for the generated-question portion of the shared Stage-1 corpus. Intervals are 95% Wilson confidence intervals.

Audit measure	Result
Usable examples	99.6% (1,992/2,000), CI [99.2, 99.8]
Strictly issue-free examples	98.1%
Severe errors	0.4% (8/2,000), CI [0.2, 0.8]
Lowest source-level usable rate	97.5% (200 audited)
Independent double annotation	98.0% agreement; Cohen's $\kappa = 0.86$ (400 examples)

Stage-1 object grounding follows the same deterministic matching hierarchy. For GQA, we first use official object IDs from the question, answer, full-answer annotation, and semantic-program arguments. If these IDs do not yield a target, we apply normalized lexical overlap followed by a deterministic, bounded whole-image fallback. Adapters for the remaining sources preserve their source subject/object IDs and target indices, avoiding post-hoc text-to-box matching.

For every selected relation target, its box B is projected to the merged dynamic-resolution grid defined by Qwen2.5-VL's `image_grid_thw` and `spatial_merge_size`. If G_i is token cell i, its supervision weight is

$$a_i = \frac{\mathrm{area}(B \cap G_i)}{\sum_j \mathrm{area}(B \cap G_j)}.$$

Multi-image examples use the corresponding image-specific token offsets. A training example with K selected relation targets inserts exactly K latent slots and aligns target t with slot t; padded positions are masked, and preprocessing rejects a record if the target and slot counts disagree.

The 2,111-class relation vocabulary is fixed before training and optimized with sample-level cross-entropy without class weights. The most frequent labels are `to_the_left_of` (15.3%), `to_the_right_of` (10.1%), `has_attribute` (4.1%), `on` (2.0%), and `wearing` (2.0%). Tail

examples including `waving`, `cooking`, `feeding`, `pouring`, and `surfing` each occur in less than 0.01% of the corpus.

Table 7: Stage-transition preparation counts. Stage 2 samples randomly with replacement from its eligible pool.

Processing stage	Count
Stage-1 relation-grounding examples	351,139
Candidates with compiler-valid supported FOL programs	148,394
Final Stage-2 pool ($\leq$ 16 entities and resource-valid)	141,021
Candidates rejected by entity/step resource limits	7,373
Stage-2 sample exposures with replacement	64,000 (1,000 updates $\times$ global batch 64)
Final qualified Stage-3 pool (one sample per unique image)	7,838

Stage-2 FOL program generation. We follow the compositional program formulation of NeSy-CoCo (Kamali et al., 2025) and express each question using typed first-order predicates, logical connectives, quantifiers, entity selection, counting, comparison, and family-specific attribute queries. GPT-6 receives the question, canonical answer, and normalized scene annotations and generates one FOL program for every Stage-2 candidate sample. Program generation is performed offline before training; neither the generated text nor GPT-6 is used during model inference.

The generated program is not accepted directly. A deterministic parser and compiler verify syntax, predicate membership and arity, argument types, variable binding, relation direction, answer type, and execution consistency with the canonical answer. Unsupported or malformed programs, unresolved references, and programs whose execution disagrees with the answer are rejected. Of the Stage-1 records, 148,394 receive a supported compiler-valid program. Applying the entity and resource constraints then yields 141,021 Stage-2 samples, each paired with one frozen FOL program.

Preparation joins the original example, corrected scene annotation, and generated FOL program using a stable case key. Duplicate keys, missing records, and conflicting annotations are errors. Entity identity uses the complete entity key, including image membership, rather than row order or class name. Triples refer to entity keys; box coordinates reside in the entity table as pixel-space $[x_1, y_1, x_2, y_2]$ values. Corrected cases include attribute queries initially represented as existence questions, duplicate entity IDs, reversed relations, and binary relations whose endpoints collapse to the same region. The compiler consumes the validated FOL text directly; legacy serialized syntax trees are not authoritative.

B.2 Splits, registries, and sampling

Neither SFT stage uses an internal training–validation split. Stage 1 trains on the complete prepared RGROUNDING-351K corpus, and Stage 2 samples from its complete 141,021-example eligible pool. The preparation pipeline groups candidates by connected components of image identity to track questions that share an image, including those linked through multi-image examples. Semantic vocabularies, class hierarchies, and predicate-type maps are built from the complete Stage-2 pool. Possible image overlap with the evaluation benchmarks has not yet been fully audited.

Stage 2 randomly samples with replacement from the 141,021-example training pool. The documented mixture is 50% binary-relation, 30% unary/localization/attribute, and 20% aggregate examples, with source balancing. The 1,000-update schedule and global batch of 64 yield 64,000 sample exposures. Each sampled case key must have a corresponding training-format record. Preparation fails if any record is missing.

Stage-2 targets are ordered as binary relations, direct class labels, then attributes. The valid latent count therefore need not match Stage 1. Prepared records persist total/per-image entity budgets, latent-step count, and the source of the slot budget. The resource checks require $N \leq 16$ and $T \leq 16$ and reject excess cases without truncating the targets. Retained classifier row identities

must match the Stage-1 checkpoint. Alias normalization may reuse existing rows, but `is_a` and `is` remain semantically distinct.

B.3 Reliable supervision and reward

Counting requires a sufficiently complete domain, while negation and universal statements require evidence that supports their closed-world interpretation. A missing annotation is not automatically false. For Boolean multiple-choice questions, an explicit option map links answer letters to truth values; A is not assumed to mean yes. Attribute answers are normalized within the appropriate semantic family.

RL uses a stricter subset of the Stage-2 pool. We retain only questions whose annotated FOL programs admit a unique valid logical resolution and return the canonical correct answer, thereby excluding ambiguous supervision and solutions that can reach the answer for unintended reasons. The qualified manifest records the supporting evidence for each decision; questions with multiple valid resolutions or incomplete scene evidence are rejected. After qualification, we rebalance the proportions of canonical answer options and retain one sample per image to reduce option-frequency and repeated-image bias in Stage-3 training. The resulting 16-slot pool contains 7,838 samples from 7,838 unique images. Stage 3 processes every retained sample once, i.e., one complete epoch over this qualified pool.

C Differentiable execution details

C.1 Supported neuro-symbolic operations

Table 8 gives the complete operator inventory used by the differentiable executor. Unary and binary atoms read the valuation banks U and B. For a formula body with slotwise score μ_i, define the existence-weighted score $\widetilde{\mu}_i = o_i\mu_i$ and null mass $q_0 = \prod_i(1-\widetilde{\mu}_i)$. A selector has weights $w_i = \widetilde{\mu}_i/(\sum_j \widetilde{\mu}_j + q_0 + \epsilon)$. These definitions keep absent or unresolved entities from being replaced by an arbitrary uniform selection.

Table 8: Supported differentiable neuro-symbolic operations. All operations return probabilities or probability distributions and remain differentiable with respect to their neural valuations.

Operation	Differentiable semantics
Unary atom $p(i)$	Read U_{ip} for typed unary relation $p \in \mathcal{P}_1$.
Binary atom $p(i,j)$	Read B_{ijp} for typed binary relation $p \in \mathcal{P}_2$.
Equality / difference	Use slot identity $\mathbf{1}[i=j]$ and its complement.
Conjunction	$\mathrm{And}(a,b) = ab$.
Disjunction	$\mathrm{Or}(a,b) = a+b-ab$.
Negation	$\mathrm{Not}(a) = 1-a$.
Existential quantification	$\mathrm{Exists}(\mu) = 1-\prod_i(1-\widetilde{\mu}_i)$.
Universal quantification	$\mathrm{Forall}(\mu) = \prod_i[1-o_i(1-\mu_i)]$.
Entity selection ι	Return selector weights w_i and unresolved mass $q_0/(\sum_j \widetilde{\mu}_j + q_0 + \epsilon)$.
Selected binary relation	Contract one or both entity axes of B_{ijp} with the corresponding selector weights.
Counting	Compute the Poisson–binomial distribution with the recurrence in Equation 20.
Count comparison	Sum the joint count mass over pairs satisfying $=, \neq, <, \leq, >, \geq$.
Attribute query	For family f, return $\sum_i w_i Q_{i,f}(c)$ for each candidate value c, plus unresolved mass.

C.2 Variable binding and quantification

Intermediate tensors carry named free-variable axes. Unary predicates contribute one axis and binary predicates two; logical operations align axes by variable name before broadcasting. Quantification removes the bound axis. Evaluating a binary predicate on the same variable twice selects its diagonal, while a selected entity contracts the corresponding axis with selector weights. This prevents accidental positional broadcasting from changing variable identity.

For body scores μ_i, occupancy enters quantification as $v_i = o_i\mu_i$. Existential quantification uses $1 - \prod_i(1 - v_i)$. Universal quantification uses $\prod_i(1 - o_i + o_i\mu_i)$, so an absent entity does not violate the universal statement. The typed program determines the scope of each negation, including whether it applies inside or outside a quantifier.

C.3 Entity and attribute readout

For an entity description, $q_0 = \prod_i(1 - v_i)$ is a null-evidence term and $w_i = v_i/(\sum_j v_j + q_0 + \epsilon)$. If no object supports the description, the selector does not become a uniform distribution over arbitrary objects. Its total mass is at most one. A family query returns

$$\widetilde{P}(a = c) = \sum_i w_i Q_{i,f}(c), \qquad \widetilde{P}(a = \text{unknown}) = 1 - \sum_i w_i. \tag{19}$$

The answer family is fixed by the training registry independently of the target label. The loss may sum probabilities for synonymous canonical answers. Unsupported labels, singleton answer families, unbound variables, and type mismatches raise errors.

C.4 Counting and comparisons

Let $D_i(n)$ be the count distribution after considering the first i slots. With $D_0(0) = 1$, the recurrence is

$$D_i(n) = D_{i-1}(n)(1 - v_i) + D_{i-1}(n - 1)v_i. \tag{20}$$

This retains the full support $\{0, \ldots, N\}$ instead of rounding an expected count. For a target $n^\star$, the loss is $-\log D_N(n^\star)$. A comparison of two count distributions uses $\widetilde{P}(A > B) = \sum_{a>b} D_A(a)D_B(b)$, with the corresponding mask for other comparisons.

Two counts involving overlapping predicates can be correlated, in which case the product above approximates their joint distribution. Likewise, product composition gives $p(x)^2$ for the soft score of $p(x) \wedge p(x)$, whereas a reused Boolean sample obeys $b \wedge b = b$. Discrete execution preserves this consistency across repeated atoms, giving a different training signal from the soft surrogate.

C.5 Semantic binding targets

Predicted slots are permutation-free, so supervision first establishes a correspondence with annotated entities. A GT box is converted into a uniform distribution t_g over visual-token centers inside the box and the correct image; if no center falls inside, the nearest center in that image is used. For GT entity g and predicted slot i,

$$C_{gi} = -\sum_p t_{gp} \log(r_{ip} + \epsilon) \tag{21}$$

is the assignment cost, with cross-image matches disallowed. We obtain $\mathcal{M} = \mathrm{Hungarian}(\mathrm{stopgrad}(C))$; C resolves slot permutation but is not added to the training objective.

For a matched pair (g, i), let $\mathcal{A}_g$ contain the direct class labels and typed attributes of g that occur in the fixed training vocabulary. Let $\mathcal{M}_+ \subseteq \mathcal{M}$ retain pairs with $\mathcal{A}_g \neq \varnothing$. The full reduction underlying Equation 11 is

$$\mathcal{L}_{\text{sem}} = \frac{1}{|\mathcal{M}_+|} \sum_{(g,i)\in\mathcal{M}_+} \frac{1}{|\mathcal{A}_g|} \sum_{p\in\mathcal{A}_g} \mathrm{softplus}(-s_{ip}), \tag{22}$$

which equals the positive-label negative log-likelihood because $\mathrm{softplus}(-s_{ip}) = -\log\sigma(s_{ip}) = -\log U_{ip}$. We first average labels within each entity and then average entities, preventing densely annotated entities from dominating. Unannotated concepts remain unknown rather than becoming negatives.

The assignment and attention clustering are discrete, detached decisions. Conditional on them, gradients from $\mathcal{L}_{\text{sem}}$ flow through the selected pooled region features and the trainable visual and text projections W_v, W_t, but not through the assignments. The box-derived cost is used only for correspondence; Stage-1 endpoint supervision $\mathcal{L}_{\text{att}}$ remains responsible for directly training visual localization. GT coordinates are never passed to the FOL executor.

D A WORKED NEURO-SYMBOLIC CHAIN

Figures 1–2 use GQA image 2322897 (375×500 pixels), sample `gqa_025486`, question 05980687: *What animal is to the left of the girl?* The answer is dog, and the released record contains exactly three GT facts:

```
dog[3422204]  -- has_attribute --> old
dog[3422204]  -- to_the_left_of -> girl[3297496]
girl[3297496] -- to_the_right_of -> dog[3422204]
```

The GT boxes $[x_1, y_1, x_2, y_2]$ are dog $[20, 145, 148, 337]$ and girl $[148, 72, 372, 487]$, covering 13.11% and 49.58% of the image. Their intersection area and IoU are zero. We take the image-content crop $[0, 62, 375, 437]$ and resize it to 500×500, producing an $H = W$ image without padding, and store each role-specific attention distribution on a normalized 32×32 square visual-token grid. For visualization only, a distance field inside each hand-traced visible-object silhouette is used to reconstruct the colored overlay. This follows a conventional relation-grounding visualization; it is neither measured model attention nor a dataset segmentation annotation. Pixels outside each silhouette are unchanged.

The three steps yield dog references $\{1\mathrm{S}, 2\mathrm{S}, 3\mathrm{O}\}$ and girl references $\{2\mathrm{O}, 3\mathrm{S}\}$, where S/O denote subject/object roles. Stage 2 merges these references into two entity slots. Old remains an attribute value and creates no slot. The differentiable execution uses the illustrative scores $U_{e_d,\mathrm{dog}} = 0.97$ and $B_{e_d,e_g,\mathrm{left_of}} = 0.95$, giving the illustrative answer score $0.97 \times 0.95 = 0.9215$. The inverse relation and dog attribute remain supervised scene context.

The complete query program is

$$\begin{aligned} e_g &= \iota y\,[\mathrm{girl}(y)], \\ e_d &= \iota x\,[\mathrm{dog}(x) \wedge \mathrm{left_of}(x, e_g)], \\ \varphi_q &= \mathrm{answer}(e_d). \end{aligned} \tag{23}$$

D.1 FROM NEURAL PROBABILITIES TO A HARD SCENE

Each primitive class, binary, attribute, and occupancy fact is sampled with the clipped Bernoulli probability used in its log-likelihood. Figure 2 shows the query-relevant girl-class, dog-class, and `left_of` probabilities $(0.98, 0.97, 0.95)$ and one possible sampled outcome $(1, 1, 1)$. A probability p is the chance of a one, not a deterministic threshold. The illustration assumes both slots are present and no competing class facts are active; the remaining sampled facts are omitted for clarity.

The open query above illustrates Stage 2. Stage 3 in both figures uses the derived verification question *Is a dog to the left of the girl?*, with target yes, as a compact example of discrete execution:

$$\varphi_{\mathrm{verify}} = \exists x\,[\mathrm{dog}(x) \wedge \mathrm{left_of}(x, \iota y\,\mathrm{girl}(y))].$$

This is an illustrative reformulation, not a claim about the released question's answer type. `SELECT UNIQUE girl` resolves the anchor; filtering and conjunction test the candidate; `RETURN EXISTS x` yields yes. With the shown facts, $r_i^s = 1$; a sampled text answer yes gives $r_i^t = 1$, hence $R_i = 1$. The inverse relation and old attribute remain scene context.

For illustrative joint rewards $(1, 0.5, 0, 1, 0.5, 1, 0.5, 0)$, the mean is 0.5625. At $G = 8$, $A_i = \frac{8}{7}(R_i - 0.5625)$, giving 0.5, $-1/14$, or $-9/14$ for rewards 1, 0.5, or 0. Each pair contains one scene; the baseline uses the other seven pairs.

The scene log-probability includes every sampled action, including zero-valued facts:

$$\log \pi_\omega^s(b \mid x) = \sum_\ell [b_\ell \log \hat{p}_\ell + (1 - b_\ell) \log(1 - \hat{p}_\ell)]. \tag{24}$$

The frequency of hard answers estimates success under the sampled scene policy. It need not match either the soft executor's score or the neural decoder's answer confidence.

E THE HARD-EXECUTION CONTRACT

The scene policy samples each program-relevant binary-relation probability as a Bernoulli fact. Relations declared symmetric are sampled once per unordered pair and mirrored. The joint actor retains this contract; aggregate benchmark scores alone do not verify its implementation.

Hard execution ranges over sampled present slots. It reuses identical facts for repeated atoms and supports the answer types admitted by the compiled program, including Boolean connectives, quantifiers, integer counting and comparisons, typed attribute queries, and unique ι denotations. Unresolved or ambiguous readouts are invalid. Unsupported operators cause program errors. Invalid selections earn zero reward; type/registry errors fail qualification rather than being counted as useful negative examples.

Stages 2 and 3 both cap their object-slot representations at 16. Stage 2 activates $k \leq 16$ slots according to the selected question's annotated entity count and masks the remaining capacity, whereas Stage 3 uses the same 16-slot cap for on-policy scene construction. Reference probabilities are evaluated on the detached live regions with the same action ordering. Programs whose answers exceed the configured count capacity are rejected. The fixed Stage-3 capacity removes the need for ground-truth counts during on-policy scene construction, while Stage-2 clustering still uses them.

A target count within capacity is only a necessary check. Establishing uniqueness, negation, or a nested selection can require more entities than the numeric answer indicates. We therefore retain an example only when its program-relevant domain is sufficiently complete within the image capacity; answer agreement alone is not used as evidence of valid execution.

F WHAT DISCRETE EXECUTION GUARANTEES

A semantic sanity check. On a fixed finite domain, hard execution reuses each sampled atom and follows Boolean truth tables. For $B \sim \mathrm{Bernoulli}(q)$ and integer $k \geq 1$,

$$\mathbb{E}[\underbrace{B \wedge \cdots \wedge B}_{k \text{ occurrences}}] = q, \qquad \mathrm{SoftAnd}(q, \ldots, q) = q^k.$$

These follow from Boolean idempotence and product composition. Likewise, $B \wedge \neg B$ is always false, while the soft product gives $q(1-q)$. These operator properties do not predict model performance.

The stages also differ in clipping, slot capacity, eligibility, and selection rules; their objectives need not agree. Invalid ι selections follow the reward contract, and correlated counts need not match products of separate marginals.

Scope of the execution model. Equation 13 counts unique random facts, with mirrored entries treated as lookups. The joint actor adds text actions to this scene policy while preserving the same fact-sharing semantics.

G GRADIENT CONTRACT AND INTERPRETATION

Conditional derivatives. For fixed clustering and matching assignments, the symbolic loss differentiates through selected attention maps, pooled features, and valuation heads, but not discrete assignments or the detached concept embedding lookup. Both actor branches recompute the deterministic latent prefix with gradients enabled; cached rollout states must not sever this path.

For a realized group of rollouts, treating their bits and advantages as fixed gives

$$\frac{\partial \mathcal{L}_{\mathrm{pg}}}{\partial p_\ell} = -\frac{1}{G} \sum_i \mathrm{sg}(A_i) \frac{b_{i\ell} - \hat{p}_\ell}{\hat{p}_\ell(1-\hat{p}_\ell)} \frac{\partial \hat{p}_\ell}{\partial p_\ell}.$$

The clipping derivative is one inside $(0.02, 0.98)$ and zero strictly outside; boundary behavior follows the implementation. Other facts can still update shared parameters. Rewards, sampled bits, and advantages are held fixed.

Joint score-function estimator. Fix the program, action domain, and grading rule. Independent text and fact draws with dropout disabled give $\Pi_\omega = \pi_\theta^t \pi_\omega^s$, even though both branches share parameters. Let $J(\omega) = \mathbb{E}_{\Pi_\omega}[R]$ and $g_i = \nabla_\omega \log \Pi_\omega(Y_i, b_i \mid x)$. For independent pairs, the leave-one-out baseline is independent of pair i, so $\mathbb{E}[b_{-i} g_i] = 0$ and

$$\mathbb{E}\left[\frac{1}{G}\sum_i A_i g_i\right] = \nabla_\omega J, \qquad J = \alpha\mathbb{E}[r^t] + (1-\alpha)\mathbb{E}[r^s].$$

The actor surrogate therefore estimates $-\nabla J$ at the sampling parameters for one on-policy update, conditional on the fixed discrete layout. Pairing supplies a common training signal; the linear reward does not introduce an additional interaction term in the expected objective. Identical-reward groups have zero actor gradient. In particular, at $\alpha = 0.5$, every pair with exactly one correct branch receives 0.5 even if branch outcomes differ.

Reference and reduction conventions. The frozen reference comes from the complete Stage-2 checkpoint. Let $\hat{q}_\ell$ be its clipped fact probability on the detached live layout. We use

$$\mathcal{L}_{\mathrm{KL},s} = \sum_\ell D_{\mathrm{KL}}(\mathrm{Bern}(\hat{p}_\ell) \| \mathrm{Bern}(\hat{q}_\ell)), \tag{25}$$

$$H_s = -\sum_\ell [\hat{p}_\ell \log \hat{p}_\ell + (1-\hat{p}_\ell)\log(1-\hat{p}_\ell)], \tag{26}$$

$$\mathcal{L}_{\mathrm{KL},t} = \frac{1}{G}\sum_i \frac{1}{|Y_i|}\sum_u D_{\mathrm{KL}}(P_\theta(\cdot \mid c_{iu}) \| P_{\mathrm{ref}}(\cdot \mid c_{iu})). \tag{27}$$

Here c_{iu} fixes the input and sampled text history; each model computes its own latent prefix. The text KL averages full categorical distributions at sampled positions, holding the history distribution fixed. It is a prefix-conditional regularizer, not the exact full-sequence KL gradient. Fact KL and entropy sum over unique actions once per input. Actor log-probabilities are summed without length normalization. Sampling and recomputation use the same token distribution and fact clipping; forced positions are excluded.

What a decoder intervention can establish. With the symbolic branch absent at inference, compare relevant-region perturbations with matched irrelevant-region perturbations to test the decoder's use of visual evidence. This measures dependence, not internal execution of the annotated program.

H Additional experimental analyses

H.1 Evaluation protocol and aggregation

For V*, MMVP, BLINK, SEED-Bench-2-Plus, HallusionBench, and HRBench, we follow the same evaluation protocols used by LVR (Li et al., 2025a) and ReGuLaR (Wang et al., 2026c), including benchmark splits, input preprocessing, answer parsing, scorer definitions, denominators, aggregation, and treatment of invalid predictions. MMVP is reported with question-level accuracy, and HRBench-4K/8K retains the cyclic-option evaluation protocol. This shared protocol makes the reported scores directly comparable to those two baselines.

For the primary aggregate, we first average HRBench-4K and 8K, then give HRBench and each of the other five benchmarks equal weight. From the displayed rounded scores, the means are 72.53 for Sylva, 70.20 for ReGuLaR, and 69.09 for LASER. This aggregation does not provide uncertainty estimates.

H.2 Category-level results and reliability

We supplement the aggregate benchmark results with category-level accuracy, model provenance, multi-seed variation, and latent-step efficiency. The SEED-Bench-2-Plus public split contains 810 Chart, 807 Map, and 660 Web questions. HallusionBench contains 625 Easy and 504 Hard questions. MMVP is kept at its official paired aggregate because its question file does not provide per-example category labels.

Table 9: SEED-Bench-2-Plus category accuracy (%) on the public split. Best results are in bold.

Method	Chart	Map	Web
LLaVA-OneVision	55.06	55.02	76.36
Qwen2.5-VL-7B	63.95	54.28	80.45
DeepEyes	66.91	59.97	82.88
PAPO	51.98	54.65	56.06
Vision-R1	67.04	59.73	82.57
LVR	42.96	55.02	43.48
MONET	62.10	55.89	82.73
LASER	67.65	60.59	84.54
ReGuLaR	67.78	60.72	**84.85**
SYLVA	**69.63**	**62.33**	84.70

Table 10: HallusionBench difficulty-slice accuracy (%). Best results are in bold.

Method	Easy	Hard
LLaVA-OneVision	60.32	40.08
Qwen2.5-VL-7B	75.52	34.13
DeepEyes	77.92	42.06
PAPO	70.88	37.70
Vision-R1	79.20	43.05
LVR	82.56	45.43
MONET	69.28	37.10
LASER	84.48	45.43
ReGuLaR	82.56	45.63
SYLVA	**86.40**	**47.62**

Table 11 records the model families and evaluated checkpoints used for the comparisons.

Table 11: Backbone and checkpoint provenance for the evaluated methods.

Method	Backbone	Evaluated checkpoint
LLaVA-OneVision	SigLIP-SO400M + Qwen2-7B	`lmms-lab/llava-onevision-qwen2-7b-ov`
GPT-4o	Closed API model	Official API
Qwen2.5-VL-7B	Qwen2.5-VL-7B	`Qwen/Qwen2.5-VL-7B-Instruct`
DeepEyes	Qwen2.5-VL-7B	`ChenShawn/DeepEyes-7B`
PAPO	Qwen2.5-VL-7B	`PAPOGalaxy/PAPO-G-H-Qwen2.5-VL-7B`
Vision-R1	Qwen2.5-VL-7B	`Osilly/Vision-R1-7B`
LVR	Qwen2.5-VL-7B	`vincentleebang/LVR-7B`
MONET	Qwen2.5-VL-7B	`NOVAglow646/Monet-7B`
LASER	Qwen2.5-VL-7B	`wybb/laser-7b`
ReGuLaR	Qwen2.5-VL-7B	Author-trained checkpoint
SLVR	Qwen2.5-VL-7B	Author-trained checkpoint
SYLVA	Qwen2.5-VL-7B	Anonymous paper checkpoint

H.2.1 MULTI-SEED RELIABILITY AND LATENT-STEP EFFICIENCY

Table 12 reports SYLVA across independent training seeds under the same data, optimization, checkpoint-selection, and evaluation protocols. HRBench-4K and HRBench-8K are reported separately to expose resolution-specific variation. Seed 42 is the run reported in the main results. We report the mean and sample standard deviation across the three runs.

Table 12: SYLVA accuracy (%) across independent training seeds.

Seed	V*	HRBench-4K	HRBench-8K	MMVP	BLINK	SEED-2-Plus	HallusionBench
42	85.86	74.88	69.13	75.33	61.49	71.41	69.09
0	85.34	75.00	68.88	75.00	61.23	70.93	69.44
2026	85.34	74.50	69.00	75.67	61.39	71.06	68.73
Mean $\pm$ std.	85.51 ± 0.30	74.79 ± 0.26	69.00 ± 0.13	75.33 ± 0.34	61.37 ± 0.13	71.13 ± 0.25	69.09 ± 0.36

Figure 3b evaluates 1, 4, 8, and 16 latent steps on V*, HRBench, and MMVP. Averaging the three plotted scores gives 77.73% at four steps and 77.96% at sixteen steps. The 8- and 16-step settings provide no consistent gain over four steps, so we use four steps for the remaining evaluations. TTFA denotes time to first answer token.

Table 13: SYLVA latent-step efficiency. Accuracy averages V*, HRBench, and MMVP from Figure 3b.

Method	Steps	Three-benchmark avg. (%)	TTFA mean (s)	End-to-end median (s)	Peak memory (GiB)
SYLVA	4	77.73	1.58	1.61	$\sim$17
SYLVA	16	77.96	2.01	2.05	$\sim$17

H.3 STAGE-WISE PROGRESSION

Figure 4 in the main text reports the stage-wise progression under a shared four-step evaluation budget. The mean accuracy increases from 67.95% after Stage 1 to 71.18% after Stage 2 and 72.53% after joint RL. Because data, objectives, and training duration change across stages, these values describe curriculum progression rather than a controlled component ablation.

I DISCUSSION AND LIMITATIONS

Learning from a finite world. Executable tasks constrain entity and predicate use during training. The decoder can combine these representations with pretrained knowledge on other questions. Generalization beyond the program language requires controlled concept and composition splits.

Annotation and representation limits. Stage 2 uses annotated entity counts, and repeated attention can assign several slots to one object. Box supervision cannot restore details missing from visual tokens. Sparse scene annotations also limit supervision for negation, universal statements, and counting; missing facts must remain unknown. The current predicate vocabulary does not cover general OCR, global image judgments, or dense cross-image correspondence.

Execution and grounding errors. Soft scores are not calibrated joint probabilities. Hard execution preserves repeated facts, but independently sampled scenes can be incoherent or incorrectly bound. Semantic binding and reference penalties constrain these errors without proving that the decoder used the auxiliary scene.

Empirical scope. Reported results cover Qwen2.5-VL-7B, including the updated joint RL checkpoint. The latent-step ablation covers V*, HRBench, and MMVP, but not the remaining benchmarks or runtime tradeoffs. Stage comparisons conflate objectives, data, and duration. Attribution requires matched controls, multiple seeds, and common evaluators. Other backbones, memory, symbolic calibration, and transfer remain to be evaluated.