PROBABILISTIC REASONING OVER VISUAL EVIDENCE FOR COMPOSITIONAL VQA

Anonymous authors
Paper under double-blind review

ABSTRACT

Training-free compositional methods answer visual questions by chaining pre-trained vision and language models to collect intermediate evidence and compose answers. Methods that execute a program to answer a question make hard decisions on each perception, discarding its uncertainty, while methods that keep perceptions uncertain follow a fixed evidence collection strategy. We introduce PROVE (Probabilistic Reasoning Over Visual Evidence), a training-free neuro-symbolic framework for compositional visual question answering that adaptively collects uncertain visual evidence. PROVE preserves perception uncertainty with calibrated probabilistic logic facts that define a Bayesian network, then answers the query by exact inference with ProbLog. Under matched components on four benchmarks, PROVE outperforms five training-free compositional methods by up to 20.4 points and improves over every VLM it is built on by up to 28.2 points. A deterministic ablation that rounds each perception to true or false shows that preserving uncertainty adds up to 6.8 points, with the gain rising to 8.1 points on questions with the most borderline perceptions. PROVE also inherits gains in LLM reasoning, as thinking mode alone adds up to 26.0 points.

1 INTRODUCTION

Compositional visual question answering (Hudson & Manning, 2019; Johnson et al., 2017a; Suhr et al., 2019; Bogin et al., 2021) requires a system to understand both the visual context and the natural-language query and conduct multi-step reasoning over visual perceptions to produce an answer. Leveraging the code generation capabilities of large language models (LLMs) (Chen et al., 2021; Austin et al., 2021; Gao et al., 2023), training-free compositional methods (Gupta & Kembhavi, 2023; Surís et al., 2023; Ge et al., 2024; Li et al., 2024; Kamali & Kordjamshidi, 2026) have emerged as a promising direction. These methods use the LLM as a planner that writes a program calling various vision models to understand the visual context, and execute that program to answer the query. They need no task-specific training, inherit gains in foundation-model components, and expose an explicit intermediate structure rather than computing the answer in a forward pass.

Despite these advantages, answering questions that require extensive visual reasoning remains challenging for these methods (Wang et al., 2024; Stanić et al., 2024). First, LLMs still struggle with logical reasoning, especially when the reasoning requires multiple steps and needs to consider the relationships between multiple entities (Saparov & He, 2023; Dziri et al., 2023; Mirzadeh et al., 2025). Second, even strong foundation models produce uncertain outputs on visually ambiguous or borderline cases, yet these methods do not effectively maintain this uncertainty while collecting evidence. Methods that resolve a query by executing a program make a hard decisions on the result of each perception, discarding its uncertainty, while methods that keep each perception uncertain cannot make such decisions and fix the evidence collection strategy.

This naturally leads to a question: *how can we adaptively collect evidence while considering perception uncertainty and conduct robust logical reasoning?* To this end, we introduce PROVE (**P**robabilistic **R**easoning **O**ver **V**isual **E**vidence), a training-free neuro-symbolic framework for compositional VQA that adaptively collects visual evidence, represents its uncertainty as calibrated probabilities, and conducts probabilistic logical reasoning to answer the query.

Given a query and one or more images, PROVE introduces an adaptive evidence collection module, in which a coordinator LLM requests detections, verifications, and scene context step by step until sufficient evidence has been collected, representing each image as a set of probabilistic logic facts

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.

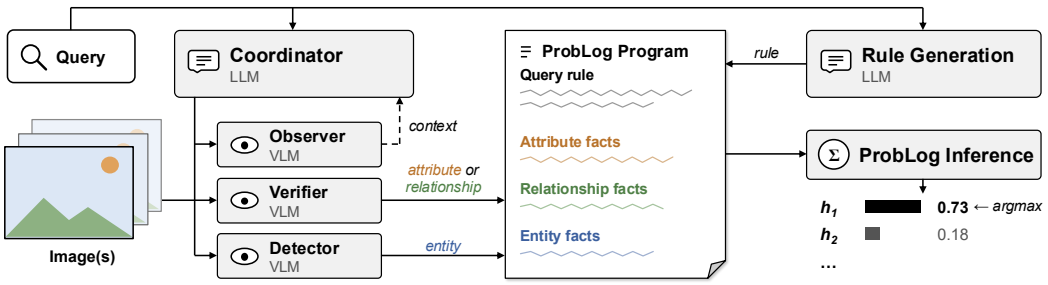


FIGURE 1: Overview of PROVE. An LLM coordinator collects the evidence a query needs through atomic calls to a detector and VLM. Detections and verifications enter a ProbLog program as probabilistic entity, attribute, and relationship facts, while the observer returns visual context only to the coordinator. A rule-generation LLM writes the query rule over these facts, and ProbLog infers each hypothesis's probability exactly. The most probable is the answer, reported with its probability.

(entities, attributes, relationships) that define a Bayesian network. PROVE avoids making a hard decision on a perception while collecting evidence and does not fix its evidence collection strategy, as the coordinator is guided only by the query and its history of actions. PROVE then converts the query into a symbolic logic rule over the collected evidence and computes the probability of each hypothesis, returning the most probable as the answer. By delegating that inference to a symbolic reasoning engine rather than the LLM, PROVE resolves uncertainty only at inference, keeping the logical reasoning robust and carrying the uncertainty of every perception through to the answer.

To evaluate PROVE, we compare it with five training-free compositional methods (Gupta & Kembhavi, 2023; Surís et al., 2023; Ge et al., 2024; Li et al., 2024; Kamali & Kordjamshidi, 2026) on four compositional VQA benchmarks, with every method using the same detector, vision-language model (VLM), and LLM (Section 4.1). PROVE achieves the highest accuracy on all four, improving over the best compositional method by 20.4 points on CLEVR (NePTune), 8.4 on COVR (RVP), 4.8 on NLVR2 (RVP), and 4.0 on GQA (ViperGPT). To isolate the effect of preserving uncertainty, we construct DePROVE, a deterministic ablation that rounds each calibrated perception to true or false and reasons with boolean logic (Section 4.2). PROVE gains 6.8 points on GQA and COVR and 2.6 on CLEVR over DePROVE. Preserving uncertainty matters most where perceptions are borderline, with the gain rising to 8.1 points when the least certain perception lies within 0.1 of the threshold and falling to 1.6 when every perception is at least 0.3 away. PROVE also outperforms every VLM it is built on, by up to 28.2 points over BLIP-2 on CLEVR. Finally, PROVE improves with stronger LLM reasoning, and thinking mode alone adds up to 26.0 points (Section 4.3).

Overall, this work makes the following contributions:

- **C1** We introduce PROVE, a training-free neuro-symbolic framework for compositional VQA that represents adaptively collected visual evidence as calibrated probabilistic facts in a Bayesian network and answers by exact inference with ProbLog.
- **C2** We propose an adaptive evidence collection module guided only by the query and its history of actions, which collects evidence without making a hard decision on any perception and without a fixed evidence collection strategy.
- **C3** We show that, under matched components, PROVE outperforms five training-free compositional methods on four benchmarks, gains most from preserving uncertainty on borderline perceptions, outperforms every VLM it is built on, and improves with the LLM's reasoning.

2 RELATED WORK

Compositional VQA. Compositional VQA (Hudson & Manning, 2019; Johnson et al., 2017a; Suhr et al., 2019; Bogin et al., 2021) requires multi-step reasoning over several visual perceptions. Most work trains end-to-end models that map an image and question directly to an answer (Alayrac et al., 2022; Li et al., 2023a; Liu et al., 2023b; Dai et al., 2023; Bai et al., 2023), and recent work improves their reasoning with test-time reasoning tokens (Wei et al., 2022; OpenAI, 2024) and reinforcement learning (Guo et al., 2025; Kimi Team, 2025), which also teaches them to call visual tools (Zheng et al., 2026; Su et al., 2025a;b). Perception and reasoning still share one forward pass, so their errors cannot be separated and nothing enforces logical reasoning, even when perception uncertainty

is supplied as input (Li, 2026). Compositional methods separate the two by making reasoning an explicit program over vision-model outputs.

Visual Programming. Visual programming (Surís et al., 2023; Gupta & Kembhavi, 2023; Subramanian et al., 2023) has an LLM (Brown et al., 2020; Chen et al., 2021) write programs that coordinate pretrained vision and language models without training, so gains in those models carry over. VisProg (Gupta & Kembhavi, 2023) composes predefined tools through a domain-specific language, ViperGPT (Surís et al., 2023) writes Python over an exposed API, RVP (Ge et al., 2024) decomposes programs recursively, and others improve the prompting (Shen et al., 2024; 2025; Ukai et al., 2024). A second line has an LLM coordinator choose each step from the text of previous tool outputs (You et al., 2023; Hu et al., 2023), or trains one to do so (Ke et al., 2024; Cai et al., 2026). All resolve the query by executing the program, making a hard decision on each perception. ExoViP (Wang et al., 2024) keeps the best-scoring candidate at each step, and TFAR (Zhi et al., 2025) calibrates tool outputs to select the regions an MLLM reasons over, but both use confidence to make a hard decision, not to reason over uncertain perceptions. EVPG (Wan et al., 2025) keeps perceptions uncertain by executing the program as exact inference over a probabilistic graph, but requires fine-tuning its vision models on task labels and fixes its evidence collection strategy.

Neuro-Symbolic Reasoning. Neuro-symbolic reasoning combines neural perception with formal symbolic inference (Yu et al., 2023). Neural module networks (Andreas et al., 2016; Hu et al., 2017; Johnson et al., 2017b) and early neuro-symbolic VQA (Yi et al., 2018; Mao et al., 2019) answer with question-specific module layouts or symbolic programs, and later variants execute probabilistic programs (Vedantam et al., 2019), soft logic (Aditya et al., 2018; Amizadeh et al., 2020), or LLM-written logic over learned grounding modules (Hsu et al., 2023; Kamali et al., 2025), but all require task-specific training. Vieira (Li et al., 2024) embeds foundation models in the probabilistic language Scallop (Li et al., 2023b), but a hand-written per-dataset language fixes what evidence it collects, with the LLM used only to parse the query into it. NAVER (Cai et al., 2025) generates ProbLog (De Raedt et al., 2007) rules for visual grounding, but its inferred probabilities only rank candidate boxes for a VLM to accept or reject. NePTune (Kamali & Kordjamshidi, 2026) scores atomic predicates with a VLM and composes them with fuzzy operators rather than probabilistic inference, taking a VLM's text answer as fact wherever its program needs a decision. These methods keep perceptions uncertain, but each fixes its evidence collection strategy when it writes its program. PROVE needs no per-dataset program, collects evidence guided only by the query and its history of actions without a hard decision on any perception, and answers by probabilistic logical reasoning over all the evidence.

3 METHOD

We present PROVE, a training-free neuro-symbolic framework for compositional VQA that performs probabilistic logical reasoning over uncertain perception. Rather than making a hard decision on each perception, PROVE retains the uncertainty of perceptions as probabilities and resolves it only at inference. Given one or more images and a natural-language query, PROVE returns the most probable answer with its probability. As shown in Figure 1, PROVE operates in two stages: (1) an LLM coordinator adaptively collects the visual evidence (entities, attributes, and relationships) the query needs via tool calls to a VLM and detector, constructing a Bayesian network over it (Sections 3.1 and 3.2); (2) the evidence is converted into ProbLog facts, a rule-generation LLM generates the logic rule equivalent to the query, and ProbLog performs exact inference over the network (Section 3.3).

3.1 BAYESIAN NETWORK OVER VISUAL EVIDENCE

To perform probabilistic inference over uncertain perception, PROVE models visual evidence and query logic as a three-layer Bayesian network depicted in Figure 2. A query defines a hypothesis space $\mathcal{H}$, the set of possible answers. The space is closed when the query determines it and open when the answer must be found in the image. The rule-generation LLM defines its members when it writes the rule (Section 3.3, Section A.4). The network yields the probability that the query holds with each hypothesis $h \in \mathcal{H}$, and the most probable is selected as the final answer. Each perception is a separate observation with its own probability, and the network composes them under two assumptions.

A1 Entities are mutually independent.

A2 Attributes and relationships are conditionally independent given the entities.

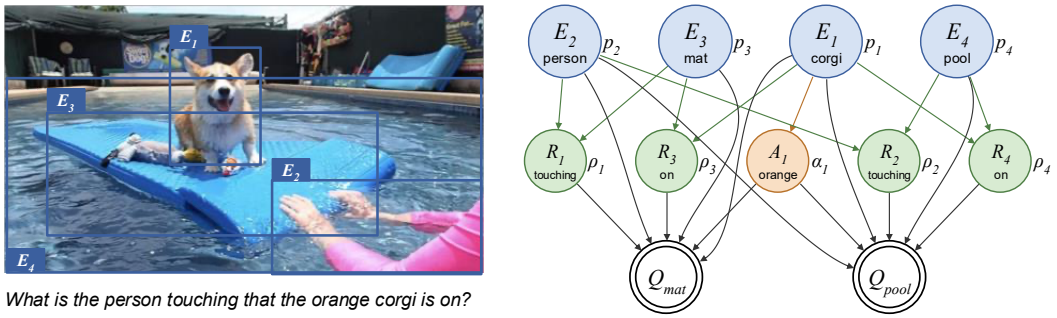


FIGURE 2: The three-layer Bayesian network for "What is the person touching that the orange corgi is on?". Entity nodes E_i carry detection probabilities p_i, attribute and relationship nodes A_j and R_k carry conditional probabilities α_j and ρ_k given their parent entities, and the hypothesis nodes Q_{mat} and Q_{pool} are deterministic functions of the nodes they reference, given by the query rule.

Since multiple perceptions of the same image by the same model may be correlated, we measure the extent to which these assumptions hold in practice in Section 4.4.

Entity layer. The first layer of the network consists of entity nodes $E_i \in \{0, 1\}$, each indicating the presence of an object at a specified location, with probability:

$$P(E_i = 1) = p_i. \tag{1}$$

Each image also contributes a scene entity with probability 1, which allows scene-level attributes (e.g., dark, indoors) and relations relative to the scene (e.g., in the foreground, near the center) to be represented like any other. Facts from every image of a question enter the same network.

Attribute and relationship layer. The second layer contains attribute nodes $A_j \in \{0, 1\}$, each with a single parent entity $E_{\pi(j)}$, and pairwise relationship nodes $R_k \in \{0, 1\}$, each with two parent entities $E_{\pi_1(k)}$ and $E_{\pi_2(k)}$. Each node is parameterized by its conditional probability given the presence of its entities:

$$P\big(A_j = 1 \mid E_{\pi(j)} = 1\big) = \alpha_j \tag{2}$$

$$P\big(R_k = 1 \mid E_{\pi_1(k)} = 1,\ E_{\pi_2(k)} = 1\big) = \rho_k \tag{3}$$

If any of its entities are missing, the node has zero probability as an attribute or relationship cannot hold for a non-existent entity.

Logic layer. The third layer is composed of hypothesis nodes $Q_h \in \{0, 1\}$, one per hypothesis $h \in \mathcal{H}$. Each is a deterministic logical function of the entity, attribute, and relationship nodes it references. It is equal to 1 when they satisfy the query with hypothesis h and 0 otherwise, and adds no uncertainty of its own. This function is the query's logic rule, generated in Section 3.3.

Inference. Inference computes $P(Q_h = 1)$ for each hypothesis node. A scene $S = (E, A, R)$ is an assignment to the first two layers, and since Q_h is a deterministic function of them, each scene determines it. Under A1 and A2, the probability of a scene factorizes as

$$P(S) = \prod_i P(E_i) \prod_j P\big(A_j \mid E_{\pi(j)}\big) \prod_k P\big(R_k \mid E_{\pi_1(k)},\ E_{\pi_2(k)}\big). \tag{4}$$

Writing $\mathcal{S}_h$ for the scenes in which the query holds with hypothesis h,

$$P(Q_h = 1) = \sum_{s \in \mathcal{S}_h} P(s). \tag{5}$$

Each hypothesis is inferred separately, so the probabilities do not need to sum to one over $\mathcal{H}$, and $P(Q_h = 1)$ is the probability that the query holds with h. The answer is the most probable hypothesis, reported with its probability,

$$h^* = \arg\max_{h \in \mathcal{H}} P(Q_h = 1). \tag{6}$$

3.2 Adaptive Evidence Collection

PROVE constructs the network of Section 3.1 incrementally, using a coordinator to determine the visual evidence required to resolve the query. Each piece of evidence is obtained through an atomic

(a) Evidence Collection Trace

#	action		result	p
1	`detect("corgi")`		`ent(corgi_1, corgi)`	0.98
2	`detect("person")`		`ent(person_1, person)`	0.96
3	`verify(corgi_1, "This corgi is orange.")`		`attr(corgi_1, orange)`	0.94
4	`observe("What is the person touching?")`		`"hands in the pool water, gripping the edge of a blue mat"`	—
5	`detect("mat")`		`ent(mat_1, mat)`	0.95
6	`detect("pool")`		`ent(pool_1, pool)`	0.93
7	`verify(person_1, mat_1, "This person is touching this mat.")`		`rel(person_1, mat_1, touching)`	0.92
8	`verify(person_1, pool_1, "This person is touching this pool.")`		`rel(person_1, pool_1, touching)`	0.88
9	`verify(corgi_1, mat_1, "This corgi is on this mat.")`		`rel(corgi_1, mat_1, on)`	0.95
10	`verify(corgi_1, pool_1, "This corgi is on this pool.")`		`rel(corgi_1, pool_1, on)`	0.25
11	`terminate`			

(b) ProbLog Program

PROBABILISTIC FACTS

```
0.98::ent(corgi_1, corgi).
0.96::ent(person_1, person).
0.95::ent(mat_1, mat).
0.93::ent(pool_1, pool).
0.94::attr_v(corgi_1, orange).
0.92::rel_v(person_1, mat_1, touching).
0.88::rel_v(person_1, pool_1, touching).
0.95::rel_v(corgi_1, mat_1, on).
0.25::rel_v(corgi_1, pool_1, on).
```

GUARD RULES

```
attr(E, A) :-
  ent(E, _), attr_v(E, A).
rel(E1, E2, R) :-
  ent(E1, _), ent(E2, _),
  rel_v(E1, E2, R).
```

QUERY RULE

```
q(X) :-
  ent(C, corgi), ent(P, person),
  ent(X, _), attr(C, orange),
  rel(P, X, touching), rel(C, X, on).
```

QUERY PER HYPOTHESIS

```
query(q(mat_1)).  → 0.73
query(q(pool_1)).  → 0.18
```

FIGURE 3: PROVE on "What is the person touching that the orange corgi is on?". (a) The coordinator's actions, the crop each acts on, and its result with calibrated probability p. The observer's answer is context for the coordinator and does not become a fact. (b) The resulting ProbLog program: probabilistic facts, guard rules, the query rule with the hypothesis as its head argument, and one query per hypothesis, which ProbLog answers with 0.73 for the mat and 0.18 for the pool.

call to a vision foundation model, isolating perception uncertainty from logical reasoning. The evidence collection stage loops until the network holds the evidence needed to resolve the query, as traced in Figure 3(a). The same LLM is also used as the rule generator (Section 3.3). We evaluate several LLMs in both roles and several VLMs as the verifier and observer in Section 4.3. The prompt for every role is listed in Section C.

Coordinator. The coordinator is an LLM which reasons over the query and its own history of actions to decide which tool to call next. It collects evidence to resolve the query, and when it lacks the information to decide what evidence to collect, it asks the observer for visual context. Deciding an action requires the LLM to reason, so stronger reasoning capabilities yield better evidence (Section 4.3). The network is initialized with only the scene entities, and the coordinator repeatedly chooses one of three actions, `detect`, `observe`, or `verify`, to build the rest of the network. The coordinator sees the result of every call, which are the entities a `detect` call finds, the fact that a `verify` call checked, or the answer an `observe` call returns, but never a probability. Detection and verification probabilities enter the network directly, withheld from the coordinator so that its reasoning is never conflated with perception uncertainty and it never makes a hard decision on a perception. This sequence of actions collects only the query-relevant components of the scene graph rather than an exhaustive description of the image. The coordinator calls `terminate` once it judges that sufficient evidence has been collected to resolve the query. A termination is refused while an entity the query names has not been detected, and a question that exhausts the action budget must be answered from the evidence collected so far (Section A.3).

Detector. A `detect` call takes an image and an object class and returns every instance as a bounding box with a score, each becoming an entity node whose p_i in Equation (1) is the calibrated score. Scores are calibrated to probabilities on a held-out set described in Section 4.4, and proposals are pruned and overlapping boxes merged before they become entities (Section A.1).

Verifier. We use a VLM as the verifier for its perception alone, passing it only atomic statements so that it is never asked to reason. A `verify` call takes an atomic true/false statement and a crop, and returns the probability that the statement is true. For an attribute verification, the statement references a single attribute and the crop is the entity's detection box; for a relationship verification, it references a single relationship between a pair of entities and the crop is the union of their two boxes. The statement may name any property the VLM can judge, whether concrete, like "red", or abstract, like "looking sad". Since the relevant entities must already have been detected to make the call, the statement and crop assume the entity is present, so "This corgi is orange" and "This person is touching this mat" return the conditional probabilities α_j and ρ_k of Equations (2) and (3). To obtain the confidence, we read the VLM's token-level logits and take the maximum, z_{true}, over the

tokens {`True`, `true`, `TRUE`}, and z_{false} over {`False`, `false`, `FALSE`}. The probability that the statement is true is the sigmoid of their margin, Equation (7), which is calibrated to a probability as described in Section 4.4.

$$P(\text{true}) = \sigma\big(z_{\text{true}} - z_{\text{false}}\big). \tag{7}$$

Observer. We use a VLM as the observer, which processes an `observe` call by taking an image or crop and a question and returns a free-form answer. The coordinator itself never sees the image, and it is the observer's role to supply the visual context the coordinator needs. The answer is never treated as evidence so it contributes nothing to the network. The coordinator instead uses it to disambiguate the query and to form candidate answers that it then verifies. For example, "Is the woman watching the boy with binoculars?" can be interpreted two ways, and asking the observer "Who is holding the binoculars?" resolves which interpretation to verify. In Figure 3(a), for "What is the person touching that the orange corgi is on?" the coordinator asks the observer what the person is touching, extracts the candidates—a mat and a pool—from its description, and detects and verifies each.

3.3 Probabilistic Logical Reasoning

The second stage computes the probability of each hypothesis from the collected evidence and selects the most probable. It does so exactly with ProbLog (De Raedt et al., 2007), a probabilistic logic programming language whose semantics match the network of Section 3.1: its probabilistic facts are independent and carry the probabilities of Equations (1) to (3), and its rules are deterministic and define the attribute, relationship, and hypothesis nodes. Each hypothesis node Q_h of Section 3.1 is the head of a rule over the collected facts, and the success probability ProbLog computes for that head is $P(Q_h = 1)$ of Equation (5). We utilize the code generation capability of the LLM to write the rule as a few lines of logic, as in Figure 3(b).

Converting visual evidence to probabilistic facts. Each entity node becomes a probabilistic fact. Each attribute or relationship node becomes a guard rule that holds only when its entities are present, with an auxiliary probabilistic fact carrying its conditional probability:

```
p_i::ent(e_i, c_i).  α_j::attr_v(e, a).  ρ_k::rel_v(e_1, e_2, r).
attr(E, A) :- ent(E, _), attr_v(E, A).
rel(E1, E2, R) :- ent(E1, _), ent(E2, _), rel_v(E1, E2, R).
```

ProbLog treats probabilistic facts as independent and rules as deterministic, and the query rule reaches `attr_v` and `rel_v` only through the guards, so the program has exactly the distribution of Equation (4).

Query rule generation. A rule-generation LLM translates the query into a ProbLog rule whose head is the hypothesis and whose body composes the collected facts through conjunction, disjunction, and negation, following the grammar in Section B. It may write and use helper rules of the same form. The hypothesis is the argument of the rule's head, bound to each $h \in \mathcal{H}$ by one query per hypothesis, so every hypothesis is scored by the same rule. The LLM takes $\mathcal{H}$ from the query for a closed set and from the candidates the coordinator collected for an open set. If a rule fails to compile or ground, the LLM regenerates it from the error message, up to a fixed number of attempts (Section A.5).

Inference and decision. ProbLog computes $P(Q_h = 1)$ exactly for each hypothesis, one query per h, and the answer is h^* from Equation (6), reported with $P(Q_{h^*} = 1)$. For the example in Figure 3(b), ProbLog returns 0.73 for the mat and 0.18 for the pool, and the mat is the correct answer. Exact inference stays tractable as the answer depends on few facts, one or two for 67% of questions, three to five for 14% of questions, and six or more for only 19% of questions (Table 3), and no query reached the ProbLog time limit (Table 9).

4 Experiments

Benchmarks. We evaluate on four compositional VQA benchmarks that require multi-step reasoning over several perceptions without domain-specific knowledge. We use the balanced test-dev split of GQA, which asks open-ended questions about real images. We use the validation split of CLEVR, which asks questions about synthetic scenes.[1] We use the balanced subset of the test2 split of NLVR2, which asks whether a sentence is true of a pair of photographs. We use the validation split of COVR, which asks questions over a set of real images.[1] From each split we sample 500 questions

[1]The test answers of CLEVR and COVR are not released.

(Section A.6), and every method answers the same set. We report exact-match accuracy with 95% bootstrap confidence intervals, and compare PROVE with DePROVE by a two-sided McNemar test.

Models. PROVE uses its VLM only for observing visual context and verifying atomic statements, and leaves reasoning to the LLM (Sections 3.2 and 3.3). We therefore use a relatively small VLM, Qwen3.5-2B (Qwen Team, 2026), and a capable LLM, Qwen3-32B (Yang et al., 2025) in thinking mode, with SAM3 (Carion et al., 2026) as the detector (Section A.2). In Section 4.3, we further use two additional VLMs, the Flan-T5-XL variant of BLIP-2 (Li et al., 2023a) and the Vicuna-13B variant of LLaVA v1 (Liu et al., 2023b). None of the three VLMs explicitly states that it was trained or tuned on GQA, CLEVR, NLVR2, or COVR. We use LLaVA v1 rather than LLaVA-1.5, since LLaVA-1.5's training data includes GQA. We also experiment with a weaker LLM, Qwen3-8B, with thinking on and off, which gives three levels of LLM reasoning.

Baselines. We compare against five training-free compositional methods, VisProg (Gupta & Kembhavi, 2023), ViperGPT (Surís et al., 2023), RVP (Ge et al., 2024), Vieira (Li et al., 2024), and NePTune (Kamali & Kordjamshidi, 2026). The compositional methods were originally published and scored using different detectors, LLMs, and VLMs. To ensure a fair comparison, we replace each method's original models with Qwen3.5-2B, Qwen3-32B, and SAM3, exactly the components PROVE uses for its main results, and leave the rest of its pipeline unchanged. Section E lists both the original published results and the results we obtained from these compositional methods after the component change. We also compare PROVE against directly using the VLMs to answer the questions, and Section D gives the protocol we used for this comparison. Additionally, we compare PROVE against DePROVE, a deterministic ablation of PROVE that thresholds every perception, described in Section 4.2.

TABLE 1: Accuracy (%) with 95% confidence intervals on 500 questions per benchmark, with every method using Qwen3.5-2B, Qwen3-32B, and SAM3. A dash marks a method that does not support the benchmark's number of images.

Method	GQA	CLEVR	NLVR2	COVR
VisProg	48.6 ±4.6	56.6 ±4.4	74.8 ±3.9	–
ViperGPT	58.2 ±4.2	61.2 ±4.3	–	–
RVP	54.4 ±4.4	69.2 ±4.0	76.6 ±3.7	61.6 ±4.3
Vieira	48.8 ±4.6	73.2 ±4.0	–	–
NePTune	32.4 ±4.5	75.6 ±3.9	–	–
PROVE	**62.2** ±4.0	**96.0** ±1.7	**81.4** ±3.3	**70.0** ±4.0

4.1 Comparison with Training-Free Compositional Methods

PROVE achieves the highest accuracy on all four benchmarks, improving over the best training-free compositional method by 20.4 points on CLEVR (NePTune), 8.4 on COVR (RVP), 4.8 on NLVR2 (RVP), and 4.0 on GQA (ViperGPT). Only VisProg and RVP accept an image pair (NLVR2), and only RVP accepts the larger image sets of COVR, so the remaining methods are evaluated on GQA and CLEVR alone. Since every method uses the same detector, VLM, and LLM, these gains stem from method design rather than the underlying components. The baselines cover both sides of the trade-off in Section 1. VisProg, ViperGPT, and RVP make a hard decision on each perception, while Vieira and NePTune keep perceptions uncertain but fix what evidence to collect when their program is written. PROVE outperforms both by at least 13.4 points on GQA and 20.4 on CLEVR. NePTune suffers most from its fixed evidence collection strategy, since its programs operate only on the entities its noun extractor reads off the question, which leaves most GQA questions without the object they ask about (Section E).

4.2 Effect of Preserved Uncertainty

To isolate the effect of preserving perception uncertainty we construct DePROVE, a deterministic ablation that keeps all of PROVE's evidence and rule but rounds each calibrated detection and verification probability (Section 4.4) to true or false at 0.5, reducing the ProbLog program to boolean logic over deterministic facts. Thresholding after calibration separates the effect of preserving probabilities from the effect of calibrating them. Since DePROVE has no concept of uncertainty, it equally prefers hypotheses that are all true, or all false, and must select one arbitrarily (Section A.6), highlighting the penalty of disregarding perception uncertainty. PROVE gains 6.8 points on GQA, 2.6 on CLEVR, and 6.8 on COVR from preserving uncertainty, each significant under McNemar (Table 2).

TABLE 2: Accuracy (%) of PROVE and DePROVE on the same 500 questions per benchmark from the same evidence and rules. Δ is the percentage-point difference computed before rounding. * marks $p < 0.05$ under a two-sided exact McNemar test.

Method	GQA	CLEVR	NLVR2	COVR
DePROVE	55.4	93.4	81.6	63.2
PROVE	62.2	96.0	81.4	70.0
Δ	+6.8*	+2.6*	−0.2	+6.8*

Table 3 breaks these gains down by perception uncertainty and by reasoning complexity. We measure perception uncertainty by the evidence margin, the distance of the least certain fact the answer depends on from the threshold, as the minimum identifies the single most borderline piece of evidence. We measure reasoning complexity by the number of probabilistic facts the answer depends on. PROVE gains 8.1 points when the least certain fact lies within 0.1 of the threshold but only 1.6 points when every fact is at least 0.3 away, indicating the benefit of preserving uncertainty comes from the most borderline perceptions. The number of facts shows no clear trend, since both methods evaluate the same rule over the same facts. Chaining multiple facts separates the two only when several uncertain facts are chained together.

TABLE 3: Accuracy (%) of PROVE and DePROVE pooled over GQA, CLEVR, NLVR2, and COVR, by the number of probabilistic facts the answer depends on and by evidence margin, the distance of the least certain of those facts from 0.5. The 28 questions whose answer depends on no probabilistic fact are excluded.

	Number of facts				Evidence margin, $\min \lvert p - 0.5 \rvert$			
	1	2	3–5	6+	<0.1	0.1–0.2	0.2–0.3	≥0.3
Questions	837	487	281	367	409	289	299	975
DePROVE	67.7	65.9	78.3	94.0	63.6	59.5	65.2	84.7
PROVE	71.0	72.9	82.6	95.9	71.6	64.4	70.9	86.4
Δ	+3.2	+7.0	+4.3	+1.9	+8.1	+4.8	+5.7	+1.6

4.3 EFFECT OF COMPONENT CHOICE

To measure how PROVE depends on its components, we conduct one experiment that varies the VLM with all other components fixed, and one that varies the LLM with all other components fixed. PROVE gains 11.2–28.2 points over direct VLM answering with BLIP-2, 11.6–23.0 with LLaVA, and 0.8–12.2 with Qwen3.5-2B (Table 4). BLIP-2 and LLaVA gain the most on CLEVR, the most compositional benchmark, where an answer depends on a median of 7 facts against 1–2 on the others. Qwen3.5-2B reasons well over a single image, answering 90.4% of CLEVR questions directly against 57.8% of COVR questions, which span several images, and it gains the most on COVR. Since PROVE uses the VLM only for perception, this suggests that replacing the VLM's reasoning with PROVE's makes up the most ground in the areas where the VLM struggles.

TABLE 4: Accuracy (%) of PROVE by VLM, with each VLM answering directly and serving as PROVE's verifier and observer, with Qwen3-32B and SAM3 fixed. Δ is the gain of PROVE over direct VLM answering.

VLM		GQA	CLEVR	NLVR2	COVR
BLIP-2	Direct	38.4 ±4.3	34.2 ±4.2	55.2 ±4.4	42.4 ±4.3
	PROVE	**51.0** ±4.5	**62.4** ±4.3	**66.4** ±4.2	**56.8** ±4.4
	Δ	+12.6	+28.2	+11.2	+14.4
LLaVA	Direct	35.6 ±4.2	29.4 ±4.1	50.2 ±4.4	39.8 ±4.2
	PROVE	**47.2** ±4.7	**52.4** ±4.4	**64.2** ±4.2	**53.2** ±4.4
	Δ	+11.6	+23.0	+14.0	+13.4
Qwen3.5-2B	Direct	61.4 ±4.7	90.4 ±2.5	77.8 ±3.7	57.8 ±4.4
	PROVE	**62.2** ±4.0	**96.0** ±1.7	**81.4** ±3.3	**70.0** ±4.0
	Δ	+0.8	+5.6	+3.6	+12.2

In the LLM experiment, accuracy rises with the LLM's reasoning (Table 5). Turning on thinking in Qwen3-8B gains 3.2–26.0 points across the four benchmarks, and scaling to Qwen3-32B adds a further 3.6–11.4. The LLM decides what evidence to collect (Section 3.2) and writes the rule that composes it (Section 3.3), and a stronger LLM improves both, so PROVE's accuracy rises.

TABLE 5: Accuracy (%) of PROVE by LLM, with Qwen3.5-2B and SAM3 fixed. Thinking mode is on except where marked †.

LLM	GQA	CLEVR	NLVR2	COVR
Qwen3-8B†	49.4 ±4.5	64.0 ±4.3	66.8 ±4.2	46.2 ±4.4
Qwen3-8B	58.6 ±4.1	90.0 ±2.7	70.0 ±4.1	63.4 ±4.2
Qwen3-32B	**62.2** ±4.0	**96.0** ±1.7	**81.4** ±3.3	**70.0** ±4.0

4.4 RELIABILITY OF PROBABILITY ESTIMATES

Inference multiplies p_i, α_j, and ρ_k (Equation (4)), which is valid only if each is the probability that its fact is true and A1 and A2 hold. We therefore evaluate the calibration of the components, rather than of the pipeline's final answer, and then test both assumptions.

Calibration. The detector's box score and the verifier's logit margin (Equation (7)) are mapped to probabilities by Platt scaling (Platt, 2000),

TABLE 6: Expected calibration error (%, 10 equal-mass bins) of each fact type before and after calibration, on held-out labeled facts.

	Synthetic		Natural	
Fact type	Raw	Cal.	Raw	Cal.
Detector	15.15	0.67	2.47	1.49
Attribute verifier	5.20	0.49	5.12	2.46
Relationship verifier	25.67	0.68	2.65	1.14

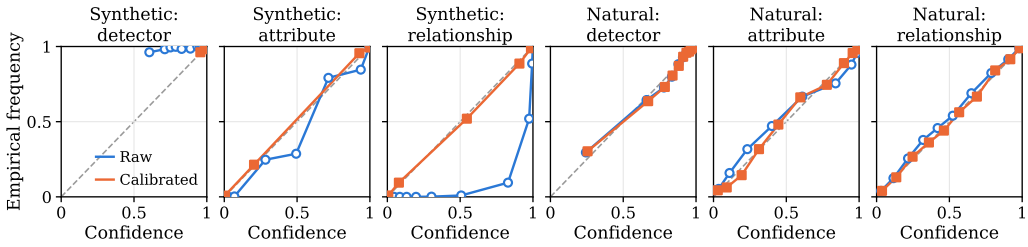


FIGURE 4: Reliability diagrams of each fact type before and after calibration, on held-out labeled facts, with 10 equal-mass bins.

$P(\text{true} \mid s) = \sigma(as + b)$, fitted by maximum likelihood on labeled claims, separately for entities, attributes, and relationships. Since a map holds only for the distribution it is fitted on (Saerens et al., 2002; Alexandari et al., 2020), we fit one set on held-out CLEVR training scenes for CLEVR and one on COCO (Lin et al., 2014), VAW (Pham et al., 2021), and VSR (Liu et al., 2023a) annotations of held-out Visual Genome photographs (Krishna et al., 2017) for GQA, NLVR2, and COVR (Sections F.1 and F.2). Neither set shares an image with the evaluation questions. We report expected calibration error (ECE) with 10 equal-mass bins (Nixon et al., 2019; Roelofs et al., 2022) over every box and statement the modules produced on images not used to fit the maps.

Figure 4 and Table 6 show the reliability curves and ECE before and after calibration. On synthetic scenes the raw scores are far from calibrated, the detector under-confident, the attribute verifier mixed, and the relationship verifier over-confident throughout, with ECE up to 25.67, and calibration brings all three below 1. On photographs the raw scores are already close, with ECE at most 5.12, and calibration roughly halves each error. After calibration every fact type is within 2.5 points, so each parameter of the network is close to the probability that its fact is true.

TABLE 7: Pairwise independence check. Δ is the mean of $y_i y_j - p_i p_j$ over all pairs of held-out facts of the same type in the same image, in percentage points, zero under A1 and A2; gray is half the width of the 95% bootstrap interval over images.

	Synthetic		Natural	
Assumption	Δ	Pairs	Δ	Pairs
A1: entities	-0.30 $\pm$0.68	28,910	-0.49 $\pm$2.68	9,512
A2: attributes	0.06 $\pm$0.13	299,703	-1.18 $\pm$1.78	7,416
A2: relationships	-0.10 $\pm$0.30	404,630	-0.53 $\pm$3.12	1,102

Independence. A1 and A2 imply that any two facts of the same type in one image are independent, so their joint probability is the product $p_i p_j$, $\alpha_i \alpha_j$, or $\rho_i \rho_j$ (Equation (4)). We test this consequence on the held-out calibration facts, each with a truth $y \in \{0, 1\}$ and calibrated probability p, by forming every within-image pair of the same type and reporting Δ, the mean of $y_i y_j - p_i p_j$, which is zero under the assumption and positive when errors are positively correlated (Table 7). On both sets and all three fact types Δ is within 1.2 points of zero with an interval containing zero, so the pairwise products of Equation (4) are within about a point of the joint frequencies they stand for.

5 CONCLUSION

We presented PROVE, a training-free neuro-symbolic framework for compositional VQA that collects visual evidence adaptively without a hard decision on any perception and answers by exact probabilistic inference over a Bayesian network of calibrated facts. Prior compositional methods either round each perception to a fact or keep perceptions uncertain under a fixed evidence collection strategy. PROVE avoids both by letting a coordinator LLM decide what evidence to collect from the query and its history of actions, and by leaving every perception uncertain until inference resolves it. Under matched components, PROVE outperforms five training-free compositional methods on four benchmarks and every VLM it is built on. The gain from preserving uncertainty is largest on borderline perceptions, and accuracy rises with the LLM's reasoning. Its perception probabilities are calibrated, and its independence assumptions hold in practice. With the LLM responsible for reasoning and the VLM for perception, PROVE inherits improvements in both unchanged.

AI USE STATEMENT

We used generative AI tools to assist with writing and editing this paper, limited to copy-editing, tightening prose, and checking that the phrasing of claims matched the supporting results. We did not

use generative AI tools for any of the other tasks for which ICLR requires disclosure. All AI-assisted text was reviewed and revised by the authors, and every claim it touches was checked against the corresponding table, figure, or experimental result. We take responsibility for the final content of this work.

Reproducibility statement

We have taken various steps to ensure the reproduciblity of our work. Sections 3.1 to 3.3 specify the method, and Section C lists every prompt. Section A.2 gives the model checkpoints and decoding settings, Sections A.1, A.3 and A.5 the remaining hyperparameters, and Sections F.1 and F.2 the calibration sets and fitted parameters. Section A.6 describes the question samples and scoring, Section E the baseline adaptations, and Section D the direct VLM protocol. We will release the code publically upon acceptance of this paper.

References

Somak Aditya, Yezhou Yang, and Chitta Baral. Explicit reasoning over end-to-end neural architectures for visual question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pp. 629–637, 2018.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob L. Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikołaj Bińkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karén Simonyan. Flamingo: A visual language model for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 23716–23736, 2022.

Amr Alexandari, Anshul Kundaje, and Avanti Shrikumar. Maximum likelihood with bias-corrected calibration is hard-to-beat at label shift adaptation. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pp. 222–232, 2020.

Saeed Amizadeh, Hamid Palangi, Alex 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)*, volume 119, pp. 279–290, 2020.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 39–48, 2016.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-VL: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023.

Ben Bogin, Shivanshu Gupta, Matt Gardner, and Jonathan Berant. COVR: A test-bed for visually grounded compositional generalization with real images. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9824–9846, 2021.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 1877–1901, 2020.

Zhixi Cai, Fucai Ke, Simindokht Jahangard, Maria Garcia de la Banda, Reza Haffari, Peter J. Stuckey, and Hamid Rezatofighi. NAVER: A neuro-symbolic compositional automaton for visual grounding with explicit logic reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 24078–24089, 2025.

Zhixi Cai, Fucai Ke, Kevin Leo, Sukai Huang, Maria Garcia de la Banda, Peter J. Stuckey, and Hamid Rezatofighi. MATA: A trainable hierarchical automaton system for multi-agent visual reasoning. In *International Conference on Learning Representations (ICLR)*, 2026.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. Sam 3: Segment anything with concepts. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 138846–138923, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/e0982cbc81401df3430ee1ff780dc7a2-Paper-Conference.pdf.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Vincent S. Chen, Paroma Varma, Ranjay Krishna, Michael Bernstein, Christopher Ré, and Li Fei-Fei. Scene graph prediction with limited labels. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2580–2590, 2019. doi: 10.1109/ICCV.2019.00267.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. InstructBLIP: Towards general-purpose vision-language models with instruction tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. ProbLog: A probabilistic Prolog and its application in link discovery. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2468–2473, 2007.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

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)*, volume 202, pp. 10764–10799, 2023.

Jiaxin Ge, Sanjay Subramanian, Baifeng Shi, Roei Herzig, and Trevor Darrell. Recursive visual programming. In *European Conference on Computer Vision (ECCV)*, pp. 1–18, 2024.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pp. 1321–1330, 2017.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025.

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 (CVPR)*, pp. 14953–14962, 2023.

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

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 804–813, 2017.

Ziniu Hu, Ahmet Iscen, Chen Sun, Kai-Wei Chang, Yizhou Sun, David A. Ross, Cordelia Schmid, and Alireza Fathi. AVIS: Autonomous visual information seeking with large language model agent. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023.

Drew A. Hudson and Christopher D. Manning. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6700–6709, 2019.

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 (CVPR)*, 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 (ICCV)*, pp. 2989–2998, 2017b.

Danial Kamali and Parisa Kordjamshidi. NePTune: A neuro-pythonic framework for tunable compositional reasoning on vision-language. In *International Conference on Learning Representations (ICLR)*, 2026.

Danial Kamali, Elham J. Barezi, and Parisa Kordjamshidi. NeSyCoCo: A neuro-symbolic concept composer for compositional generalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 4184–4193, 2025.

Fucai Ke, Zhixi Cai, Simindokht Jahangard, Weiqing Wang, Pari Delir Haghighi, and Hamid Rezatofighi. HYDRA: A hyper agent for dynamic compositional visual reasoning. In *European Conference on Computer Vision (ECCV)*, pp. 132–149, 2024.

Kimi Team. Kimi k1.5: Scaling reinforcement learning with LLMs. *arXiv preprint arXiv:2501.12599*, 2025.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *Int. J. Comput. Vis.*, 123(1):32–73, 2017. doi: 10.1007/S11263-016-0981-7. URL https://doi.org/10.1007/s11263-016-0981-7.

Fabian Küppers, Jan Kronenberger, Amirhossein Shantia, and Anselm Haselhoff. Multivariate confidence calibration for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 326–327, 2020.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202, pp. 19730–19742, 2023a.

Zichao Li. Probabilistic scene graph prompting: Uncertainty-aware structured reasoning in multimodal LLMs. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops*, pp. 1695–1702, 2026.

Ziyang Li, Jiani Huang, and Mayur Naik. Scallop: A language for neurosymbolic programming. *Proceedings of the ACM on Programming Languages*, 7(PLDI):1463–1487, 2023b.

Ziyang Li, Jiani Huang, Jason Liu, Felix Zhu, Eric Zhao, William Dodds, Neelay Velingker, Rajeev Alur, and Mayur Naik. Relational programming with foundational models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 10635–10644, 2024.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (eds.), *Computer Vision – ECCV 2014*, pp. 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1.

Fangyu Liu, Guy Emerson, and Nigel Collier. Visual spatial reasoning. *Transactions of the Association for Computational Linguistics*, 11:635–651, 2023a. doi: 10.1162/tacl_a_00566. URL https://aclanthology.org/2023.tacl-1.37/.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 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 (ICLR)*, 2019.

Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Jeremy Nixon, Michael W. Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

OpenAI. OpenAI o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Khoi Pham, Kushal Kafle, Zhe Lin, Zhihong Ding, Scott Cohen, Quan Tran, and Abhinav Shrivastava. Learning to predict visual attributes in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13018–13028, June 2021.

John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In Alexander J. Smola, Peter L. Bartlett, Bernhard Schölkopf, and Dale Schuurmans (eds.), *Advances in Large Margin Classifiers*, pp. 61–74. MIT Press, 2000.

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

Rebecca Roelofs, Nicholas Cain, Jonathon Shlens, and Michael C. Mozer. Mitigating bias in calibration error estimation. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera (eds.), *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 4036–4054. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/roelofs22a.html.

Marco Saerens, Patrice Latinne, and Christine Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation*, 14(1):21–41, 2002.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *International Conference on Learning Representations (ICLR)*, 2023.

Ruoyue Shen, Nakamasa Inoue, and Koichi Shinoda. Pyramid coder: Hierarchical code generator for compositional visual question answering. In *IEEE International Conference on Image Processing (ICIP)*, pp. 430–436, 2024.

Ruoyue Shen, Nakamasa Inoue, Dayan Guan, Rizhao Cai, Alex C. Kot, and Koichi Shinoda. ContextualCoder: Adaptive in-context prompting for programmatic visual question answering. *IEEE Transactions on Multimedia*, 27:4936–4949, 2025.

Aleksandar Stanić, Sergi Caelles, and Michael Tschannen. Towards truly zero-shot compositional visual reasoning with LLMs as programmers. *Transactions on Machine Learning Research*, 2024.

Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel reasoner: Incentivizing pixel space reasoning via curiosity-driven reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025a.

Zhaochen Su, Linjie Li, Mingyang Song, Yunzhuo Hao, Zhengyuan Yang, Jun Zhang, Guanjie Chen, Jiawei Gu, Juntao Li, Xiaoye Qu, and Yu Cheng. OpenThinkIMG: Learning to think with images via visual tool reinforcement learning. *arXiv preprint arXiv:2505.08617*, 2025b.

Sanjay Subramanian, Medhini Narasimhan, Kushal Khangaonkar, Kevin Yang, Arsha Nagrani, Cordelia Schmid, Andy Zeng, Trevor Darrell, and Dan Klein. Modular visual question answering via code generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 747–761, 2023.

Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. A corpus for reasoning about natural language grounded in photographs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 6418–6428, 2019.

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 (ICCV)*, pp. 11888–11898, 2023.

Mahiro Ukai, Shuhei Kurita, Atsushi Hashimoto, Yoshitaka Ushiku, and Nakamasa Inoue. AdaCoder: Adaptive prompt compression for programmatic visual question answering. In *Proceedings of the 32nd ACM International Conference on Multimedia (ACM MM)*, pp. 9234–9243, 2024.

Ramakrishna Vedantam, Karan Desai, Stefan Lee, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. Probabilistic neural symbolic models for interpretable visual question answering. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pp. 6428–6437, 2019.

Wentao Wan, Kaiyu Wu, Qingyang Ma, Nan Kang, Yunjie Chen, Keze Wang, and Liang Lin. Enhancing visual programming for visual reasoning via probabilistic graphs. *arXiv preprint arXiv:2512.14257*, 2025.

Tzu-Jui Julius Wang, Selen Pehlivan, and Jorma Laaksonen. Tackling the unannotated: Scene graph generation with bias-reduced models. In *31st British Machine Vision Conference 2020, BMVC 2020, Virtual Event, UK, September 7-10, 2020*. BMVA Press, 2020. URL https://www.bmva-archive.org.uk/bmvc/2020/assets/papers/0541.pdf.

Yuxuan Wang, Alan Yuille, Zhuowan Li, and Zilong Zheng. ExoViP: Step-by-step verification and exploration with exoskeleton modules for compositional visual reasoning. In *First Conference on Language Modeling (COLM)*, 2024.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 24824–24837, 2022.

Danfei Xu, Yuke Zhu, Christopher B. Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5410–5419, 2017.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

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 (NeurIPS)*, volume 31, 2018.

Haoxuan You, Rui Sun, Zhecan Wang, Long Chen, Gengyu Wang, Hammad Ayyubi, Kai-Wei Chang, and Shih-Fu Chang. IdealGPT: Iteratively decomposing vision and language reasoning via large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 11289–11303, 2023.

Dongran Yu, Bo Yang, Dayou Liu, Hui Wang, and Shirui Pan. A survey on neural-symbolic learning systems. *Neural Networks*, 166:105–126, 2023.

Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural motifs: Scene graph parsing with global context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5831–5840, 2018.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139, pp. 12697–12706, 2021.

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 (ICLR)*, 2026.

Zhuo Zhi, Chen Feng, Adam Daneshmend, Mine Orlu, Andreas Demosthenous, Lu Yin, Da Li, Ziquan Liu, and Miguel R. D. Rodrigues. TFAR: A training-free framework for autonomous reliable reasoning in visual question answering. *Transactions on Machine Learning Research*, 2025.

A Implementation Details

A.1 Detector Settings

SAM3 candidates are retrieved at a score of 0.005, merged by non-maximum suppression at an IoU of 0.70, and cut relative to the best box of the query, keeping boxes with score at least $\max(\tau,\, \alpha \cdot s_{\max})$, since SAM3 scores are not comparable across class phrases. On natural images we use $\tau = 0.04$, $\alpha = 0.80$, and keep the top 4 boxes per `detect` call; on synthetic images, where counting needs every instance, $\tau = 0.30$, $\alpha = 0.60$, and the top 16. Attribute verifications see the box padded by 15% of its side, and relationship verifications see the padded union of the two boxes.

A.2 Models and Decoding

Table 8 lists the detector and VLM checkpoints. The LLMs, Qwen3-32B and Qwen3-8B, are called with temperature 0, seed 0, and a budget of $6{,}144$ tokens when thinking is on and 256 when it is off. The observer decodes greedily with at most 160 new tokens per `observe` call. Each ProbLog query has a 90 s time limit.

Table 8: Detector and VLM checkpoints on the Hugging Face Hub.

Model	Checkpoint	Revision
SAM3	`facebook/sam3`	`3c879f3`
Qwen3.5-2B	`Qwen/Qwen3.5-2B`	`15852e8`
BLIP-2 Flan-T5-XL	`Salesforce/blip2-flan-t5-xl`	`0eb0d3b`
LLaVA v1 (Vicuna-13B)	`liuhaotian/llava-v1-0719-336px-lora-merge-vicuna-13b-v1.3`	`cf9e42a`

A.3 Coordinator

The coordinator has a budget of 32 actions per question. In the last two actions of the budget only `terminate` is accepted, and a question that reaches the budget without terminating is answered from the evidence collected so far. A termination is refused while an entity the query names has not been detected, and the coordinator is asked to continue. At most 5 terminations are refused per question. A question on which the pipeline produces no rule, no candidate answer, exceeds the ProbLog time limit, or hits a runtime error is counted as incorrect. Table 9 counts these questions on each benchmark.

Table 9: Questions without an answer, out of 500 per benchmark. Each is counted as incorrect.

	GQA	CLEVR	NLVR2	COVR
No rule after the last attempt	0	0	3	0
No candidate answer	7	0	0	5
ProbLog timeout	0	0	0	0
Runtime error	0	0	0	0
Total	7	0	3	5

A.4 Hypothesis Spaces

Table 10 lists how the hypothesis space $\mathcal{H}$ of Section 3.1 is constructed for each question type. For open questions, the coordinator asks the observer the question, or a sub-question, extracts the candidates from its answer, and verifies each; the number of candidates is not capped. The rule-generation LLM writes one query per member of $\mathcal{H}$ (Section 3.3).

A.5 Rule Generation

A generated rule is checked for the expected head, the grammar of Figure 5, and ProbLog grounding and compilation before use. On failure the LLM regenerates it with the error message, for at most 3 attempts. Rules that still fail after the last attempt are counted in Table 9.

TABLE 10: Construction of the hypothesis space by question type.

Question type	$\mathcal{H}$	Source of members
Binary (yes/no, true/false)	$\{\texttt{true}, \texttt{false}\}$	Fixed by the query
Counting	$\{0, \ldots, n\}$	The n detected entities of the counted class
Closed-set (options in the query)	The listed options	Fixed by the query
Open-set	Verified candidates	Extracted by the coordinator from the observer's answers

A.6 EVALUATION PROTOCOL

GQA: 500 questions of the balanced test-dev split over 270 images, 207 binary, 54 closed-set, and 239 open. CLEVR: 500 questions of the validation split, 203 binary, 179 attribute queries, and 118 counting, seed 20260918. NLVR2: 500 statements of the balanced test2 split, one per image pair, 250 true and 250 false, seed 20260923. COVR: 500 questions of the validation split, 274 binary, 69 counting, 99 closed-set, and 58 open, seed 20260923. Answers are lowercased and stripped of surrounding whitespace before exact match; articles are not removed. Counts are compared as digits, and true/false answers as `True`/`False`. Confidence intervals are 95% percentile intervals from 1,000 bootstrap resamples of the 500 questions. When DePROVE's rounded probabilities leave several hypotheses tied, it selects one of them at random. The sampled question identifiers are released with the code.

B EXAMPLE QUERY RULES

$$
\begin{aligned}
&\texttt{q}(X) \texttt{ :- } \mathcal{G}\texttt{.}\\
&\mathcal{G} \quad ::= \mathcal{L} \mid (\mathcal{G} \texttt{ , } \mathcal{G}) \mid (\mathcal{G} \texttt{ ; } \mathcal{G})\\
&\mathcal{L} \quad ::= \mathcal{A} \mid \texttt{\textbackslash+}\ (\mathcal{G})\\
&\mathcal{A} \quad ::= \texttt{ent}(E, C) \mid \texttt{attr}(E, A) \mid \texttt{rel}(E_1, E_2, R)
\end{aligned}
$$

FIGURE 5: Grammar of the query rule. The head $\texttt{q}(X)$ takes the hypothesis X, bound to each $h \in \mathcal{H}$ at query time. A goal $\mathcal{G}$ composes literals by conjunction and disjunction, a literal $\mathcal{L}$ is an atom or a negated goal, and an atom $\mathcal{A}$ is a collected fact. Counts and helper rules are extensions shown in the examples below.

Each example gives a question, its hypothesis space $\mathcal{H}$, and the rule the rule-generation LLM writes over the facts of Section 3.3, following the grammar of Figure 5 and the prompt of Section C. The facts themselves are omitted. A rule refers to entities through variables, which range over the detected entities of the named class. The scene entity of each image is named after the image and has probability one, and `rel(E, S, in)` holds for every entity `E` detected in image `S`. Two extensions of the grammar appear: count predicates, where `count_eq(N, E, G)` holds when exactly `N` distinct bindings of `E` satisfy the goal `G` and is expanded into plain ProbLog before inference, and integer comparison between counts. Every hypothesis is scored by the same rule through one `query(q(h))` per $h \in \mathcal{H}$. For a binary question the false hypothesis is the negation of the true one, so the two probabilities sum to one; we show it once.

Existential. "Is there a red car on the road?" with $\mathcal{H} = \{\texttt{true}, \texttt{false}\}$.

```
q(true) :- ent(C, car), attr(C, red), ent(R, road), rel(C, R, on).
q(false) :- \+ (ent(C, car), attr(C, red), ent(R, road),
    rel(C, R, on)).
```

The free variables `C` and `R` range over the detected cars and roads, so the rule is a disjunction over every pairing.

Negation. "Is there a car that is not red?"

```
q(true) :- ent(C, car), \+ attr(C, red).
```

The `ent` atom is required. Because the network assigns an absent entity's attributes zero probability, `\+ attr(C, red)` alone is satisfied by a car that is not there.

Universal. "Are all the cars red?"

```
q(true) :- \+ (ent(C, car), \+ attr(C, red)).
```

A universal is the negation of a counterexample. It holds vacuously when no car is detected; a rule can add `ent(_, car)` when the question presupposes one.

Disjunction. "Is there either a cat or a dog?"

```
q(true) :- (ent(C, cat) ; ent(D, dog)).
```

The probability is that of at least one disjunct holding, not their sum, since both may hold at once.

Options in the query. "Is the cat on the sofa or on the bed?" with $\mathcal{H} = \{\texttt{sofa}, \texttt{bed}\}$, the options listed in the question.

```
q(X) :- ent(C, cat), ent(O, X), rel(C, O, on).
query(q(sofa)).
query(q(bed)).
```

The hypothesis is bound to a class, so one rule serves every option. The hypotheses are inferred separately, so neither need hold, as when the cat is on the floor, and the answer is the more probable.

Scene-level attribute. "Is it dark outside?"

```
q(true) :- ent(img, scene), attr(img, dark).
```

The scene entity `img` has probability one, so the hypothesis probability is the verifier's probability for the statement.

Position in the image. "On which side of the picture is the car?" with $\mathcal{H} = \{\texttt{left}, \texttt{right}\}$.

```
q(X) :- ent(C, car), ent(img, scene), rel(C, img, X).
query(q(left)).
query(q(right)).
```

A relation to the scene entity is verified on the whole image (Section C), so position in the picture is a relationship fact like any other.

Open set over entities. "What is the man holding?" with $\mathcal{H}$ the candidates the coordinator extracted from the observer's answer and verified, e.g. $\{\texttt{umbrella}, \texttt{bag}\}$.

```
q(X) :- ent(M, man), ent(O, X), rel(M, O, holding).
query(q(umbrella)).
query(q(bag)).
```

A candidate the detector did not find has no `ent` fact, so its hypothesis has probability zero.

Open set over attribute values. "What color is the cube left of the sphere?" with $\mathcal{H}$ the colors proposed by the observer, e.g. $\{\texttt{red}, \texttt{blue}\}$.

```
q(X) :- ent(C, cube), ent(S, sphere), rel(C, S, left_of), attr(C, X).
query(q(red)).
query(q(blue)).
```

The hypotheses are not mutually exclusive, since the network does not forbid a cube from being both red and blue; the answer is the most probable, and its probability is the probability that the query holds with it.

Shared variable. "Does the sphere have the same color as the cube?"

```
q(true) :- ent(S, sphere), ent(Q, cube), attr(S, X), attr(Q, X).
```

`X` ranges over the colors verified for both entities, so the rule is a disjunction over colors of the conjunction of the two verifications.

Helper rule. "Is there a red cube to the left of another red cube?"

```
red_cube(E) :- ent(E, cube), attr(E, red).
q(true) :- red_cube(A), red_cube(B), rel(A, B, left_of).
```

The helper is a deterministic rule of the same form as the query rule and adds no uncertainty; `A` and `B` bind to distinct entities because no relationship fact relates an entity to itself.

Counting. "How many red cars are there?" with $\mathcal{H} = \{0, \ldots, n\}$ for the n detected cars.

```
q(N) :- count_eq(N, C, (ent(C, car), attr(C, red))).
query(q(0)). ... query(q(n)).
```

The count is over distinct bindings of `C`, so an entity is counted once however many facts mention it, and the hypotheses are mutually exclusive and exhaustive.

Count comparison. "Are there more cars than trucks?"

```
q(true) :- count_eq(N1, C, ent(C, car)),
    count_eq(N2, T, ent(T, truck)), N1 > N2.
```

Each count predicate binds its count nondeterministically, so the rule is a disjunction over every pair of counts satisfying the comparison.

Multiple images. "There is a dog in the left image but not in the right image." over an image pair, with the images named `left` and `right`.

```
q(true) :- ent(D, dog), rel(D, left, in),
    \+ (ent(E, dog), rel(E, right, in)).
```

Facts from both images enter one network, and the `in` relation restricts each entity to its image.

Quantifying over images. "Is there a dog in every image?" over a set of images.

```
q(true) :- \+ (ent(S, scene), \+ (ent(D, dog), rel(D, S, in))).
```

The scene variable `S` ranges over the images of the question, so a quantifier over images follows the same pattern as a quantifier over entities.

C PROMPTS

We list the prompt templates for each LLM and VLM role in PROVE. The same prompts are used on every benchmark and for every question type; only the images, the query, and the collected evidence vary. Placeholders are in braces.

Coordinator. The coordinator prompt defines the three actions and `terminate` (Section 3.2).

```
You collect visual evidence for a query about one or more images that you cannot see. A logic rule later
      answers the query from the evidence you collect, so you never answer it yourself.
The images are named {images}. Each image starts as a scene entity with the same name, which stands for the
      whole image.

On each turn, reply with exactly one action:

detect <image> | <class>
    Ask the detector for every instance of one object class in one image. The class is one plain noun
      ("dog", "cube", "person"). You get back entity ids and where each one is: "e2 (left, lower, 12% of
      image)". Each entity is one detected object, and the object the query means may be any of them.
observe <entity> | <question>
    Ask the observer a question about an entity (its crop) or about a whole image (its scene entity). You
      get back a free-form answer. Use it to understand the query and to find the candidate answers; it is
      never evidence itself.
verify <entity> | <attribute> | <statement>
verify <entity>+<entity> | <relation> | <statement>
    Ask the verifier whether one atomic statement is true: one attribute of one entity ("orange", "wooden",
      "sitting", "dark"), or one relation between two entities, subject first ("on", "holding", "left_of").
      Name the attribute or relation in lowercase words joined by underscores, and write the statement as
      one plain sentence about "this" object, assuming it is there: "This corgi is orange.", "This person is
      touching this mat." An attribute of a whole image is verified on its scene entity ("This scene is
      dark."), and so is a relation to the image ("This lamp is on the left side of this image." as e3+img |
      left).
    You only learn that the call was made; the probability goes to the evidence. Verify every candidate
      answer, not only the one you expect, so the rule can compare them.
terminate
    Finish once you judge that the evidence the query needs has been collected, including a verification of
      every candidate answer you found. terminate is refused while an entity the query names has not been
      detected.

Reply with exactly one action and nothing else.
```

On each turn, the coordinator receives this prompt followed by the query and its own history of actions, and nothing else. It chooses exactly one action per turn, so the history has one line per action taken so far, in order. Each line records the call and its result: the entities a `detect` call finds, the fact that a `verify` call was made, or the answer an `observe` call returns, but never a probability. For the query of Figure 3, on the fifth turn, after four actions, the coordinator receives the prompt followed by

```
Images: img. Query: What is the person touching that the orange corgi is on?
detect img | corgi -> e1 (center, lower, 20% of image)
detect img | person -> e2 (right, middle, 30% of image)
verify e1 | orange | This corgi is orange. -> done
observe e2 | What is the person touching? -> "hands in the pool water, gripping the edge of a blue mat"

28 actions left. Your next action:
```

Each line of the history takes exactly one of the following forms, depending on the action taken on that turn; the last form appears when a termination is refused.

```
detect {image} | {class} -> e1 ({horizontal}, {vertical}, {area}% of image), e2 (...)      (or: nothing
      found)
observe {entity} | {question} -> "{free-form answer}"
verify {entity} | {attribute} | {statement} -> done
verify {entity}+{entity} | {relation} | {statement} -> done
terminate refused: {entity the query names} has not been detected
```

Rule generation. The rule-generation prompt gives the fact predicates of Section 3.3, the grammar of Figure 5 with the count extension of Section B, and the queries for each hypothesis space of Table 10. {facts} lists the collected facts without probabilities, and {candidates} lists the candidates of an open query ("CANDIDATES: mat, pool") and is empty otherwise.

```
You write the logic rule that answers a query from the evidence collected about one or more images. ProbLog
      evaluates the rule over the evidence, and the answer is the hypothesis whose query is most probable.
      You never see the images and you never write probabilities.

The evidence is listed under FACTS, without probabilities:
  ent(E, C)        entity E is an object of class C; each image is also a scene entity of class scene, named
      after the image
  attr(E, A)       entity E has attribute A
  rel(E1, E2, R)   entity E1 stands in relation R to entity E2; rel(E, S, in) holds for every entity E
      detected in image S

The rule:
  The head is q(X), where X is the hypothesis. The body combines atoms with "," (and), ";" (or, inside
      parentheses), and "\+ ( ... )" (not). Every entity the body refers to needs its ent atom: \+ attr(E,
      red) alone is also true of an entity that is not there.
  You may define helper rules of the same form.
  count_eq(N, E, G) holds when exactly N distinct bindings of E satisfy the goal G, and counts may be
      compared as integers (>, <, >=, =<, =:=).
  Use classes, attributes, and relations exactly as they are written under FACTS. Use variables for
      entities, except the scene entities, which you may name.
Then write one query per hypothesis, query(q(h)).:
  a true/false query      q(true) :- G.  q(false) :- \+ (G).  query(q(true)). query(q(false)).
  options in the query    one query per option
  an open query           one query per candidate listed under CANDIDATES
  a "how many" query      q(N) :- count_eq(N, E, G).  query(q(0)). ... query(q(n)).  with n the number of
      detected entities of the counted class

FACTS:
{facts}
QUERY: {query}
{candidates}
```

A rule that fails the head check, the grammar check, or ProbLog grounding and compilation is returned with the error:

```
Your previous rule was:
{rule}
It was rejected: {error}
Write the corrected rule and its queries, and nothing else.
```

Verifier. The verifier receives the statement of a verify call and the crop of its entity, or the union of the two boxes for a relationship (Section 3.2), and we read the logits of the first answer token (Equation (7)).

```
Determine if the following statement about this image is true or false.

Statement: "{s}"

Answer with ONLY "True" or "False".
```

Observer. The observer receives the question of an observe call and the crop of its entity, or the whole image for a scene entity.

```
Answer the question about this image in one or two sentences. Say what the answer most likely is and, if you
     are not sure, what else it could be. Describe only what you can see.
Question: {question}
```

The prompt used for direct VLM answering is given in Section D. Baseline methods use their original prompts (Section E).

D Direct VLM Answering Protocol

Each VLM in Table 4 answers every question directly from the image or images and the question, and its answer is scored by the same exact-match protocol as every other method (Section A.6).

Prompt. Questions are asked as they are, with one answer-format instruction and no examples. For Qwen3.5-2B and LLaVA v1, a question is followed by "Answer with a single word or short phrase." and a statement by "Statement: "$\langle s \rangle$" Considering both images together, is this statement TRUE or FALSE? Respond with exactly one word: True or False." BLIP-2 uses its native format, "Question: $\langle q \rangle$ Short answer:" and "Question: Is the following statement true or false? "$\langle s \rangle$" Short answer:". Decoding is greedy with no thinking, with at most 32 new tokens for Qwen3.5-2B, 16 for LLaVA v1 (32 for statements), and 10 for BLIP-2. Images are passed at their native resolution, capped at $1{,}024$ pixels on the long side.

Multiple images. Qwen3.5-2B takes several images natively: the NLVR2 pair is passed as two images labeled "LEFT image:" and "RIGHT image:", and a COVR set as "Image i:" for each photograph, with the prompt saying how many images are shown. BLIP-2 and LLaVA v1 take one image, so the images are composed into one: the NLVR2 pair is placed side by side at a common height with a 16-pixel gap, and a COVR set is tiled into a grid of at most three photographs per row, each fitted into a 512-pixel cell, padded to a square; the prompt says that the picture shows two, or k, separate photographs. No image is answered separately.

Answer extraction. The full generated string is the answer. It is casefolded, whitespace is collapsed, and boundary punctuation and quotes are stripped; number words from zero to twenty are mapped to digits. A statement or yes/no question is scored by the first `true`, `false`, `yes` or `no` token in the output, and an output with none is incorrect. An empty output is incorrect.

E Baselines Under Matched Components

Each compositional baseline was published with its own LLM, VLM, and detector and evaluated on its own question set, so its published accuracy is not directly comparable to ours. Table 11 lists each method's published accuracy next to its accuracy under the matched components of Section 4 on our question sets. Only the matched columns compare methods on the same questions, and the gap between the two columns mixes the change of components with the change of question set. The paragraphs below state for each method which components were replaced and what was left unchanged.

Table 11: Published accuracy (%) of each baseline with its original components, and accuracy under the matched components of Table 1. A dash marks a benchmark the method did not report or does not accept. Published numbers come from each paper's own evaluation set, not our 500-question samples: VisProg reports GQA on up to 20 test-dev questions per question type and NLVR2 on the full public test set, both with majority voting over five prompts; ViperGPT and RVP report GQA test-dev; RVP reports COVR on $1{,}000$ validation questions; Vieira reports Recall@1 on 500 GQA and 480 CLEVR questions; and NePTune reports CLEVR on its first $2{,}000$ questions with InternVL2.5-8B and DeepSeek-V3. PROVE has no separate published numbers; its matched-component row is that of Table 1.

	Published				Matched components			
Method	GQA	CLEVR	NLVR2	COVR	GQA	CLEVR	NLVR2	COVR
VisProg	50.5	–	62.4	–	48.6	56.6	74.8	–
ViperGPT	48.1	–	–	–	58.2	61.2	–	–
RVP	48.8	–	–	52.7	54.4	69.2	76.6	61.6
Vieira	57.9	46.3	–	–	48.8	73.2	–	–
NePTune	–	92.7	–	–	32.4	75.6	–	–
PROVE	–	–	–	–	**62.2**	**96.0**	**81.4**	**70.0**

VisProg. VisProg (Gupta & Kembhavi, 2023) generates a program in its own DSL with GPT-3 from in-context examples and executes it with a fixed set of modules, of which GQA uses ViLT for VQA and OWL-ViT for localization. We keep the released GQA prompt with its 19 examples, the interpreter, and the crop, count, and eval modules, and replace GPT-3 with Qwen3-32B, the VQA module with Qwen3.5-2B answering in at most 32 tokens, and the localizer with SAM3. NLVR2 uses VisProg's released NLVR path: the released prompt, kept verbatim, produces a program over the two images `LEFT` and `RIGHT` whose steps are VQA calls on each image combined by `EVAL`, and the result is the statement's truth value.

ViperGPT. ViperGPT (Surís et al., 2023) generates Python code against its `ImagePatch` API with Codex and executes it; the API is backed by GLIP for detection, BLIP-2 for `simple_query`, X-VLM for `verify_property`, MiDaS for depth, and GPT-3 for `llm_query`. We keep the API, the GQA prompt, the program execution, and the fallback to the VQA model when a program fails, and replace Codex with Qwen3-32B, GLIP with SAM3 (converted to the API's box conventions and crop margin), BLIP-2 with Qwen3.5-2B, and the X-VLM similarity in `verify_property` with Qwen3.5-2B's true/false score. MiDaS is kept and, as in the released GQA configuration, not used.

RVP. RVP (Ge et al., 2024) adds recursion to ViperGPT: the program may call the LLM again on a sub-question, and it runs on the ViperGPT API with GPT-3.5-turbo. We keep the released recursive prompt, executor, and `ImagePatch`, and make the same replacements as for ViperGPT. RVP takes a list of images, so NLVR2 passes the pair and COVR passes the question's two to five photographs in the dataset's order, using RVP's released COVR prompt; the program decides how to combine them.

Vieira. Vieira (Li et al., 2024) translates the question with GPT-4 into a query over a relational program executed by Scallop, whose predicates are grounded by OWL-ViT and CLIP on CLEVR and by OWL-ViT and ViLT on GQA, with spatial relations computed from box geometry. We keep the query language, the Scallop execution with top-k proofs ($k = 3$), and the published geometry, and replace the parser with Qwen3-32B, detection with SAM3, and the property scores with Qwen3.5-2B, which scores each allowed label of a property and is normalized within that property's vocabulary. Vieira's GQA program is not publicly available. Its paper describes the program as an adaptation of VisProg's, so we attempt to implement that adaptation from VisProg's released GQA modules and the same 19 demonstrations. The CLEVR program is the released one.

NePTune. NePTune (Kamali & Kordjamshidi, 2026) generates Python with DeepSeek-V3 over soft tensor operators grounded by Grounding DINO for detection, InternVL2.5-8B for scoring, and SAM 2 with Depth Anything V2 for spatial functions. We keep the prompts, the operators, the spatial functions and their SAM 2 and depth assets, the noun extraction, and the three execution attempts, and replace DeepSeek-V3 with Qwen3-32B, Grounding DINO with SAM3, and InternVL2.5-8B with Qwen3.5-2B scoring yes/no natively. NePTune's GQA accuracy is low because its programs operate only on the entities its noun extractor reads off the question, which suits CLEVR's closed vocabulary but leaves most GQA questions without the object they ask about.

F Calibration Details

F.1 Calibration Sets

Table 12 summarizes the held-out sets of Section 4.4. Each map is fitted on its module's deployment distribution (Guo et al., 2017; Küppers et al., 2020). The detector map is fitted on the proposals SAM3 returns, labeled by matching them to annotated objects, rather than on annotated boxes, since scores on detected and on annotated boxes differ (Xu et al., 2017; Zellers et al., 2018). The verifier maps are fitted on statements in the form the verifier receives at inference time, since a VLM's scores change with the wording (Zhao et al., 2021). The claims are kept at their natural ratio of true to false, since re-balancing them would change the class prior for which the map is calibrated.

Synthetic set. Rendered scenes come with complete annotations of every object, attribute, and relationship, so every label is exact. We build the set from 120 CLEVR training scenes. For the detector, SAM3 is queried for each annotated object class, and a proposal is a positive if it correctly localizes an object of that class (Küppers et al., 2020) and a negative otherwise. For the attribute verifier, each matched object is verified against every value of each attribute type ("The object is red."), the annotated value being true and the others false. For the relationship verifier, every ordered pair of matched objects is verified for the size comparison ("The object outlined in red is larger

than the object outlined in blue."), labeled from the annotated sizes. The maps are fitted on these scenes and evaluated on the facts collected for the evaluation questions, whose scenes carry the same annotations.

Natural-image set. Photographs have no complete annotation. Scene graphs such as GQA's are not exhaustive (Chen et al., 2019; Wang et al., 2020), so a missing annotation cannot serve as a negative. We therefore build the set from Visual Genome photographs (Krishna et al., 2017) that do not appear in the evaluation questions, using only sources that explicitly annotate negatives. For the detector, we use the photographs that COCO also annotates, exhaustively over its 80 classes (Lin et al., 2014). SAM3 is queried for classes present in and absent from each image, and proposals are labeled as above. For the attribute verifier, we use VG photographs from VAW (Pham et al., 2021), which annotates each object with attributes verified to be present or absent, and verify each as a statement ("The shirt is red."). For the relationship verifier, we use VG photographs from VSR (Liu et al., 2023a), which provides statements about a pair of objects with a human-validated truth value ("The cat is behind the laptop."). SAM3 localizes the two objects and the verifier scores the statement on their union crop. Evaluation photographs lack such labels, so the maps are fitted and evaluated by ten-fold cross-fitting over images, each fact scored by a map fitted on the other folds.

TABLE 12: Held-out calibration sets. A fact is a detector proposal or a verifier statement; a positive is a proposal that localizes an annotated object or a statement labeled true.

Fact type	Source	Images	Facts	Positives
Synthetic				
Detector	CLEVR training scenes	120	1,469	1,453
Attribute verifier	CLEVR training scenes	120	10,995	2,932
Relationship verifier	CLEVR training scenes	120	4,376	1,043
Natural				
Detector	COCO annotations of VG photographs	499	2,640	2,159
Attribute verifier	VAW annotations of VG photographs	439	2,078	1,098
Relationship verifier	VSR statements on VG photographs	1,136	1,766	921

F.2 FITTED MAPS

Table 13 lists the fitted parameters of $P(\text{true} \mid s) = \sigma(as + b)$, where s is the logit of the SAM3 score for a detection and the margin $z_{\text{true}} - z_{\text{false}}$ of Equation (7) for a statement, so $a = 1$, $b = 0$ is the uncalibrated map. On the natural set the parameters are the mean over the ten folds; their standard deviation across folds is at most 0.03.

TABLE 13: Fitted Platt parameters with Qwen3.5-2B as the verifier.

	Synthetic		Natural	
Map	a	b	a	b
Detector	2.18	2.37	0.98	0.06
Attribute verifier	0.93	−0.98	0.71	0.07
Relationship verifier	1.45	−5.22	1.00	0.18