NESYCAPE: NEURO SYMBOLIC CONSTRAINT AWARE PROGRAM EXECUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

One effective strategy for handling complex compositional Question Answering is to decompose questions into atomic-concept questions and compose the answers through executable programs, a so-called neurosymbolic methodology. Although effective, these methods often predict atomic concepts locally, which can produce individually plausible but globally inconsistent predictions and lead to incorrect program outputs. To address this limitation, we propose NeSyCAPE, a framework that integrates global consistency constraints into differentiable program execution. We represent each task as a graph of entities, relations, and atomic concepts. A query-specific executable program is then defined over this graph to compose the concepts required to answer the question, while domain knowledge is encoded as logical constraints over the involved concepts. This shared representation allows query-specific execution and global consistency to jointly influence the final answers. During training, executable supervision and differentiable global constraints jointly update the atomic concept predictors. At inference time, constrained optimization combines the neural prediction scores while satisfying the global constraints to produce a feasible concept assignment from which program execution determines the answer. We evaluate NeSyCAPE across three reasoning problems with increasing execution and global-consistency requirements: Temporal QA, Spatial VQA, and KB-VQA. Our results show that executable supervision substantially improves over local prediction baselines, while incorporating global consistency provides further gains in program execution.

1 INTRODUCTION

Answering complex natural language questions given a context often requires identifying entities, their relations, and composing multiple reasoning steps. Spatial and temporal QA provide representative problem settings for studying these challenges for AI models, and several benchmarks have been developed to evaluate them (Johnson et al., 2017; Ning et al., 2018; Mirzaee et al., 2021; Shi et al., 2022). Despite recent progress, AI models still struggle to maintain consistent capabilities and generalize to unseen reasoning compositions (Hsu et al., 2023; Liu et al., 2023; Sinha et al., 2024; Wang et al., 2025; Zhang et al., 2026). To address these challenges, recent neuro-symbolic approaches decompose complex questions into simpler questions about atomic concepts and compose them through program executions (Andreas et al., 2016; Yi et al., 2018; Hsu et al., 2023; Kamali et al., 2025; Kamali & Kordjamshidi, 2026). This decomposition provides a more interpretable reasoning process and can improve compositional generalization compared to direct end-to-end prediction. For instance, LEFT (Hsu et al., 2023) and NeSyCoCo (Kamali et al., 2025) compose language-generated symbolic representations through probabilistic programs using a domain-specific language (DSL), while NePTune (Kamali & Kordjamshidi, 2026) combines soft logical operations with native Python execution. However, even

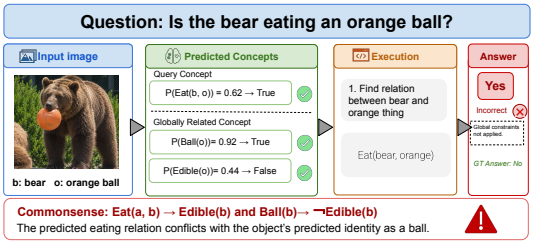


Figure 1: Illustration of conflicting atomic concept predictions. The predicted eating relation conflicts with the object's predicted identity.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

with a correctly specified executable program, the final answer can be incorrect when its underlying atomic predictions are inaccurate or inconsistent with domain constraints. For example, Figure 1 shows an object predicted to be a ball while a bear is also predicted to be eating it. However, eating an object implies that it is edible, while a ball is not edible. Given this domain knowledge, the predictions are not consistent. Because the program relies on the predicted eating relationship to determine the answer, such errors can propagate through program execution and produce an incorrect final answer. Similar inconsistencies arise in temporal and spatial reasoning when independently predicted relations violate properties such as inverse consistency or transitivity (Han et al., 2019; Wu et al., 2025; Premsri & Kordjamshidi, 2025). Prior work has shown that global inference and constraint-aware training can reduce such violations and improve task performance (Ning et al., 2017; 2019; Wu et al., 2025; Premsri & Kordjamshidi, 2025). However, these approaches do not explicitly integrate global constraints with program execution.

To address this gap, we propose NeSyCAPE, which connects compositional program execution and global logical constraints over a set of atomic concept predictions. Figure 2 provides an overview of the framework using an example from the VQAR benchmark (Huang et al., 2021). For each problem, we define a shared graph that specifies the possible concepts and relations in the domain. Given a specific instance that includes an image and a question, detected objects instantiate entities and candidate relations based on this shared graph, while a neural predictor assigns scores to the corresponding atomic concept candidates in that instance. The question is represented by a program execution that composes the query-relevant concepts to produce an answer. The key idea of NeSyCAPE is that program execution and global logical constraints operate over the same grounded concept predictions. During training, answer-level supervision from program execution and global-constraint losses jointly update the concept predictors. At inference, constrained optimization selects concept assignments that satisfy the global constraints before program execution determines the answer. This idea allows these constraints to connect query-relevant concepts to predictions outside the program's immediate execution path; globally related predictions can influence the final answer. Technically, NeSyCAPE translates program-answer conditions into constraints and scores candidate answers through optimization over the shared atomic predictions. We evaluate NeSyCAPE across three reasoning domains: Temporal QA, Spatial-VQA, and KB-VQA, which exhibit different forms of compositional reasoning and global consistency. Across these settings, we examine the contributions of program-execution supervision, global-consistency training, and constrained inference. Our contributions are summarized as follows:

1. We propose NeSyCAPE, which combines question-specific program-execution supervision and global-consistency training over shared grounded atomic concept predictions.

2. We integrate executable queries with constraint-aware inference, allowing global rules to influence the atomic predictions used to produce an answer.

3. We evaluate the contributions of execution supervision, global-consistency training, and constrained inference across Temporal QA, Spatial-VQA, and KB-VQA, including reduced-data settings and different reasoning complexity levels.

2 Related Work

Neuro-Symbolic Reasoning. To solve complex reasoning tasks, several neuro-symbolic methods decompose questions into smaller formal operations and compose them into executable programs. For example, NS-VQA (Yi et al., 2018) predicts a structured scene representation and executes a generated program with a symbolic executor. Other approaches, as we will point to briefly, integrate neural predictions directly into differentiable logic programs. DeepProbLog (Manhaeve et al., 2018) introduces neural predicates into probabilistic logic programming based on Prolog, while Scallop (Huang et al., 2021) supports scalable differentiable reasoning over probabilistic deductive programs based on Datalog. More recent methods, such as LEFT (Hsu et al., 2023) and NePTune (Kamali & Kordjamshidi, 2026), use language models to construct executable logical or Python programs for compositional reasoning. These approaches make compositional reasoning steps explicit and interpretable. However, existing frameworks have not handled deductive reasoning through program execution and global constraints over the program well beyond what the underlying logical languages support. Our work addresses this gap by integrating global logical constraints with question-specific program execution during both training and inference.

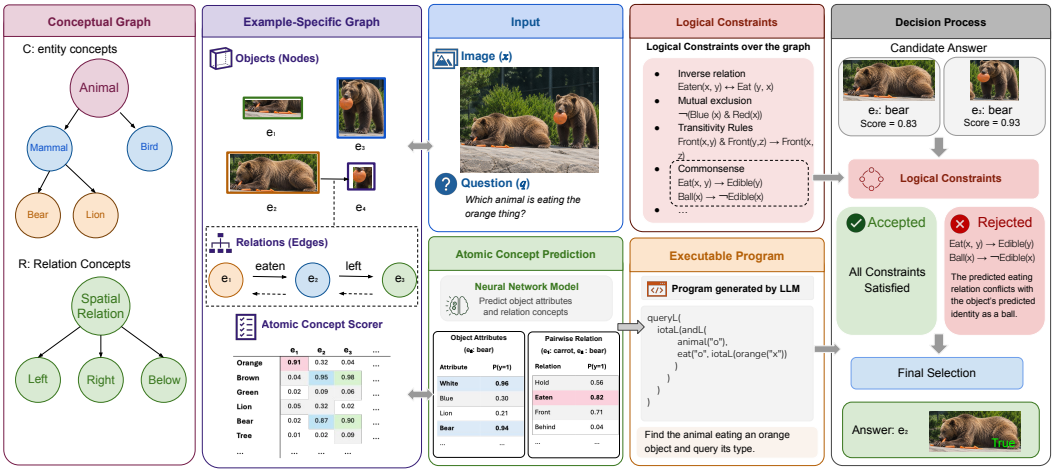


Figure 2: Overview of NeSyCAPE on VQAR. Given an image and a question, neural predictors assign scores to grounded atomic concepts in an example-specific graph. A program execution composes query-relevant concepts to produce the answer, while global constraints enforce consistency over the shared graph.

Logical Consistency. Logical consistency has been considered through training objectives, constrained inference, or both. For example, temporal relation extraction approaches use structured learning and global inference to jointly predict target relations while enforcing consistency constraints (Ning et al., 2017; 2019). Other approaches incorporate logical knowledge directly into neural learning. Semantic Loss (Xu et al., 2018) and DL2 (Fischer et al., 2019) convert logical constraints into differentiable objectives, while Logic Tensor Networks (Badreddine et al., 2022) integrate logical knowledge with learned concepts. DomiKnowS provides declarative graph specifications and supports logical constraints during both training and constrained inference (Rajaby Faghihi et al., 2021). These approaches primarily focus on enforcing consistency over learned predictions rather than studying their interaction with question-specific program execution. Our setting focuses on imposing global constraints on the predictions used in question-specific executable programs. Building on the idea of DomiKnowS, we define executable programs and domain constraints over the same concept predictions. During training, constraint violations provide an additional differentiable loss. At inference, the constraints are enforced through hard-constrained optimization. We also study constraints involving predictions outside the program's execution path.

3 Methodology

3.1 Problem Formulation

We define an input as $x = \{T, I\}$, where T denotes a textual context and I denotes an input image, together with a question q. Either modality may be absent depending on the task setting. The output, denoted as y, can be a class label, relation, entity, or set of entities as an answer to the input question. We keep this formulation generic to cover question answering, visual question answering, and referring expression grounding problem settings used in this paper.

3.2 Graph Representation

To represent the symbolic structure of each example, we assume a set of domain concepts is provided. We represent these as a **conceptual graph**,

$$G = (\mathcal{C}, \mathcal{R}), \qquad \mathcal{C} = \{c_1, \ldots, c_{n_c}\}, \qquad \mathcal{R} = \{r_1, \ldots, r_{n_r}\}$$

where $\mathcal{C}$ contains the possible entity-level concepts, such as object types and attributes, while $\mathcal{R}$ contains the possible relation types. This graph specifies which concepts and relations can be used in representing input and output structures and is shared across all examples during training and

inference; we therefore refer to it as the *shared graph*. Figure 2 shows an example containing entity-level concepts such as BIRD and BEAR, and relation concepts such as BELOW and LEFT.

For each input example x, we then instantiate an **example-specific graph**,

$$G_x = (\mathcal{E}, \mathcal{L}), \qquad \mathcal{E} = \{e_1, e_2, \ldots, e_M\}, \qquad \mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_K\}$$

where $\mathcal{E}$ denotes the set of entities, such as textual event mentions or detected object bounding boxes, while $\mathcal{L}$ denotes the set of edges between candidate entity pairs. Each edge $\ell_k = (e_i, e_j)$ connects two entities that may participate in a relational concept. Figure 2 illustrates an example-specific graph in which entities are instantiated from object bounding boxes and edges connect candidate object pairs. These entities and edges ground the task-level concepts and relations in the current example. A neural model f_θ assigns the probabilities of atomic concepts associated with the entities and edges in the example-specific graph G_x, yielding a set of atomic concept distributions $\mathcal{C}_{G_x}$. We then represent the question as a program execution $\mathrm{Program}_q$ that composes these predictions to estimate the final answer:

$$\hat{y} = \mathrm{Program}_q(G_x, \mathcal{C}_{G_x}).$$

We additionally assume a set of given global logical constraints Φ that encode knowledge over the shared graph. For each example, these constraints are grounded over the corresponding entities and edges in G_x. While $\mathrm{Program}_q$ operates on the concept predictions required to answer the question, the grounded constraints may involve additional concepts and entities outside the execution path. Our goal is therefore to produce atomic concept predictions that support program execution while remaining globally consistent with Φ.

3.3 NEUROSYMBOLIC REPRESENTATION

Graph Construction To obtain the concept graph G, we provide the LLM with a natural-language description of the domain, the concepts and relations involved, and a few in-context examples. After obtaining G, we then instantiate an example-specific graph G_x using the entities and candidate relations identified from the input x for each data instance. For example, in VQA, object bounding boxes in the image define the entities and their corresponding regions. We leverage a visual encoder to extract region features, which are passed to neural predictors to produce scores for the grounded object concepts and relations in G_x. In fact, we implement the graph concepts, relations, and example-specific representations on top of DomiKnowS (Rajaby Faghihi et al., 2021).

Atomic Concept Prediction. For each entity or edge in G_x, the neural model f_θ predicts scores for the corresponding grounded atomic concepts. For example, in VQA, given an image x and an object bounding box o_i, the model predicts the probability that the object has the attribute BROWN: $\widehat{\text{BROWN}}(o_i) = p_\theta(\text{BROWN} \mid x, o_i)$. Predictions for other attributes and relations are obtained similarly. These grounded atomic predictions provide the inputs for program execution.

LLM-based Program Execution. We use an LLM to obtain a question-to-program translation given the domain concepts defined in the shared graph, supported logical and compositional operators[1], and the manually created few-shot examples. The resulting translation yields the $\mathrm{Program}_q$. We validate and revise the generated translation using DomiKnowS compilation and execution feedback. This process occurs only during development of the symbolic representations; therefore, there is no additional call to the LLM during training or inference. Further details are provided in Appendix A.2.

3.4 CONSISTENCY CONSTRAINTS

In addition to the concept graph, we assume there is a set of logical consistency constraints over the values of nodes and edges in the graph. When grounded in G_x, these logical constraints can impose restrictions that are external to the program on the concept values used by $\mathrm{Program}_q$. For example, in the spatial domain, the constraint

$$\text{LEFT}(o_i, o_j) \Rightarrow \text{RIGHT}(o_j, o_i)$$

[1]Logical And, Not, Quantifiers, etc.

constrains the possible predictions for the relations between two objects no matter where these concepts appear in the question's program. These constraints will impact the final assignments to the prediction variables in the program.

LLM-based Consistency Constraints. During symbolic representation development, we use an LLM to obtain task-level global constraints over concepts in the shared graph. The LLM receives a natural language description of the domain, supported logical operators, and manually created few-shot examples. We use DomiKnowS compilation and execution feedback to check and revise the logical constraints. Appendix A.3 details the rule sources, selection, and validation checks.

3.5 Constraint-Aware Inference-Time Execution

Without constrained inference, program execution composes the predicted atomic concept scores $\mathcal{C}_{G_x}$ through differentiable soft-logic operations. For example, under the product t-norm, a conjunction $A \wedge B$ is evaluated as $a \odot b$, where a and b are the corresponding concept probabilities. The highest-scoring candidate gives the unconstrained answer. During training, the same differentiable computation propagates answer supervision to the atomic concept predictors. Supported operators and their translations are detailed in Appendix A.1.

Constraint-Aware Inference. To enable constraint-aware inference, we formulate prediction as an integer linear program (ILP) (Roth & Yih, 2004). Given an example-specific graph G_x, let $\mathbf{z} \in \{0,1\}^N$ denote possible assignments to N grounded binary concepts, and let p_i be the predicted probability that concept i is true. For example, $p_i = p_\theta(\textsc{Red} \mid x, o_j)$ scores whether object o_j is red, while $z_i = 1$ assigns $\textsc{Red}(o_j)$ as true. We score an assignment $\mathbf{z}$ by

$$Q_\theta(\mathbf{z}; G_x) = \sum_{i=1}^{N} \left[z_i \log p_i + (1 - z_i) \log(1 - p_i) \right].$$

For categorical concepts, the corresponding class variables use the log probabilities of their selected states. To incorporate program execution into the same constrained optimization, for each candidate answer $y \in \mathcal{Y}_q$, we encode the condition $\mathrm{Program}_q(G_x, \mathbf{z}) = y$ as an additional constraint over $\mathbf{z}$. Note that when enumerating candidate answers is impractical, we instead execute the program on the highest-scoring feasible concept assignment. These program constraints are combined with the global constraints Φ to define the feasible set

$$\mathcal{F}_{\Phi,q,y}(G_x) = \left\{ \mathbf{z} \mid \mathbf{z} \models \Phi,\ \mathrm{Program}_q(G_x, \mathbf{z}) = y \right\}.$$

We score each candidate answer by the highest-scoring concept assignment that both produces y and satisfies the global constraints

$$F_q(y) = \max_{\mathbf{z} \in \mathcal{F}_{\Phi,q,y}(G_x)} Q_\theta(\mathbf{z}; G_x), \qquad \hat{y} = \arg\max_{y \in \mathcal{Y}_q} F_q(y).$$

where $\mathcal{Y}_q$ denotes the set of candidate answers. We solve the resulting ILP using Gurobi (Gurobi Optimization, LLC, 2026) to obtain the optimal answer $\hat{y}$.

3.6 Constraint-Aware Training-Time Execution

The original logical operations in program execution and global constraints are discrete and therefore cannot directly propagate gradients to the neural predictors. At training time, we evaluate both program executions and global constraints using differentiable soft-logic operations. For example, under the product t-norm (Li et al., 2019), a conjunction $A \wedge B$ is evaluated as ab, where a and b are the predicted probabilities of A and B, respectively. The details of the supported operations are provided in Appendix A.1. This allows answer supervision and, when enabled, global-consistency losses to update the underlying concept predictors.

Let $\mathbf{s}_q$ denote the answer scores produced by differentiable program execution. We define the program-execution loss as $\mathcal{L}_{\mathrm{exec}}(\theta) = \ell_q(\mathbf{s}_q, y)$, where y is the ground-truth answer. The loss function depends on the task output, such as binary cross-entropy for Boolean answers. Appendix A.4 provides the loss used for each output type.

We define $\Phi_x = \{\phi_1, \ldots, \phi_K\}$ as the set of grounded global constraints for example x. Let $h_k(\theta; x)$ denote the aggregated soft-logic violation of constraint ϕ_k. The global-consistency loss is $\mathcal{L}_{\mathrm{GC}}(\theta; x) = \sum_{\phi_k \in \Phi_x} w_k h_k(\theta; x)$, where w_k denotes the weight assigned to each constraint.

Inspired by the Primal-Dual formulation (Nandwani et al., 2019), we combine global-consistency and program-execution supervision as

$$\mathcal{L}(\theta) = \alpha_{\mathrm{exec}} \mathcal{L}_{\mathrm{exec}}(\theta) + \alpha_{\mathrm{GC}} \mathcal{L}_{\mathrm{GC}}(\theta),$$

where α_{exec} and α_{GC} are fixed training weights. Both losses depend on the same concept predictions and therefore update the same neural predictors. For execution-only training, we set $\alpha_{\mathrm{GC}} = 0$.

4 Experimental Setting

4.1 Datasets

We evaluate NeSyCAPE in three settings with different forms of program composition and global constraints: Temporal QA, Spatial-VQA, and KB-VQA. Temporal QA provides a simple executable setting with meaningful global consistency, Spatial-VQA introduces complex compositional execution with relatively simple constraints, and KB-VQA combines compositional execution with a larger and more complex global reasoning structure. Appendix B provides full dataset statistics.

Temporal QA. We use MATRES (Ning et al., 2018) as the first evaluation setting. Each instance queries the temporal relation between two events from {BEFORE, AFTER, EQUAL, VAGUE}. We train on the TimeBank and AQUAINT portions, approximately 10K training examples, and evaluate on all 837 event pairs in the Platinum split. The Temporal QA graph represents the four relation labels and includes global rules such as inverse consistency.

Spatial-VQA. We use 3D-FORCE (Kamali et al., 2026) to evaluate challenging synthetic compositional spatial reasoning under camera-centric and object-centric frames of reference. It contains two tasks: 3D FORCE-SAG, which answers yes/no spatial questions, and 3D FORCE-REF which identifies an object from a referring expression. We train on approximately 10K questions and evaluate on 1,150 SAG instances and 1,822 REF instances. The task graph represents object attributes and spatial relations under both frames of reference.

KB-VQA. We use VQAR (Huang et al., 2021) as the third and most complex setting. Each instance combines visual facts, symbolic knowledge-base facts, and a question, requiring visual grounding together with reasoning over an external knowledge graph. We represent visual objects, symbolic entities, and their relations as atomic concepts that are composed by the program execution. The corresponding graph contains a larger, input-dependent set of concepts and global rules.

4.2 Implementation Details

We implement NeSyCAPE on top of DomiKnowS (Rajaby Faghihi et al., 2021), using its graph declarations, constraint learning, and ILP infrastructure. The conceptual graphs, program executions, and global constraints follow Section 3. For **MATRES**, we use Llama-3.1-8B (Meta, 2024) and Qwen3-8B (Yang et al., 2025) as atomic concept predictors, fine-tuned with LoRA (Hu et al., 2022) while keeping the remaining backbone parameters frozen. For **3D-FORCE**, we encode each image using frozen SigLIP2-SO-400M (Tschannen et al., 2025), which also underlies the Qwen3-VL-8B vision encoder. We pool visual features within each object bounding box and use lightweight MLPs to predict object properties. A separate MLP estimates camera orientation from features across views, and a geometric predictor combines object positions and estimated orientations to predict spatial relations. For **VQAR**, we use Scallop-style MLP predictors over the dataset's precomputed object features and bounding boxes, extracted using fixed Mask R-CNN (He et al., 2017) and ResNet models (He et al., 2016). These predictors estimate atomic object categories, attributes, and pairwise relations. Our neural predictions are connected to graph concepts through the DomiKnowS `ModuleLearner` interface. In the results tables, **Exec.**, **GC**, and **ILP** denote program execution supervision, global-consistency training, and constrained inference, respectively. Appendix A.6 provides all architectures and training configurations. All experiments are run on eight NVIDIA RTX A6000 GPUs.

Table 1: Results on 837 MATRES Platinum event pairs. Micro-F1 is reported over the non-VAGUE classes. Exec., GC, and ILP denote program-execution supervision, global-consistency training, and constrained inference, respectively. Bold marks the best score within each backbone and data regime. **20%** and **100%** denote 20% and 100% training-data settings, respectively. Training and test times are reported in seconds per sample and per question. Published results follow their original protocols; a dash indicates that the result is not reported.

	Training		Inference	Micro-F1 (%)		Time	
Method	**Exec.**	**GC**	**ILP**	**20%**	**100%**	**Train (s/sample)**	**Test (s/q)**
Published reference results							
GDLLM w/o GD (Llama3.1) (Zhao et al., 2025)	–	–	–	–	78.10	–	–
GDLLM (Llama3.1 + GAT) (Zhao et al., 2025)	–	–	–	–	90.90	–	–
NeSyCAPE with LoRA fine-tuning							
Llama3.1-8B (Meta, 2024)							
Direct SFT	–	–	–	47.43	74.18	1.22	0.259
Exec.	✓	–	–	61.63	74.88	1.47	0.288
Exec. + ILP	✓	–	✓	61.63	74.62	–	0.282
Exec. + GC	✓	✓	–	**62.27**	**75.18**	1.57	0.258
Exec. + GC + ILP	✓	✓	✓	**62.27**	74.95	–	0.307
Qwen3-8B (Yang et al., 2025)							
Direct SFT	–	–	–	53.94	78.59	1.31	0.270
Exec.	✓	–	–	**62.65**	**78.73**	1.49	0.249
Exec. + ILP	✓	–	✓	**62.65**	78.52	–	0.289
Exec. + GC	✓	✓	–	60.22	78.46	1.48	0.243
Exec. + GC + ILP	✓	✓	✓	60.22	78.30	–	0.292

Table 2: Results on 3D-FORCE under our evaluation protocol. REF reports top-1 object accuracy, while SAG reports exact-answer accuracy. **20%** and **100%** denote training with 2,000 and 9,747 training questions, respectively. Exec., GC, and ILP denote program-execution supervision, global-consistency training, and constrained inference.

	Training		Inference	REF (%)		SAG (%)		Time efficiency (Full)		
Method	**Exec.**	**GC**	**ILP**	**20%**	**100%**	**20%**	**100%**	**Train (s/sample)**	**REF test (s/q)**	**SAG test (s/q)**
Qwen3-VL-8B (Bai et al., 2025)										
Zero-shot	–	–	–	45.60		52.20		–	1.870	17.04
Direct SFT	–	–	–	56.8	61.5	52.50	52.80	1.18	4.99	18.48
NeSyCAPE: frozen SigLIP2 and predictors										
Execution	✓	–	–	68.94	79.80	68.10	81.30	1.74	2.51	0.65
Execution + ILP	✓	–	✓	74.04	81.06	68.10	81.30	–	9.82	10.15
Execution + GC	✓	✓	–	69.48	81.23	69.20	82.40	2.57	9.87	1.78
Execution + GC + ILP	✓	✓	✓	72.56	80.52	69.20	82.40	–	8.95	9.68

Comparison Baselines. For MATRES, we compare NeSyCAPE with direct supervised fine-tuning (SFT) and a relevant baseline using the same language-model backbones. For 3D-FORCE, we compare against Qwen3-VL-8B in both zero-shot and fine-tuned settings. For VQAR, we report published results from Scallop (Huang et al., 2021), Neural Module Networks (Huang et al., 2021), and LXMERT (Tan & Bansal, 2019) as reference baselines.

4.3 EVALUATION METRICS

For MATRES, we report micro-F1 over the non-VAGUE relation classes on the MATRES Platinum subset. Predicting a non-VAGUE relation for a gold VAGUE pair counts as a false positive. For 3D-FORCE, we report binary-answer accuracy for SAG and top-1 object accuracy for REF. For VQAR, we report Recall@5 across complexity splits C2–C6. For each question, Recall@5 is the number of gold objects among the five highest-ranked candidates divided by $\min(5, |\mathcal{A}_q|)$, averaged over questions with nonempty gold answer sets. Appendix A.5 describes how the ILP-based object ranking is constructed. For efficiency, we report training time in seconds per processed sample and inference time in seconds per question. Both are under the full training data setting. ILP is applied only at inference and incurs no additional training cost.

Table 3: Recall@5 (%) on VQAR C2–C6 with 20% and 100% training data. All NeSyCAPE variants use program-execution supervision; **GC** denotes global-consistency training and **ILP** denotes constrained inference applied on top of the corresponding trained model. Bold and underline indicate the best and second-best NeSyCAPE results within each data regime. Runtime is measured from our replicated implementations on our GPUs; published task results follow their original protocols. A dash indicates an unreported result.

	Training	Inference	Recall@5 (%) – 20%					Recall@5 (%) – 100%					Time Efficiency	
Method	**GC**	**ILP**	**C2**	**C3**	**C4**	**C5**	**C6**	**C2**	**C3**	**C4**	**C5**	**C6**	**Train (s/samp)**	**Test (s/q)**
Published reference results (full data) (Huang et al., 2021)														
LXMERT	–	–	–	–	–	–	–	66.75	61.69	63.82	64.05	56.51	–	–
Neural Module Networks	–	–	–	–	–	–	–	79.32	61.98	71.17	74.62	72.04	–	–
Scallop	–	–	–	–	–	–	–	85.17	82.82	83.25	85.53	84.30	1.02	0.80
NeSyCAPE with trained MLP predictors														
Execution	–	–	82.07	80.75	77.83	80.65	79.57	83.23	82.55	80.49	82.92	80.89	3.54	3.98
Execution + ILP	–	✓	<u>82.24</u>	<u>81.82</u>	<u>80.89</u>	<u>83.38</u>	**82.01**	82.93	<u>83.54</u>	<u>83.45</u>	<u>85.50</u>	**84.26**	–	5.74
Execution + GC	✓	–	**83.06**	80.42	77.61	81.19	79.57	<u>83.60</u>	83.25	82.26	83.45	81.14	4.56	4.10
Execution + GC + ILP	✓	✓	82.10	**82.71**	**81.71**	**84.14**	<u>81.94</u>	**83.97**	**84.82**	**84.02**	**86.09**	<u>83.16</u>	–	6.39

4.4 RESULTS

RQ1. How effective is program-execution supervision compared with direct supervised fine-tuning? We compare direct supervised fine-tuning (SFT) with a setting that uses program execution as the source of supervision (Exec), trained with the whole training dataset. On Temporal QA, where queries often target a single relation, Exec yields only moderate gains for both Llama3.1 and Qwen3 (Table 1). This suggests that when the final answer is closely aligned with a single atomic prediction, program structure provides limited additional supervision. In contrast, Exec provides larger gains on 3D-FORCE and VQAR, where answering the question requires composing multiple concepts (Tables 2 and 3). On VQAR, for example, it outperforms Neural Module Networks by 3.91–20.57 Recall@5 points across C2–C6. The training overhead of program-execution supervision is small on Temporal QA but larger on 3D-FORCE. Overall, program-execution supervision is more useful when the task requires explicit composition, while providing smaller gains when the answer depends primarily on a single relation.

RQ2. What is the impact of global-consistency training along with program-execution supervision? We compare the setting that uses execution as the only source of supervision (Exec.) with the setting that uses both execution and global consistency (GC) without ILP inference. All use the whole training dataset. On Temporal QA (Table 1), GC slightly improves Llama3.1-8B but hurts Qwen3-8B, showing that its benefit is not uniform across predictors. On 3D-FORCE, GC improves REF and SAG by 1.43 and 1.1 points, respectively (Table 2). VQAR also improves across all C2–C6 splits, with gains of 0.25–1.77 Recall@5 points and the largest improvement on C4 (Table 3). Because these comparisons exclude ILP inference, the gains reflect improvements learned during training rather than inference-time constraint enforcement. Overall, GC provides more consistent gains on the compositional visual tasks, while its effect on Temporal QA remains predictor-dependent. These gains come with additional training cost from computing the constraint-violation loss, as reflected in generally increased training times.

RQ3. How does ILP inference affect task performance? We compare models trained on full training data with and without ILP inference. ILP provides the clearest gains on VQAR. Under execution-only supervision (Exec), it improves Recall@5 by 0.99–3.37 points on C3–C6, although C2 drops by 0.30 points. With GC, it improves all C2–C6 splits (Table 3). The benefit is less consistent on 3D-FORCE REF (Table 2). ILP improves Exec accuracy, but with GC it corrects 59 answers while changing 72 previously correct answers to incorrect ones. On Temporal QA, ILP slightly reduces performance for both backbones. Overall, ILP is most consistently beneficial on VQAR, mixed on 3D-FORCE REF, and slightly harmful on Temporal QA. One possible explanation is that VQAR involves richer compositions and domain knowledge, creating more opportunities for global constraints to assist query-relevant predictions. We further examine how global constraints influence VQAR answers through ILP inference (Appendix C.1). Among 80 eligible cases, we reverse the probability of one prediction not directly used by the program but connected through a

global constraint to a concept used during execution. All other probabilities are fixed. The answer set changes in all cases when the connecting constraints are present and remains unchanged when the corresponding constraints are removed. This controlled intervention establishes that predictions not directly used by the program can affect its answer through connecting global constraints; it does not assess the accuracy of the intervened predictions.

5 Ablation Study

RQ4. How data-efficient is NeSyCAPE training? We compare models trained on the whole training dataset with models trained under reduced-data settings. On Temporal QA, execution-only supervision (Exec) is particularly effective with limited data: using 20% of the training set, it improves over direct SFT by 14.20 micro-F1 points for Llama3.1-8B and 8.71 points for Qwen3-8B, substantially larger than the corresponding gains when using the whole training dataset (Table 1). This suggests that program execution provides useful structured supervision when direct answer supervision is limited. The trend differs across the visual tasks. On 3D-FORCE, Exec substantially outperforms direct SFT under both training-data settings (Table 2), although the larger gains with the full training dataset indicate that this task still benefits from additional training examples. In contrast, VQAR retains most of its performance from training on the whole dataset under reduced supervision (Table 3). Overall, NeSyCAPE remains effective with reduced training data, but its data efficiency is task-dependent. Temporal QA benefits strongly from execution supervision in the low-data setting, VQAR retains most of its performance with reduced training data, while 3D-FORCE continues to benefit substantially from additional training examples.

RQ5. How does NeSyCAPE perform across reasoning complexity? We examine VQAR performance across C2–C6 splits. With the whole training dataset, adding both global constraints as additional supervision (GC) and using ILP inference improves Recall@5 at every level, with larger gains on C3–C6 than on C2 (Table 3). The strongest improvements occur on C3–C5, where the GC + ILP setting also exceeds the published Scallop results. At C6, ILP alone still provides a 3.37-point gain over execution, showing that constrained inference remains useful at the highest reported complexity. Overall, NeSyCAPE remains effective across more compositional questions, although the relative contributions of GC and ILP vary by complexity level.

RQ6. How effective are LLM-generated constraints? We evaluate LLM-generated constraints on 20% of the VQAR training data by comparing GT-KB constraints, LLM-generated constraints, and their combination. All settings are fine-tuned from the same baseline. LLM-generated constraints perform comparably to GT-KB constraints across C2–C6, suggesting that useful domain constraints can be obtained from LLMs. Combining both sources does not consistently improve performance, indicating that adding more constraints does not necessarily lead to better downstream reasoning.

Table 4: Constraint-source ablation on VQAR with 20% training data. We report Recall@5 (%), without ILP inference.

Source	C2	C3	C4	C5	C6
GT-KB	81.97	80.70	78.24	81.18	79.46
LLM	82.84	81.45	77.96	81.13	79.43
GT-KB + LLM	83.06	80.42	77.61	81.19	79.57

6 Conclusion

We present NeSyCAPE, a neurosymbolic model that integrates program execution with global logical constraints over neural atomic concept predictions. During training, the constraints provide additional supervision to the concept predictors used by the program, while during inference they are enforced through constrained optimization. We evaluate the framework on Temporal QA, Spatial-VQA, and KB-VQA. Program-execution supervision provides the clearest benefit in low-data and compositional settings. Adding global-consistency training further improves several results, particularly on 3D-FORCE and VQAR, while ILP inference provides the largest gains on VQAR. Our intervention analysis also demonstrates that global constraints allow predictions not directly used by a program to affect its final answer. Overall, the results show that combining program execution with global constraints can improve both learning and inference, although the gains depend on the task, neural predictor, and available domain knowledge.

AI USE STATEMENT

We used LLMs during framework development to assist with graph, program, and constraint construction, and to refine the writing. Generated rules were checked before use; these checks do not guarantee semantic correctness. We also used an AI tool to generate part of a figure. We have reviewed all AI-assisted work.

ETHICS STATEMENT

Our experiments use existing research benchmarks and do not involve human subjects or private data. Our experiments also require substantial GPU resources, which may limit accessibility.

REPRODUCIBILITY STATEMENT

We provide implementation details, task graphs, training settings, constraint configurations, and evaluation procedures in the main paper and appendix. After development-time validation, we fix graphs, program converters, and constraint sets and use them deterministically during training and inference. We plan to release the code, configurations, and evaluation scripts upon publication.

REFERENCES

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

Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger. Logic tensor networks. *Artificial Intelligence*, 303:103649, 2022. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2021.103649. URL https://www.sciencedirect.com/science/article/pii/S0004370221002009.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, et al. Qwen3-VL technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Marc Fischer, Mislav Balunovic, Dana Drachsler-Cohen, Timon Gehr, Ce Zhang, and Martin Vechev. DL2: Training and querying neural networks with logic. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1931–1941. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/fischer19a.html.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2026. URL https://www.gurobi.com.

Rujun Han, Qiang Ning, and Nanyun Peng. Joint event and temporal relation extraction with shared representations and structured prediction. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 434–444, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1041. URL https://aclanthology.org/D19-1041/.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

Joy Hsu, Jiayuan Mao, Joshua B. Tenenbaum, and Jiajun Wu. What's left? concept grounding with logic-enhanced foundation models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=sq4o3tjWaj.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jiani Huang, Ziyang Li, Binghong Chen, Karan Samel, Mayur Naik, Le Song, and Xujie Si. Scallop: From probabilistic deductive databases to scalable differentiable reasoning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 25134–25145. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/d367eef13f90793bd8121e2f675f0dc2-Paper.pdf.

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, 2017.

Danial Kamali and Parisa Kordjamshidi. NePTune: A neuro-pythonic framework for tunable compositional reasoning on vision-language. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=8H0TkSusWI.

Danial Kamali, Elham J. Barezi, and Parisa Kordjamshidi. Nesycoco: a neuro-symbolic concept composer for compositional generalization. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i4.32439. URL https://doi.org/10.1609/aaai.v39i4.32439.

Danial Kamali, Tanawan Premsri, Shreya Rajpal, Amir Zadeh, Chuan Li, and Parisa Kordjamshidi. Saturn: Symbolic spatial reasoning for multi-perspective grounding, 2026. URL https://arxiv.org/abs/2606.22694.

Tao Li, Vivek Gupta, Maitrey Mehta, and Vivek Srikumar. A logic-driven framework for consistency of neural models. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3924–3935, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1405. URL https://aclanthology.org/D19-1405/.

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

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: neural probabilistic logic programming. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 3753–3763, Red Hook, NY, USA, 2018. Curran Associates Inc.

Meta. Introducing llama 3.1: Our most capable models to date, 2024. URL https://ai.meta.com/blog/meta-llama-3-1/.

Roshanak Mirzaee, Hossein Rajaby Faghihi, Qiang Ning, and Parisa Kordjamshidi. SPARTQA: A textual question answering benchmark for spatial reasoning. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4582–4598, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.364. URL https://aclanthology.org/2021.naacl-main.364/.

Yatin Nandwani, Abhishek Pathak, Mausam, and Parag Singla. A primal dual formulation for deep learning with constraints. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox,

and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/cf708fc1decf0337aded484f8f4519ae-Paper.pdf.

Qiang Ning, Zhili Feng, and Dan Roth. A structured learning approach to temporal relation extraction. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel (eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1027–1037, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1108. URL https://aclanthology.org/D17-1108/.

Qiang Ning, Hao Wu, and Dan Roth. A multi-axis annotation scheme for event temporal relations. In Iryna Gurevych and Yusuke Miyao (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1318–1328, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1122. URL https://aclanthology.org/P18-1122/.

Qiang Ning, Sanjay Subramanian, and Dan Roth. An improved neural baseline for temporal relation extraction. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 6203–6209, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1642. URL https://aclanthology.org/D19-1642/.

Tanawan Premsri and Parisa Kordjamshidi. Neuro-symbolic training for reasoning over spatial language. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 2395–2414, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.128. URL https://aclanthology.org/2025.findings-naacl.128/.

Hossein Rajaby Faghihi, Quan Guo, Andrzej Uszok, Aliakbar Nafar, and Parisa Kordjamshidi. DomiKnowS: A library for integration of symbolic domain knowledge in deep learning. In Heike Adel and Shuming Shi (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 231–241, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-demo.27. URL https://aclanthology.org/2021.emnlp-demo.27/.

Dan Roth and Wen-tau Yih. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pp. 1–8, Boston, Massachusetts, USA, May 6 - May 7 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-2401/.

Zhengxiang Shi, Qiang Zhang, and Aldo Lipani. Stepgame: A new benchmark for robust multi-hop spatial reasoning in texts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 11321–11329, Jun. 2022. doi: 10.1609/aaai.v36i10.21383. URL https://ojs.aaai.org/index.php/AAAI/article/view/21383.

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. ISSN 2835-8856. URL https://openreview.net/forum?id=BXDxwItNqQ. Survey Certification.

Hao Tan and Mohit Bansal. LXMERT: Learning cross-modality encoder representations from transformers. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 5100–5111, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1514. URL https://aclanthology.org/D19-1514/.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff,

Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. Siglip 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features, 2025. URL https://arxiv.org/abs/2502.14786.

Xingrui Wang, Wufei Ma, Tiezheng Zhang, Celso M de Melo, Jieneng Chen, and Alan Yuille. Spatial457: A diagnostic benchmark for 6d spatial reasoning of large multimodal models. 2025. URL https://arxiv.org/abs/2502.08636.

Jiarui Wu, Zhuo Liu, and Hangfeng He. Mitigating hallucinations in multimodal spatial relations through constraint-aware prompting. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3450–3468, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.192. URL https://aclanthology.org/2025.findings-naacl.192/.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5502–5511. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/xu18h.html.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, et al. 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 *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 1039–1050, Red Hook, NY, USA, 2018. Curran Associates Inc.

Pingyue Zhang, Zihan Huang, Yue Wang, Jieyu Zhang, Letian Xue, Zihan Wang, Qineng Wang, Keshigeyan Chandrasegaran, Ruohan Zhang, Yejin Choi, Ranjay Krishna, Jiajun Wu, Li Fei-Fei, and Manling Li. Theory of space: Can foundation models construct spatial beliefs through active exploration? In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=8iPwqr6Adk.

Jie Zhao, Wanting Ning, Yuxiao Fei, Yubo Feng, and Lishuang Li. GDLLM: A global distance-aware modeling approach based on large language models for event temporal relation extraction. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 8080–8091, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.428. URL https://aclanthology.org/2025.findings-emnlp.428/.

A Implementation Details

A.1 Logical T-Norm Translation

Table 5 presents the t-norm soft-logic translation for the function used in DomiKnowS. For counting constraints, let $\mathbf{a} = (a_1, \ldots, a_n)$ denote the input truth scores, where $a_i \in [0, 1]$. The score for exactly k true inputs is

$$P_k(\mathbf{a}) = \sum_{\substack{S \subseteq \{1,\ldots,n\} \\ |S|=k}} \prod_{i \in S} a_i \prod_{j \notin S} (1 - a_j),$$

where S indexes inputs assigned true, and its complement indexes inputs assigned false.

A.2 Graph Construction

We construct conceptual graphs during framework development using GPT-5.6-Terra. The model receives the task schema, domain description, and manually prepared few-shot graph examples, then generates graph-construction code that defines the concepts, relations, and their connections. The generated graph is validated in DomiKnowS and revised using execution feedback when necessary.

Table 5: Soft-logic translations under the DomiKnowS product configuration. Here, $a, b, a_i \in [0, 1]$ denote input truth scores. Counting expressions assume product-based counting and $0 \leq k \leq n$.

Operator	DomiKnowS	Soft Logic Form
Negation	`notL`	$1 - a$
Conjunction	`andL`	$\prod_{i=1}^{n} a_i$
Disjunction	`orL`	$1 - \prod_{i=1}^{n}(1 - a_i)$
Negated conjunction	`nandL`	$1 - \prod_{i=1}^{n} a_i$
Negated disjunction	`norL`	$\prod_{i=1}^{n}(1 - a_i)$
Implication	`ifL`	$\begin{cases} 1, & a \leq b, \\ b/a, & a > b \end{cases}$
Exclusive disjunction (binary)	`xorL`	$a(1 - b) + (1 - a)b$
Equivalence (binary)	`equivalenceL`	$u + v - uv, u = ab, v = (1 - a)(1 - b)$
Existential quantification	`existsL`	$1 - \prod_{i=1}^{n}(1 - a_i)$
Exactly k true groundings	`exactL`	$P_k(\mathbf{a})$
At least k true groundings	`atLeastL`	$\sum_{j=k}^{n} P_j(\mathbf{a})$
At most k true groundings	`atMostL`	$\sum_{j=0}^{k} P_j(\mathbf{a})$

Graph Generation. The generator receives the domain description, entity types, applicable atomic concepts, relation signatures, and examples from manually implemented DomiKnowS graphs. In VQA, for example, the resulting graph connects visual objects, symbolic entities, and visual and knowledge-base relations, allowing program execution and global constraints to operate over the same graph concepts.

Graph Representation. The generated graph defines the correspondence between task entities and DomiKnowS concepts and relations. Each atomic concept is associated with its applicable entity or relation type, while relational concepts preserve the required argument order.

Validation and Revision. We instantiate representative task examples and check whether (1) the required concepts and relations are correctly defined, (2) entity bindings and relation arguments can be resolved, and (3) the resulting graph and logical expressions can be constructed and executed. When validation fails, the DomiKnowS error feedback and relevant code are provided to the LLM for revision. This process is repeated until the graph is executable or the candidate is discarded.

A.3 Constraint Construction

We construct global constraints using GPT-5.6-Terra from the task schema, domain description, and manually created few-shot examples. Related concepts are grouped into semantic families, and generated constraints are translated into DomiKnowS logic and validated before inclusion in the final rule set. Invalid constraints are revised using DomiKnowS feedback or discarded, and duplicate constraints are removed. The final constraint set is fixed before downstream training and inference, with no LLM calls at runtime.

Rule Generation. The generator is instructed to use only predicates and logical operators supported by the task schema. Each generated rule specifies its predicate bindings and logical structure,

which are translated into DomiKnowS logic. For example,

$$\forall o_i, o_j, \qquad \textsc{Left}(o_i, o_j) \Rightarrow \textsc{Right}(o_j, o_i).$$

The translation preserves predicate identity and argument order.

Validation and Revision. We validate each generated rule by checking whether it (1) follows the expected structure, (2) uses valid predicates with compatible argument types, and (3) can be constructed and executed within the corresponding DomiKnowS graph. Failed rules are revised using DomiKnowS diagnostic feedback for up to three rounds. Rules that remain invalid are discarded, and duplicate constraints are removed before constructing the final rule set. These checks establish that a rule is well-formed and executable, but do not guarantee its semantic correctness.

A.4 Task-Specific Program Execution Loss

Program execution produces different types of answers across tasks, so we use the appropriate execution loss for each answer type.

For Temporal QA, the program produces scores $s_q(c)$ for the four temporal relations. We apply multiclass cross-entropy to the ground-truth relation y:

$$\mathcal{L}_{\text{exec}}^{\text{Temporal}}(q) = -\log \frac{\exp s_q(y)}{\sum_{c \in \mathcal{Y}_q} \exp s_q(c)}.$$

For 3D-FORCE SAG, the program produces a Boolean answer probability, which we train with binary cross-entropy.

For object-selection queries, the program produces a selection score $\hat{a}_j \in [0, 1]$ for each candidate object o_j. We apply binary cross-entropy against the gold membership vector, which is one-hot for single-target REF questions and may contain multiple positive entries for VQAR:

$$\mathcal{L}_{\text{exec}}^{\text{obj}}(q) = -\frac{1}{M_q} \sum_{j=1}^{M_q} \left[a_j \log \hat{a}_j + (1 - a_j) \log(1 - \hat{a}_j) \right],$$

where M_q is the number of candidate objects and $a_j \in \{0, 1\}$ indicates whether o_j belongs to the gold answer.

In each case, supervision is applied to the program output, and gradients propagate through its differentiable operations to the underlying atomic concept predictors.

A.5 ILP Ranking

For set-valued programs that require top-k ranking, we perform the following process to obtain the ILP results. Rather than enumerating all possible answer sets, we introduce a binary membership variable u_a for each candidate object a, with hard constraints enforcing

$$u_a \iff \left[a \in \text{Program}_q(G_x, \mathbf{z}) \right].$$

For ranked evaluation, we compute

$$F_{q,a}(b) = \max_{\substack{\mathbf{z} \models \Phi \\ u_a = b}} Q_\theta(\mathbf{z}; G_x), \qquad b \in \{0, 1\},$$

subject to the compiled program constraints, and rank objects by

$$\Delta_{q,a} = F_{q,a}(1) - F_{q,a}(0).$$

The unrestricted optimal assignment supplies the objective value for its selected membership state; an additional solve evaluates the opposite state when feasible. This avoids explicit enumeration of answer sets, although each ILP remains combinatorial in the worst case. The resulting ranking is used for Recall@5 and does not guarantee that the top-five objects form a jointly feasible answer set.

A.6 Training Details

Temporal QA. The direct-SFT models use a four-class sequence-classification head over Llama-3.1-8B or Qwen3-8B. The whole-data split contains 10,777 training and 1,959 validation examples; the reduced-data SFT setting uses 2,155 training examples with the same validation set. We train for three epochs using AdamW with weight decay 0.01, batch size one, and four-step gradient accumulation. The learning rates are 2×10^{-5} for LoRA parameters and 10^{-3} for the classification head, with 5% warm-up followed by cosine decay to 10% of the initial rate. LoRA uses rank 16, scaling factor 32, dropout 0.05, and the `q_proj` and `v_proj` modules. The maximum input length is 256 tokens, the random seed is 13, and checkpoints are selected using validation non-VAGUE micro-F1. Execution + GC models continue from the execution-only initialization with fresh optimizer and scheduler states. The learning rates are 5×10^{-6} for LoRA and 10^{-5} for the classification head, with GC weights of 0.05 for Llama and 0.005 for Qwen.

3D-FORCE: Qwen3-VL baseline. We fine-tune Qwen3-VL-8B using LoRA with rank 16, scaling factor 32, and dropout 0.05, while keeping the vision encoder and multimodal projection frozen. Training uses a learning rate of 10^{-4}, zero weight decay, batch size one, eight-step gradient accumulation, 3% warm-up, cosine scheduling, and gradient clipping at 1. The maximum sequence length is 8,192 tokens.

3D-FORCE: execution and GC training. We freeze the SigLIP2 visual encoder and train the object-property, camera-orientation, and geometric predictors using answer supervision. Training uses Adam with batch size two. The base learning rate is 5×10^{-3}, with 10^{-3} for heading and camera-orientation components and 10^{-4} for position components. GC models continue from an earlier execution-only checkpoint with fresh optimizers and the same component learning rates, using unconstrained execution training as a warm-up stage. The whole-data GC checkpoint then uses one additional pass over the 9,747-question curriculum union with a global-constraint weight of 0.001 and seed 43.

VQAR: execution and GC training. The MLP predictors follow the Scallop-style initialization (Huang et al., 2021) and operate over Mask R-CNN and ResNet-extracted features. We train sequentially across C2–C6 using Adam, batch size one, a constant learning rate of 10^{-5}, and seed 43. The reduced-data regime uses 2,000 training questions per complexity level, totaling 10,000, while the whole-data regime uses 10,000 per level, totaling 50,000. These percentages describe question-training exposure after predictor initialization. The execution-loss weight is one, and the whole-data GC weight is 0.001.

VQAR: rule-source continuations. Rule-source experiments start from the same validation-selected reduced-data execution checkpoint and retain its Adam optimizer state. They use the same training subsets, batch size one, learning rate 3×10^{-6}, and one additional pass over 10,000 questions. The global-constraint weight is 0.001.

B Dataset Statistics

Temporal QA. We use MATRES (Ning et al., 2018). Specifically, we draw training and development examples from TimeBank and AQUAINT and reserve the 837-pair Platinum set for evaluation. The task graph assigns each event pair one of four concepts: BEFORE, AFTER, EQUAL, or VAGUE. It defines 13 global rule templates, including inverse consistency, e.g.,

$$\textsc{Before}(a, b) \iff \textsc{After}(b, a).$$

We also convert TimeBank-Dense to the same four-label setting for separate evaluation where reported.

3D-FORCE. We evaluate two question types from 3D-FORCE (Kamali et al., 2026): REF, which identifies the object satisfying a referring expression, and SAG, which answers a Boolean spatial question. The full training pool contains 9,747 distinct questions, while the reduced-data setting uses a selected 2,000-question subset from the training scenes. Evaluation uses 1,822 usable REF

Table 6: Dataset sizes used in our experiments. The 3D-FORCE training pool is shared across REF and SAG, with a separate 2K subset used for reduced-data experiments. For VQAR, the Total row sums C2–C6 and is not an additional split.

Dataset	**Task or level**	**Training pool**	**Validation**	**Test**
Temporal QA	MATRES TimeBank + AQUAINT	10,777	1,959	837
3D-FORCE	REF	9,747	–	1,822
	SAG	–	–	1,150
VQAR	C2	10,000	1,000	1,000
	C3	10,000	1,000	1,000
	C4	10,000	1,000	1,000
	C5	10,000	1,000	1,000
	C6	10,000	1,000	1,000
	Total	50,000	5,000	5,000

Table 7: Results of intervention on 80 eligible VQAR validation questions identified by screening 500 questions. We reverse one atomic probability that connects to a concept in program execution by constraints but is not included in program execution, while keeping all other probabilities fixed. Answer changes are measured relative to the original prediction under the same connectivity condition. Disconnection removes the relevant constraints. Dashes indicate unreported measurements.

Connecting Constraints	External concept probability	Answer changes	Mean set-F1 (%)	Exact-set correct
Present	Original	Reference	62.99	6/80
Present	Reversed	80/80	59.91	7/80
Removed	Original	Reference	–	–
Removed	Reversed	0/80	–	–

questions and 1,150 SAG questions from disjoint scenes. Each question is provided with a program, images, and object bounding boxes, but training does not use ground-truth object attributes, spatial relations, camera poses, or scene files. The task graph contains object-property concepts, such as $\textsc{Sedan}(a)$, and camera- and object-relative relations, such as $\textsc{Left}_k(a, b)$ and $\textsc{ObjLeft}(a, b)$. In configurations with independently predicted relation scores, an inverse-consistency rule can relate $\textsc{Left}_k(a, b)$ to $\textsc{Right}_k(b, a)$; the geometric predictor already couples these directions by construction. The rules enabled in each experiment are specified with the corresponding training configuration.

KB-VQA. We use VQAR (Huang et al., 2021), which provides complexity levels C2–C6. Each level contains 10,000 training, 1,000 validation, and 1,000 test questions. The task graph represents visual-object concepts, such as $\textsc{Attribute}(o, \texttt{white})$, object–object relations, such as $\textsc{Left}(o_1, o_2)$, and links to symbolic knowledge-base entities. For example, a knowledge-base fact that a dog is a mammal yields the implication $\textsc{Name}(o, \texttt{dog}) \Rightarrow \textsc{SemanticClass}(o, \texttt{mammal})$. The graph also includes opposite-relation constraints, such as $\neg(\textsc{Left}(a, b) \wedge \textsc{Right}(a, b))$, which prevent both relations from holding for the same ordered object pair. The graph manifest contains 1,199 base concepts and 1,652 declared rule instances; these counts describe the task-level graph and should not be interpreted as per-question counts of active grounded rules.

C Additional Experiments

C.1 Constraint Influence Beyond Program Execution

Experiment Setup. To test whether global constraints can impose restrictions external to the program on the concept values used during inference, we perform a controlled intervention on VQAR. We screen 500 questions and identify 80 eligible cases in which a concept outside the program execution path is connected by a global constraint to a concept used by the program. For each case, we change the probability of one concept not directly used by the program but connected through a

global constraint to a concept used during execution. Other probabilities are fixed. If p_j denotes its original predicted probability, we reverse it as $p'_j = 1 - p_j$. We then compare ILP outputs with the connecting rule present and with its rule template removed. Answer changes are measured relative to the original probability under the same connectivity condition. Removing the rule changes the corresponding connectivity in the ILP feasible set.

Experiment Results. As shown in Table 7, the answer set changes in all 80 cases when the connecting constraint is present, while it remains unchanged when the constraint is removed. This demonstrates that, when grounded in G_x, global constraints can impose restrictions external to the program on the concept values used during execution, allowing external concepts to affect the final answer. Because the experiment selects answer-sensitive cases with eligible external connections, it establishes that such influence is possible but does not estimate its prevalence or average benefit on held-out data.

D PROMPT SPECIFICATIONS

Conceptual graph construction. We provide the LLM with a domain description, concepts and relations, and a few in-context examples to obtain the conceptual graph G.

```
Construct a reusable conceptual graph for the supplied domain.
Use the domain description, concepts and relations, and
in-context examples to define entity types and their properties.

Specify each relation's argument types and argument order.
Distinguish binary properties from categorical properties,
and observed inputs from concepts requiring neural prediction.

Describe how input entities instantiate the graph. For visual
inputs, associate objects with bounding boxes and visual
features. Specify the inputs required by relation predictors.

Return DomiKnowS graph declarations and explicit mappings
from input entities and neural outputs to graph concepts.
Use only the supplied graph interfaces. Do not include
example-specific predictions, gold answers, or unsupported
domain assumptions. List ambiguities requiring clarification.
```

VQAR relation-rule proposal. The rule generator supplies the following system instructions, followed by the supported first-order-logic forms:

```
Propose reusable group-level domain rules, not question answers.
Use only the supplied predicate/operator allowlists. Treat
allowed_variable_concepts and allowed_predicate_concepts in the
user object as exhaustive: every variable concept and every
predicate in the output must use one of those exact names.
Never invent a concept, emit a dataset alias, or copy a relation
from the format examples. Never refer to example-specific ids
or gold answers. Propose distinct, non-redundant candidate rules
where the vocabulary supports them; do not pad the response.
Each candidate must be independently verifiable. Express every
candidate as explicit first-order logic. A deterministic parser
converts the restricted FOL to DomiKnowS, and a deterministic
verifier, not you, decides acceptance.
Return JSON only, matching the response schema exactly.
```

The allowed forms are same-pair exclusion, implication, XOR, equivalence, and cardinality over two predicates, together with a reversed-pair implication:

```
forall p:object_pair. not (A(p) and B(p))
```

```
forall p:object_pair. A(p) -> B(p)
forall p:object_pair. A(p) xor B(p)
forall p:object_pair. A(p) <-> B(p)
forall p:object_pair. at_least 1 of (A(p), B(p))
forall p:object_pair. at_most 1 of (A(p), B(p))
forall p:object_pair. exactly 1 of (A(p), B(p))
forall p,q:object_pair. reverse(p,q) and A(p) -> B(q)
```

Here, A and B must be exact names in the supplied predicate inventory. The user message is a JSON object containing the domain, group key, semantic-family description, permitted relation names, typed concept signatures, and group rationale. The response must contain `schema_version`, `domain`, `group_key`, `fol_rules`, and `group_rationale`; each proposed rule contains an identifier, FOL expression, and rationale. An empty `fol_rules` array is permitted.

VQAR unary-rule proposal. For object-property rules, the system prompt restricts the model to one quantified object variable and the supplied unary predicates:

```
Propose reusable GraphQA unary global-consistency rules,
never question answers. Use exactly one quantified object
variable named o with type obj and only exact predicate names
from allowed_predicate_concepts. Return zero to eight distinct
rules. Abstain with an empty fol_rules array if no universally
valid rule is defensible. Do not infer exclusivity from missing
or rare co-occurrence: missing VQAR labels are unknown, not
false. A deterministic parser and DomiKnowS compiler will
check each candidate independently. Return strict JSON only.
```

The permitted forms mirror the same-pair forms above with `forall o:obj` and unary predicates $A(o)$ and $B(o)$. The user message supplies the semantic group, exhaustive predicate names and signatures, and training-derived evidence context.

Rule-revision call. When deterministic verification rejects a proposal, the generator retains the original system and user messages and appends a user message containing structured failure feedback:

```
The previous proposal failed deterministic verification.
A deterministic verifier, not you, decides acceptance.
Structured failure feedback -- gate, code, offending_node_path,
and message identify exactly what was wrong:
<STRUCTURED_FAILURE_JSON>

Use the same restricted-FOL response contract as the initial
request. Return the complete corrected RuleBatch JSON object,
not a diff, patch, AST object, markdown, or prose. Replace only
failed candidates; do not repeat an unsupported formula. If no
supported alternative is defensible, return an empty fol_rules
array.
```

3D-FORCE Qwen3-VL baseline. For each question, the baseline receives the camera images, candidate boxes in image 1, and one of the following user-message templates. For REF:

```
<images>
Image 1 is camera 0, image 2 is camera 1, ...
The objects in image 1 are: <candidate bbox_2d list>.
Locate <referring expression> in image 1,
output its bbox coordinates using JSON format.
```

For SAG:

```
<images>
```

```
Image 1 is camera 0, image 2 is camera 1, ...
The objects in image 1 are: <candidate bbox_2d list>.
Question: <original question> Answer yes or no.
```

The image placeholders are replaced by the corresponding images; candidate coordinates are normalized to the model's 0–1000 bounding-box scale.

Temporal QA answer prompt. For the causal-language-model relation scorer, the basic prompt is:

```
Classify the temporal relation from event E1 to event E2.
Choose exactly one label from: Before, After, Equal, Vague.
Text: <text with [E1]...[/E1] and [E2]...[/E2] markers>
Answer:
```