Explainable Zero-Shot Visual Question Answering via Logic-Based Reasoning—Extended Abstract

Thomas Eiter[1], Jan Hadl[1], Nelson Higuera[1], Lukas Lange[2], Johannes Oetsch[3], Bileam Scheuvens[4] and Jannik Strötgen[5]

[1]TU Wien, Austria
[2]Bosch Center for Artificial Intelligence, Renningen, Germany
[3]Jönköping University, Sweden
[4]University of Tübingen, Germany
[5]Karlsruhe University of Applied Sciences, Germany

Abstract

This extended abstract presents GS-VQA, a neurosymbolic system for zero-shot Visual Question Answering (VQA). GS-VQA constructs symbolic, question-conditioned scene graphs from real-world images using zero-shot vision models guided by large language models. These graphs are effectively knowledge graphs that can be used for logic-based inference using Answer-Set Programming (ASP). The system enables question answering via symbolic inference and can generate logical explanation traces using `xclingo`. Evaluations on the GQA benchmark demonstrate the method's transparency and diagnostic power despite modest accuracy in comparison to state of the art neural systems.[1]

Overview. *Grounded-Scene Visual Question Answering* (GS-VQA) [1] is a zero-shot, modular neurosymbolic system for Visual Question Answering (VQA) [2] tasks, instantiated for the challenging GQA [3] dataset. Unlike pure neural models, it decouples the perception, language understanding, and reasoning tasks into dedicated modules, enhancing transparency and explainability. Particularly, we leverage Answer Set Programming (ASP) [4] as the backbone of our reasoning module. ASP is a rule-based approach for declarative problem solving with roots in knowledge-representation and reasoning. Problems are represented by rules in the ASP modelling language such that the solutions correspond to the models found by an ASP solver. As illustrated in Figure 1, the system comprises three main components:

- **Language module**: GPT-4o [5], a large language model, parses the natural language question and extracts a structured representation of it.
- **Vision module**: OWL-ViT [6], a zero-shot vision model, is guided by the extracted question representation to detect only the relevant object types in the image. Each detected object is then classified via CLIP [7], and relations are inferred through spatial heuristics and text-based similarity. Objects and their relations form a knowledge graph describing the aspects of the scene that are relevant to answer the question.
- **Reasoning module**: The symbolic query and scene graph are translated into ASP facts and rules. These, together with a fixed ASP theory encoding generic reasoning procedures, are used to yield answers and causal explanations.

Question-conditioned scene graph materialisation. A major limitation of standard object detectors is the generation of scene graphs that do not include irrelevant or spurious entities. To mitigate

[1]This work is based on a full article accepted at NeSy 2025, supported by the Bosch Center for Artificial Intelligence. Code is available at https://github.com/pudumagico/nesy25

The Second Workshop on Knowledge Graphs and Neurosymbolic AI (KG-NeSy), co-located with SEMANTiCS'25: International Conference on Semantic Systems, September 3–5, 2025, Vienna, Austria

thomas.eiter@tuwien.ac.at (T. Eiter); jan.hadl@tuwien.ac.at (J. Hadl); nelson.ruiz@tuwien.ac.at (N. Higuera); lukas.lange@de.bosch.com (L. Lange); johannes.oetsch@ju.se (J. Oetsch); bileam.scheuvens@student.uni-tuebingen.de (B. Scheuvens); jannik.stroetgen@h-ka.de (J. Strötgen)

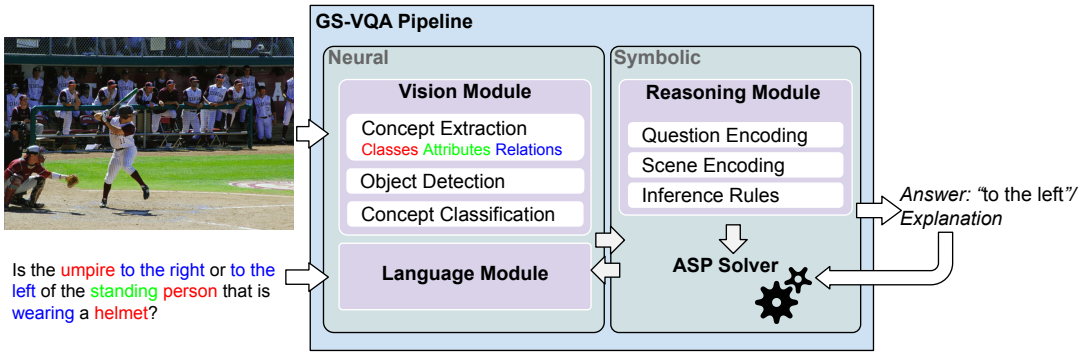

Figure 1: Overview of our GS-VQA pipeline showing the neural and symbolic modules of the pipeline.

this, we condition the extraction process on the input tuple (I, Q), where I is the image and Q an associated question. The language module uses an LLM on Q to extract three sets: $\mathcal{C}$, a list of candidate object classes; $\mathcal{A}$, relevant attributes; and $\mathcal{R}$, spatial or semantic relations. These sets constrain the visual module and are used as inputs alongside I to the following processes:

1. **Category-constrained detection**: OWL-ViT is prompted only with labels from $\mathcal{C}$, increasing precision and suppressing irrelevant detections.
2. **Attribute tagging**: For each detected object, attributes in $\mathcal{A}$ are predicted using CLIP-based template matching (e.g., "a <COLOR> <OBJECT>").
3. **Relation extraction**: Pairs of objects are evaluated for relations in $\mathcal{R}$, using spatial heuristics or CLIP-based text similarity on relation templates (e.g., "<OBJECT1> left of the <OBJECT2>").

The resulting symbolic scene graph includes only entities and relations, which are required to answer the specific question, forming a task-specific knowledge graph tailored to (I, Q).

Reasoning and explainability over scene graphs. To answer the question *"What is the person in front of?"*, GS-VQA first parses it into a symbolic query program using the language module. This program is encoded in ASP as:

```
end(3). query(3, 2, class). relate(2, 1, _, in_front_of, subject).
select(1, 0, person). scene(0).
```

The scene graph is similarly encoded as a set of ASP facts describing objects, their attributes, and spatial relations. The relevant symbolic context for this question may look like:

```
object(o0). has_attr(o0, class, glove).    object(o1). has_attr(o1, class, person).
object(o12). has_attr(o12, class, sky).
has_rel(in_front_of, o1, o0).  has_rel(in_front_of, o1, o12).
```

The theory—a fixed ASP program—specifies the semantics of the facts that represent questions as rules, which are used to derive answers from scene graphs and query programs. Rules from the ASP theory look like:

```
state(TO,ID) :- scene(TO), object(ID).
state(TO,ID) :- select(TO, TI, CLASS), state(TI, ID), has_attr(ID, class, CLASS).
```

We use `clingo` [8] as the ASP solver to compute stable models representing candidate answers. In the example shown, two alternative answers are obtained: one selecting `o0` (glove), and another selecting `o12` (sky). Only the latter is the expected answer, but both are entailed due to ambiguities in the scene graph representation.

To analyse and revise such cases, we use `xclingo`[9] to generate human-readable traces of the reasoning steps. This requires to annotate each of the rules in the theory with appropriate natural language descriptions. An example of the rules in the annotated theory is:

"What is the person in front of?"

```
|__The answer to the question is glove
|   |__We detect that object with ID o0 has class with value glove
|   |   |__  The object with ID o0 is the only object detected of
|   |   |   | this kind
|   |   |   |__  Object with ID o1 is in_front_of object with ID
|   |   |   |   | o0
|   |   |   |   |__  We select object o1 because of being in the
|   |   |   |   |   | class person
|   |   |   |   |   |__We locate a candidate from the detected objects,
                        object with ID o1

|__The answer to the question is sky
|   |__We detect that object with ID o12 has class with value sky
|   |   |__  The object with ID o12 is the only object detected of
|   |   |   | this kind
|   |   |   |__  Object with ID o1 is in_front_of object with ID
|   |   |   |   | o12
|   |   |   |   |__  We select object o1 because of being in the
|   |   |   |   |   | class person
|   |   |   |   |   |__We locate a candidate from the detected objects,
                        object with ID  o1
```

Figure 2: Derivation trees for the incorrect answer "glove" (top) and corrected answer "sky" (bottom), obtained using `xclingo` to explain and revise symbolic inference.

```
%!trace_rule {"The answer to the question is %", V}
ans(V) :- end(TO), attr_value(TO,V).
%!trace_rule {"Object with ID % is % object with ID %", ID', REL, ID}
state(TO, ID') :- relate(TO, TI, CLASS, REL, subject), state(TI, ID),
                  has_attr(ID', class, CLASS), has_rel(ID', REL, ID).
%!trace_rule {"We detect that object with ID % has % with value %", ID, ATTR, VALUE}
attr_value(TO,VALUE) :- query(TO, TI, ATTR), state(TI, ID),
                        has_attr(ID, ATTR, VALUE).
```

The top derivation in Figure 2 explains why the system initially selected "glove", following the reasoning through detected attributes and spatial relations. The bottom trace is obtained by appending a constraint enforcing that the correct object is `sky`, enforcing the ground-truth. This showcases how symbolic constraints can be used not only for explanation but also potentially for correction.

Evaluation summary. We evaluate GS-VQA on 500 GQA questions and images. Our system achieves an overall accuracy of 36.2%, with high correctness in question parsing using LLMs (84.4%). Other zero-shot systems such as ViperGPT [10] report higher accuracy (48.1%), but rely on non-modular pipelines and general programming languages as symbolic executors. These approaches lack the formal semantics and transparent derivation traces inherent to declarative logic. As stronger LLMs and VLMs become available, we expect gains in both perception and question interpretation, helping to close the performance gap with existing systems

Declaration on Generative AI

During the preparation of this work, the author(s) used `GPT-4o` in order to: Grammar and spelling check. After using this tool, the author(s) reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] T. Eiter, J. Hadl, N. Higuera, L. Lange, J. Oetsch, B. Scheuvens, J. Strötgen, Explainable zero-shot visual question answering via logic-based reasoning, in: Proceedings of the 19th International Conference on Neural-Symbolic Learning and Reasoning (NeSy 2025, Santa Cruz, California (USA), September 9-10, 2025, Lecture Notes in computer Science, Springer, 2025. To appear.

[2] A. Agrawal, J. Lu, S. Antol, M. Mitchell, C. L. Zitnick, D. Parikh, D. Batra, VQA: visual question answering - www.visualqa.org, Int. J. Comput. Vis. 123 (2017) 4–31. doi:10.1007/s11263-016-0966-6.

[3] D. A. Hudson, C. D. Manning, GQA: A New Dataset for Real-World Visual Reasoning and Compositional Question Answering, in: Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2019), Computer Vision Foundation / IEEE, 2019, pp. 6700–6709. doi:10.1109/CVPR.2019.00686.

[4] G. Brewka, T. Eiter, M. Truszczynski, Answer set programming at a glance, Communications of the ACM 54 (2011) 92–103. URL: https://doi.org/10.1145/2043174.2043195. doi:10.1145/2043174.2043195.

[5] OpenAI, GPT-4 technical report, CoRR abs/2303.08774 (2023). URL: https://doi.org/10.48550/arXiv.2303.08774. doi:10.48550/ARXIV.2303.08774. arXiv:2303.08774.

[6] M. Minderer, A. A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen, X. Wang, X. Zhai, T. Kipf, N. Houlsby, Simple open-vocabulary object detection, in: S. Avidan, G. J. Brostow, M. Cissé, G. M. Farinella, T. Hassner (Eds.), Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part X, volume 13670 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 728–755. URL: https://doi.org/10.1007/978-3-031-20080-9_42. doi:10.1007/978-3-031-20080-9_42.

[7] M. Li, R. Xu, S. Wang, L. Zhou, X. Lin, C. Zhu, M. Zeng, H. Ji, S. Chang, Clip-event: Connecting text and images with event structures, in: Conference on Computer Vision and Pattern Recognition (CVPR 2022), IEEE, 2022, pp. 16399–16408. doi:10.1109/CVPR52688.2022.01593.

[8] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Multi-shot asp solving with clingo, Theory and Practice of Logic Programming 19 (2019) 27–82.

[9] P. Cabalar, J. Fandinno, B. Muñiz, A system for explainable answer set programming, in: Technical Communications of the 36th International Conference on Logic Programming (ICLP 2020), volume 325 of *EPTCS*, 2020, pp. 124–136.

[10] D. Surís, S. Menon, C. Vondrick, ViperGPT: Visual inference via python execution for reasoning, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV 2023), IEEE, 2023, pp. 11854–11864. doi:10.1109/ICCV51070.2023.01092.