TPLP: Page 1–35. © The Author(s), 2026. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.

doi:10.1017/S1471068426100684

Distilling Answer-Set Programming Rules from LLMs for Neurosymbolic Visual Question Answering

NELSON HIGUERA RUIZ
Technische Universität Wien, Austria
(*e-mail:* nelson.ruiz@tuwien.ac.at)

JOHANNES OETSCH
Jönköping University, Sweden
(*e-mail:* johannes.oetsch@gmail.com)

THOMAS EITER
Institute of Logic and Computation, Technische Universität Wien, Austria
(*e-mail:* eiter@kr.tuwien.ac.at)

submitted 8 July 2025 UTC; revised 2 February 2026 UTC; accepted 25 June 2026 UTC

Abstract

Visual Question Answering (VQA) is the task of answering questions about images, requiring the integration of multimodal input and reasoning. Modular approaches that incorporate logic-based representations into the reasoning component offer clear advantages over end-to-end trained systems, particularly in terms of interpretability. However, adapting or extending these representations when task requirements change can place a significant burden on developers. To address this challenge, we present an approach for distilling rules from Large Language Models (LLMs). Our method prompts an LLM to extend an initial VQA reasoning theory, expressed as an answer-set program, to meet new requirements of the task. Examples from VQA datasets guide the LLM, validate the results, and help correct erroneous rules by leveraging feedback from the ASP solver. We demonstrate that our approach is effective across diverse VQA datasets. Notably, only a few examples are needed to elicit correct rules from LLMs. Our experiments suggest that rule distillation from LLMs is a promising alternative to traditional data-driven rule learning approaches. Under consideration in Theory and Practice of Logic Programming.[1]

Keywords: declarative knowledge distillation, answer-set programming, visual question answering, LLMs

[1] The code for reproducing our experiments is available as an online repository: https://github.com/pudumagico/KDASP.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

1 Introduction

Visual Question Answering (VQA) (Antol *et al.* 2015; Goyal *et al.* 2017) is a challenging problem with important applications (Barra *et al.* 2021; Lin *et al.* 2023; Ding *et al.* 2025); it requests to provide an accurate answer for a question about a visual scene. This requires not just a joint understanding of vision and text, but also to follow complex chains of reasoning operations.

Neurosymbolic approaches to VQA (Yi *et al.* 2018; Mao *et al.* 2019; Amizadeh *et al.* 2020; Eiter *et al.* 2022; Surís *et al.* 2023; Johnston *et al.* 2023, etc). use deep learning for perception, produce a symbolic representation of the input image and question, and then perform reasoning on this representation in a purely symbolic way. These approaches are interpretable, transparent, and can be extended due to their compositional structure. A promising direction in this regard is to use logic-based formalisms for the reasoning component. We are in particular interested in using Answer-Set Programming (ASP) (Brewka *et al.* 2011; Lifschitz 2019), a prominent knowledge representation formalism, to realize the reasoning module of such systems. In addition to concise representations, advantages are that non-determinism allows for multiple answers if desired, ambiguity in perception often can be resolved by reasoning (Eiter *et al.* 2022), and one can more easily add explanation capabilities (Eiter *et al.* 2023, 2025). ASP for augmenting VQA with reasoning capabilities is in fact gaining traction (Abraham *et al.* 2024; Basu *et al.* 2020; Eiter *et al.* 2022, 2023; Chowdhury *et al*. 2025), while outside of VQA it is used for visual tasks such as segmentation of medical images (Bruno *et al.* 2021) and compliance checking for control panels (Barbara *et al.* 2023); see the recent survey by Borroto *et al.* (2025).

The downside is that maintaining or extending ASP representations when requirements change, for example when there is a change in the types of question that needs to be answered, can be an additional burden on the developer. We address this issue by presenting an approach for *declarative knowledge distillation* from Large Language Models (LLMs) (Vaswani *et al.* 2017; Zhao *et al.* 2026). The premise of this work is that we have a neurosymbolic VQA system in place whose reasoning component involves an initial ASP theory. If there is a shift in requirements, reflected by new examples for which the system does not yet give the correct answer, our method is to prompt an LLM to extend this initial theory to meet the new requirement. Hence, examples from a VQA dataset are used to guide the LLM in generating new rules and validate the results, which involves using multiple prompting strategies, mending the generated rules if they are not correct using feedback from the ASP solver, and regressively testing past examples to ensure consistency.

Statistical-relational learning (SRL) (Raedt and Kersting 2017) aims to learn rules that generalize from dataset examples. A notable subfield of SRL is Inductive Logic Programming (ILP) (Cropper *et al.* 2022), where systems rely on a hard search space – explicitly defined by the developer through mode declarations or language bias. In contrast, LLMs operate within a soft search space implicitly encoded via natural language prompts. Our method leverages this capability by using examples not as training data, but solely to guide a knowledge distillation process that conditions the LLM toward suitable rule completions (see Section 6 for further discussion). Notably, our experiments show that only a few examples are sufficient to elicit correct rules from the LLM.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

This distinguishes our approach from data-driven methods, which typically require logical elaboration or substantially more training data.

We use a diverse selection of VQA datasets to evaluate our knowledge distillation method: GQA (Hudson and Manning 2019), CLEVR (Johnson *et al.* 2017), and CLEGR^V (Bauer *et al.* 2023, 2024). GQA uses real images that depict complex visual scenes and questions with a large number of possible answers that involve processing objects, their attributes, and relations between them. CLEVR uses synthetic scenes of geometric objects and is designed to challenge VQA systems with compositional questions that require breaking down the overall task into an evaluation graph of primitive operations. CLEGR^V is a recently published dataset that uses images of graphs that resemble transit networks and requires graph-based reasoning. To test our approach, we adopt the representations from modular VQA systems with ASP for reasoning that can solve these datasets (Eiter *et al.* 2023; Hadl 2023; Bauer *et al.* 2023, 2024).

Our main contributions are briefly summarized as follows:

- We propose a method for *declarative knowledge distillation* from LLMs, enabling the automated extension of ASP-based reasoning modules in VQA systems, including the ability to compose new ASP rules from existing ones.
- We design prompting strategies for the distillation method that includes multi-prompting, chain-of-thought, and rule mending.
- We experimentally evaluate our method across three diverse VQA datasets – GQA, CLEVR, and CLEGR^V. Our evaluation includes also ablation studies involving the proposed prompting strategies.
- We introduce a redundancy elimination heuristic that attempts to prune the distilled rule sets without reducing accuracy.

Our experiments show that LLMs can grasp and produce ASP rules to implement new reasoning operators, and thus can effectively assist in the process of extending a reasoning component. Ablation studies reveal that the effectiveness of prompting strategies depends on both the model and the dataset: while stronger models benefit significantly from their use, smaller models may be hindered by certain strategies. This methodology offers a promising avenue for extracting domain-specific knowledge from LLMs in the realm of VQA and offers a viable complement to data-driven rule learning approaches.

2 Background

The logic-based VQA approaches for the VQA datasets that we are going to use for our evaluation use a modular architecture and ASP to derive answers from a symbolic scene and question representation. Before we provide background on these VQA approaches, we review the basics of ASP.

2.1 Answer-set programming

Answer-Set Programming (ASP) (Brewka *et al.* 2011; Lifschitz 2019) is a well-known approach to declarative problem solving, in which solutions to a problem are described by programs that consist of sets of logical rules. Efficient ASP solvers for evaluating programs are available such as clingo Gebser *et al.* (2019), dlv Leone *et al.* (2006), or wasp Alviano *et al.* (2013).

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use , available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

For our concerns, an ASP program is a finite set P of rules r of the form

$$a \coloneq b_1, \ldots, b_n, \ not\ c_1, \ldots, \ not\ c_n \quad m, n \geq 0, \tag{1}$$

where a (the `head`), all b_i, and all c_j (the `body`) are atoms in a first-order predicate language, and *not* stands for negation as failure (weak negation). We allow that a may be missing (viewed as falsity); then r acts as a constraint. Intuitively, the rule means that whenever all b_i are true and none of the c_j can be shown to be true, then a must be true. A *fact* is a rule with empty body (in which case $\coloneq$ is omitted). Facts represent knowledge that is unconditionally true.

The semantics of a ground (variable-free) ASP program is given by answer sets, which are Herbrand models that satisfy a stability condition (Gelfond and Lifschitz 1988). A Herbrand interpretation of P is a set I of ground atoms in the language of P (intuitively, the atoms that are true); it is a model of P if for each rule $r \in P$ either

(i) $a \in I$ or
(ii) $\{b_1, \ldots, b_n\} \not\subseteq I$ or
(iii) $I \cap \{c_1, \ldots, c_n\} \neq \emptyset$; that is I satisfies r viewed as implication in classical logic.

Furthermore, I is an answer set of P (Faber *et al.* 2004)[2] if I is a $\subseteq$-minimal model of the program

$$P^I = \{r \in P \mid I \text{ satisfies neither } (ii) \text{ nor } (iii)\}.$$

Intuitively, I must result by applying the rules r whose bodies "fire" w.r.t. I starting from facts. The semantics of programs with variables is defined in terms of their groundings (uniform replacement of variables in rules with all possible ground terms).

ASP provides further constructs that extend its modeling capabilities: *choice rules* allow for selecting a subset of atoms under cardinality bounds. They are of the form

$$l\ \{a_1, \ldots, a_k\}\ u \coloneq \texttt{body}$$

where l, u are integers specifying the minimum and maximum number of atoms a_i that may be true if the `body` holds. This supports controlled nondeterminism and is useful for selection problems.

Aggregates enable reasoning over sets of terms, for instance, to impose conditions on their cardinality or sums. Here is an example of a rule that involves an aggregate expression for counting:

$$p(X) \coloneq \#\texttt{count}\{Y : q(X, Y)\} > N.$$

This example states that $p(X)$ holds if the number of Y such that $q(X, Y)$ holds exceeds N. ASP supports various aggregates, including `#sum`, `#min`, and `#max`.

While a hard constraint is a constraint that must be satisfied by any feasible solution to the ASP program, a *weak constraints* (soft constraint) can be violated, but violating it incurs a penalty in the objective function. A weak constraint has the following form:

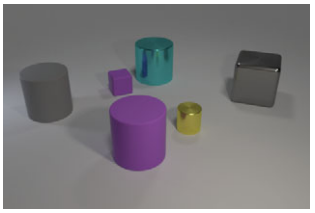
$$:\sim \texttt{body}.\ [w@l]$$

[2] This definition is equivalent to the one of Gelfond and Lifschitz (1988) for programs consisting of rules (1) and ensures benign semantics of recursive aggregates (Faber *et al.* 2004).

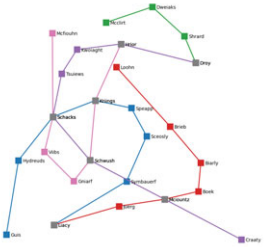
Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684



Is the umpire to the right or to the left of the standing person that is wearing a helmet?
Answer: To the left.
(a)



Is there an object of the same color as the small cube?
Answer: Yes.
(b)



How many stations are between Loohn and Boek?
Answer: 2.
(c)

Fig. 1. Example scenes and questions from GQA (a), CLEVR (b), and CLEGR^V (c).

where w is the penalty cost and l the priority level. ASP solvers seek answer sets that minimize the total penalty incurred by the weak constraints whose body is satisfied. Ties are broken using the priority level. For more details on ASP, we refer to (Brewka *et al.* 2011; Lifschitz 2019; Calimeri *et al.* 2020).

2.2 Modular VQA with answer-set programming

We consider three VQA datasets as application area for our rule distillation approach: GQA (Hudson and Manning 2019), CLEVR (Johnson *et al.* 2017), and CLEGR^V (Bauer *et al.* 2023, 2024). Example instances of these datasets are shown in Figure 1. They involve realistic scenes in GQA, synthetic scenes in CLEVR, and images of labeled graphs in CLEGR^V.

For our distillation process, we adopt the logical representations and encodings of neurosymbolic systems that use ASP for reasoning, and instantiate a generic modular neurosymbolic architecture, shown in Figure 2 that has the following core components:

- **Vision module:** Neural networks that process the visual scene to produce a symbolic representation of the scene, called the *scene graph*. It contains a structured representation of each object in the image, with their corresponding attributes and relationships.
- **Language module:** Neural networks that process the natural language question to produce a symbolic representation of the question, called a *functional program*. This is a tree-like structure, where every node defines a *primitive function*, and edges between the nodes represent the flow of information from the leaves to the root.
- **Reasoning module:** A component that uses an ASP solver to derive the answer. First the scene graph and functional program are translated into sets of ASP facts. Details of these representations depend on the dataset, but they have in common that each primitive function of the functional program is represented by a single ASP fact in the question representation. We use an ASP program, which we also refer to as ASP theory, to encode how to derive an answer based on scene and

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

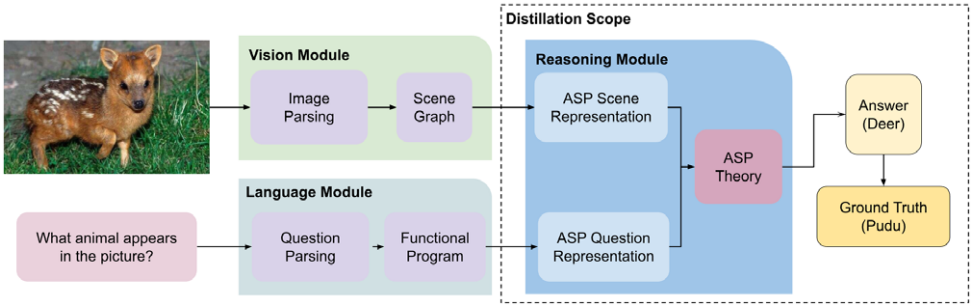


Fig. 2. Generic Neurosymbolic VQA pipeline with three main components: (i) vision module, (ii) language module and (iii) reasoning module. Our distillation method operates on the symbolic representations of scenes and questions, ASP theory, as well as the ground-truth answers to the questions.

question encodings. An ASP solver computes then an answer set from which the answer to the question is extracted.

The ASP theory is a set of rules that – in contrast to the question and scene encoding – does not change from question to question. The theory is thus *general* and encodes the semantics of the primitive functions in a *uniform* way. Using ASP as the symbolic formalism for reasoning provides us not only with a mature ecosystem of tooling and solvers but, and notably, allows one to capture the uncertainty in the predictions of the image-processing component.

The collection of primitive functions the ASP theory implements depends on the task. For GQA, the focus is on reasoning on a complex scene graph, CLEVR focuses on processing compositional questions, and CLEGR^V involves multi-modal reasoning tasks on labeled graphs, for example deciding reachability between nodes with certain labels. Also, the details of how the vision and language module are realized varies from one application to another. We next review some of these details and the characteristics of the datasets used. For each dataset, we present examples of a scene and question as ASP facts, and the complete ASP theory in Appendix A.

2.2.1 The GQA dataset

We use the state-of-the-art GQA dataset, which has been widely adopted recently (Amizadeh *et al.* 2020; Surís *et al.* 2023; Liang *et al.* 2020; Li *et al.* 2023). It contains over 22 M open and binary questions that are complex in structure, involve a wide variety of reasoning skills, and have a large number (1 878) of possible answers.

The questions cover more than 100 000 images from the Visual Genome dataset (Krishna *et al.* 2017) that present real-world scenes with a wide variety of object classes, attributes, and relations. GQA provides both a functional representation of each natural-language question and a Visual Genome scene graph for every image, which allows us to construct ASP programs that always infer the correct answer whenever there is perfect language and visual information. We later will erase parts of these programs to simulate incomplete correct programs and ask an LLM to extend them accordingly.

We adopt the representations and theory of a recent system for GQA (Hadl 2023; Eiter *et al.* 2025) that adheres to the general architecture described above.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

For the vision module, the system uses *"question-driven" partial scene-graph extraction*, where only information is extracted from the scene that is relevant for answering the question at hand. It does so by relying on the foundation models CLIP (Radford *et al.* 2021) and OWL-ViT (Minderer *et al.* 2022) for processing the visual scene. They do not need any training and thus solve the task in a zero-shot manner. A snippet of ASP facts from an ASP scene representation is:

```
object(o1).
has_attr(o1, class, alive).
has_attr(o1, class, person).
has_attr(o1, class, baseball_player).
has_attr(o1, name,  baseball_player).
```

These facts describe parts of a scene that contains an object whose most specific class is "baseball_player" (cf. Figure 1).

Questions are translated from their functional program into ASP facts. For example, the question "Is the umpire to the right or to the left of the standing person that is wearing a helmet?" may be rendered as follows:

```
scene(0).
select(1, 0, helmet).
relate(2, 1, person, wearing, subject).
filter_any(3, 2, standing).
choose_rel(4, 3, umpire, to_the_left_of, to_the_right_of, subject).
```

The numbers encode steps of an evaluation plan for deriving the answer, where the first argument is the current step, followed by numerical input steps providing input and/or additional arguments. All of our ASP question encodings adopt variations of this representation. In comparison with the original functional program, each of these facts represents a primitive function, while numbers, in them represent states, denoting the flow of computation from state to state, serving virtually as edges in a graph where the nodes are the primitives.

The ASP theory encodes the semantics of the primitive functions that can appear in the question encoding. The whole theory has 60 rules; some rules are:

```
state(TO,ID)    :-  scene(TO), object(ID).
state(TO,ID)    :-  select(TO,TI,C), state(TI,ID), has_attr(ID,_, C).
state(TO,ID)    :-  filter_any(TO, TI, VALUE), state(TI, ID),
                    has_attr(ID, ATTR, VALUE).
```

In these rules, `TI` and `TO` are variables representing input/output step references, `ID` represents an object id, and `C` an object class. These rules define how the state of candidate objects is updated based on the scene, selection criteria, and attribute filters. For example, the `select` rule propagates objects from a previous step `TI` to the current step `TO` if they have the specified class `C`.

The entire theory is solved alongside both scene and question encodings to produce an answer.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

2.2.2 The CLEVR dataset

The second dataset we consider is CLEVR (Johnson *et al.* 2017), which uses synthetic scenes but challenges VQA systems with long compositional questions. The ASP-based VQA system used for CLEVR has been extended by an explanation component (Eiter *et al.* 2022, 2023). The dataset contains synthetically generated images with different objects. The latter vary in their shape, color, size, and material. CLEVR questions require, for example identifying objects, counting, filtering for attributes, comparing attributes, and spatial reasoning. They are formulated in natural language, but, as for GQA, a functional representation is also provided that can be directly parsed into ASP facts.

The architecture of the system for CLEVR from where we take the ASP theory is similar to the one for solving GQA. However, the scenes are less complex as geometric objects are clearly distinguishable, using the popular object-detection framework YOLOv5[3] to identify all objects in the image, their attributes, and bounding-box information.

A snippet of ASP facts from a scene representation, question representation, and theory are:

```
obj(0,129,121,rubber,green,cube,small).
obj(1,131,202,metal,blue,cylinder,small).
obj(2,247,89,rubber,green,cube,small).
```

These facts describe objects and their different attributes, such as ID, center of the bounding box X, Y positions, material, etc. The question "Is there a small green cube in the scene?" is rendered as follows:

```
scene(0).
filter_small(1).
filter_green(2).
filter_cube(3).
unique(4).
exists(5).
end(6).
```

In a similar fashion to GQA, the numbers encode evaluation steps. The ASP theory consists of 72 rules; some of the rules are:

```
object(ID) :- obj(ID,_,_,_,_,_,_).
has_size(ID,SIZE) :- obj(ID,_,_,_,_,_,SIZE).
has_color(ID,COLOR) :- obj(ID,_,_,_,COLOR,_,_).

state(T+1,ID) :- scene(T), object(ID).
state(T+1,ID) :- filter_small(T), state(T,ID), has_size(ID,small).
state(T+1,ID) :- filter_green(T), state(T,ID), has_color(ID,green).
state(T+1,ID) :- unique(T), state(T,ID).
:- unique(T), state(T,ID), state(T,ID'), ID!=ID'.
```

[3] ultralytics.com/yolov5.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

Here, the `unique` rule propagates exactly one object from step `T` to `T+1`, enforced by a constraint that eliminates any answer set where multiple objects are present at step `T`. The rules follow essentially the same pattern as those for GQA, evaluating some predicate in a state and then passing this information to the following state.

2.2.3 The CLEGR^V dataset

The recently introduced CLEGR^V dataset (Bauer *et al.* 2023, 2024) consists of images of graphs that resemble transit networks; the task is to answer questions concerning such graphs. An illustration of a graph and a question that requires to find the shortest path between two stations and count the number of stations that are on that path is shown in Figure 1.

We adopt the ASP theory of the modular neurosymbolic architecture from the same publication. The vision module is multi-modal in the sense that it deals with graph and text information: it combines *optical graph recognition* (Auer *et al.* 2013) for graph parsing, a tool that parses the graph image into an abstract representation that contains nodes and edges, and a pretrained *optical character recognition* neural network (Sabu and Das 2018) to parse the image into text labels that are associated with the closest node. The language module parses the natural language questions using either regular expressions or LLMs. Finally, the ASP theory encodes the semantics of the question as a logic program.

A snippet of ASP facts from a scene representation, question representation and theory are:

```
node(thraonk,brown).
node(mcciourly,orange)
edge(shuosh,krauess,olive).
edge(krauess,sairr,olive).
line(brown).
line(olive).
```

These facts describe the different nodes, edges and the "metro lines" the nodes and edges belong to, for example the node "Thraonk" is in the brown line. The question "How many stations are in the shortest path between Thraonk and McCiourly?" is encoded as follows:

```
station(0,thraonk).
station(0,mcciourly).
shortestPath(1).
countNodesBetween(2).
end(3).
```

Again, the structure resembles the same evaluation procedure of GQA and CLEVR. This theory consists of 56 rules; some of the rules are:

```
sp((N, N')) :- shortestPath(T), station(T-1,N), station(T-1,N'), N<N'.
0 { selected((N, N'),S1,S2);
    selected((N, N'),S2,S1) } 1 :- edge(S1,S2,C), sp((N, N')).
path((N, N'),X,Y)   :- selected((N, N'),X,Y).
```

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

```
path((N, N'),X,Z)   :- path((N, N'),X,Y), path((N, N'),Y,Z).
:- sp((N, N')), not path((N, N'), N, N').
cost((N, N'), C)    :- sp((N, N')),
                       C = #count { X,Y : selected((N, N'),X,Y) }.
#minimize { C,N,N' : cost((N, N'), C) }.
```

The first rule identifies the station pair `(N, N')` for which a shortest path is requested, specified by the question encoding using the `shortestPath` predicate. The second rule uses a choice rule to non-deterministically select directed edges that may form a path between the stations in the pair, allowing at most one direction of the edge. The next two rules define the transitive closure over the selected edges, determining which stations are reachable from each other via the current selection. The constraint ensures that the chosen edges indeed connect the given station pair, eliminating answer sets that fail to do so. The penultimate rule computes the number `C` of selected edges used for each station pair using the `#count` aggregate. Finally, the `#minimize` statement ensures that among all valid edge selections, the one with the smallest number of edges is preferred.

We note that the initial ASP theory is completely manually constructed. In our experiments, we simulate the extension process by removing rules from a "perfect" theory, that is one that always calculates the ground truth answer, to create an incomplete version. This means that we only use theories we know are correct; using faulty initial theories is not in the scope of this article but is an interesting possible direction that involves the remediation of an ASP program.

3 Knowledge distillation method

Our VQA approaches start from an initial ASP theory that is constructed for a particular dataset. Given a correct representation of both an image and question, the theory is always able to compute the correct answer. If the dataset is extended in a way that the theory no longer can answer correctly, rules need to be modified or added to handle new examples. To this end, we propose an LLM-based system to aid and provide automated support.

We aim for a reasoning module that manages the theory and is able to recognize which examples it can handle. If an example cannot be solved, the module updates the theory. We propose a method of *declarative knowledge distillation*, where the model we distill from is an LLM, and the distilled knowledge is represented as ASP rules. Our method involves different prompting strategies to obtain rules and to correct the incorrect answers of the LLM by using feedback from the ASP solver.

3.1 Rule distillation algorithm

We present our rule distillation method in Algorithm 1. Assume we want to extend an initial ASP theory T and we have a distillation suite of new VQA examples at our disposal. Each example is a triple (s, q, a) consisting of a scene description s as ASP facts, a question description q as ASP facts, and the expected answer a as an ASP fact.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

Algorithm 1 Rule distillation from LLMs.

```
1  # T: initial ASP theory
2  # distillation_suite: list of examples of form
3  #          (scene, question, answer)
4  regression_examples = []
5  for ex in distillation_suite:
6    s, q, a = ex
7    if asp_solver(T + s + q) == a:
8      continue
9
10   rules  = llm(multiprmpt, T, ex)
11          + llm(CoT_simple, T, ex)
12          + llm(CoT_complx, T, ex)
13
14   for r in rules:
15     if not syntax_check(T + r, ex):
16       r = llm(syntx_prmpt, T + r, ex)
17
18     if not semantics_check(T + r, ex):
19       r = llm(sem_prmpt, T + r, ex)
20
21     if regression_tests(T + r, reg_examples):
22       T = T + r
23       regression_examples.append(ex)
24       break
```

The algorithm iterates over all examples (line 5); for each example, it calls an ASP solver to check whether the expected answer can be derived from the theory accompanied with the scene and question facts. If so, it skips to the next example (line 8); otherwise it tries to update the theory.

We apply different prompting strategies – more details follow below – to obtain a list of candidates for extending the theory. This involves one strategy in which we ask for several variants of a solution in one prompt, as well as two different chain-of-thought prompts. Typically, the quality of the resulting rules varies and they are not always correct. By synergetic use of the prompting strategies, we thus create a pool of options from which we will subsequently aim to pick one that works (lines 10–12).

For each candidate rule (line 14), we use an ASP solver to deduce the answer for the current example:

- If we get a syntax error (line 15), we pass the error message to the LLM and prompt it to revise the rules (line 16); we call this step *syntactic mending*.
- If we get an answer set from the ASP solver, we check whether it matches the expected answer (line 18). If not, we pass the answer to the LLM and ask it to update the rules (line 19); we call this step *semantic mending*.
- We then we do *regression testing*, that is testing whether a new rule appended to the theory gives us correct answers for all previously used examples (line 21). This assures consistency and is in line with an online setting where future examples cannot be assumed to be known in advance. Only after passing this step, we accept

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use , available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

the suggested theory extension (line 22), add the example to the regression list (line 23) and break to the next example (line 24).

The models used tend to generally follow the instruction to return ASP rules, yet we observed that mistakes and hallucinations broadly fall into three categories: (i) syntactic, such as unsafe rules; (ii) semantic, where the generated rules fail to produce the expected answer for the current example; and (iii) regressive, where a rule may answer a specific question correctly but introduces a semantic conflict with previously distilled rules. These errors are systematically handled by our algorithm, using the ASP solver to ensure only syntactically valid and globally consistent rules are integrated into the theory. We illustrate the three error types using the target primitive `verify_attr(TO, TI, A, V)`, which checks whether an object at state `TI` possesses an attribute `A` with value `V`, and if so, state `TO` is assigned "yes."

- **(i) Syntactic errors:** The model generates invalid ASP syntax, often hallucinating mathematical notation (e.g. ∀, ∈) symbols that are not supported by the solver.

```
bool(TO, yes) :- verify_attr(TO, TI, ATTR, VALUE),
                 ∀ ID ⊆ state(TI, ID),
                 has_attr(ID, ATTR, VALUE).
```

- **(ii) Semantic errors:** The rule is syntactically valid but fails to produce the correct answer for the current example. In the case below, the model incorrectly requires the object to be "burnt" to satisfy the verification.

```
bool(TO, yes) :- verify_attr(TO, TI, ATTR, VALUE),
                 state(TI, ID),
                 has_attr(ID, ATTR, burnt).
```

- **(iii) Regressive errors:** The model generates a rule that works for the current example but conflicts with a previously distilled rule.

```
% Consider that the first input is an uncooked red lobster
% The model distills the following rule
bool(TO, no)  :- verify_attr(TO, TI, _, cooked),
                 state(TI, ID),
                 has_attr(ID, color, lobster).

% Now consider the following input is a cooked red steak
% The model distills the following rule
bool(TO, yes) :- verify_attr(TO, TI, _, cooked),
                 state(TI, ID),
                 has_attr(ID, class, red).
```

Now the program derives yes (and no) when we present the uncooked red lobster, provoking a regression failure.

3.2 Prompting

We prompt the LLM in Algorithm 1 for different purposes: either to generate new rules or to revise rules if a syntax or semantics test failed. For generating rules, we use different

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

prompting strategies, but all extend the same preprompt that sets the stage. The prompts are rather general and only need small adjustments if different VQA tasks are considered. Although it is well known that details of how prompts are formulated can have a great impact on the results, we are striving for clarity and readability instead of performance optimization. The complete prompts are presented in Appendix B.

3.2.1 Preprompt

Before prompting the LLM with VQA examples, we pass it a preprompt to prepare it for our rule distillation process. The preprompt instructs the LLM to only return ASP rules that extend an initial ASP theory. Most parts of the preprompt are dataset agnostic, but some, such as the scene, question, answer, and theory, are tailored to be suitable for each dataset. We illustrate preprompts and note whenever they are tailored for a specific dataset.

A preprompt consists of the following parts, where each of them is illustrated with an excerpt of the complete preprompt:

1. **Contextualization.** We describe the VQA setting and clarify that we have already parsed both scene and question into ASP facts.

   ```
   We are in the domain of Visual Question Answering, where the problem
   is to take an image and question related to it as input, and produce
   as output the correct answer.
   We have already preprocessed both the image and question into
   correct Answer Set Programming representations.
   Scene/Question pairs of ASP facts serve as the instance for an ASP
   program which we call the Theory.
   ```

2. **Logical representation.** We describe the syntax of the language that we use to represent questions and scenes, in our case ASP.

   ```
   Answer Set Programming (ASP) is a form of declarative programming
   oriented towards difficult search problems.
   Its syntax and usage can be summarized as follows:

   Rules: The basic building block of an ASP program. A rule has a head
   and a body, and is written in the form: Head :- Body.
   ```

3. **Scene and question representation.** We explain the representation that is used for describing the scene and the question, respectively, and give examples (dataset tailored).

   ```
   Consider the following ASP representation for the objects
   in an image.
   Object Declaration: Each object(<id>) declares a unique object
   with a specific identifier.
   ...
   Consider the following ASP representation for natural language
   questions.
   scene(S): Initializes all objects at step S.
   ```

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

4. **Answer representation.** We describe the format of the answers to the questions (dataset tailored).

```
The answer computed by the theory will always be in a predicate of
the form 'ans(X)', where 'X' is the expected answer.
'X' can be a yes or no, it can be an attribute, it can be a
relation or the value of some attribute.

Examples of these are:
ans(yes)
```

5. **Logical theory.** We present the initial theory T that should be extended (dataset tailored)

```
state(TO,ID) :- scene(TO), object(ID).
ans(V) :- end(TO), attr_value(TO,V).
ans(V) :- end(TO), attr(TO,V).
```

6. **Task explanation.** Finally, we explain the input that the LLM will receive from now on and the expected response.

```
Your task is to keep the ASP theory updated with rules that allows
us to handle new questions.
Your will be presented with a VQA example containing a scene,
question and expected answer.
```

While the distillation algorithm itself is domain-independent, certain components of the preprompt require manual adjustment when transitioning to a new dataset. Specifically, the engineering effort per dataset involves: (i) defining the scene representation, which describes how the symbolic solver views objects and their attributes; and (ii) defining the question representation, which outlines the available primitive operations and their expected arguments. Each of these sections is accompanied by examples of how the ASP representations for scene and questions look like (see Appendix B). We note that the majority of the preprompt structure remains static across tasks.

The preprompt will generally be followed by more detailed instructions of how to produce the new rules (e.g. by using multi-prompting or chain-of-thought prompting strategies).

3.2.2 Multi-prompting

We use the strategy that instructs the LLM to produce not only one solution for the task, but to suggest several alternative solutions; we call this *multi-prompting*. We apply this strategy in line 10 of Algorithm 1. The advantage is that we can possibly obtain several solution candidates that we can evaluate using a single call to the LLM. This increases the chances of finding a correct solutions without the need for several potentially expensive LLM calls. We use the following prompt:

```
Consider you must output {N} different possibilities of the rules
that need to be implemented.
Separate the different answers by rows of three # symbols.
```

The parameter `N` is set to 5 by default.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

3.2.3 Chain-of-thought

Chain-of-Thought (CoT) prompting is a technique in which an LLM is instructed to break down complex tasks into a series of intermediate steps (Wei *et al.* 2022). This often helps the model to produce a more accurate and coherent response. We adopted this idea for generating declarative rules, where we instruct the model to break down a problem solution as a set of if-then conditions and to translate them in a subsequent step into ASP rules. We use two versions of this idea in Algorithm 1: a simple, rather straight-forward version (line 11), and a more intricate one that asks for more detailed explanations (line 12). The simple version is as follows:

```
You are required to produce a chain of thought.
Use the predicates in the theory to show how you arrive at the
correct solution.
Only output one line per predicate used.
```

The complex version of chain-of-thought prompting, which is more verbose and structured than the previous, is as follows:

```
You are required to produce a chain of thought. Follow the
steps below to break down domain knowledge and construct the rules:
1. Understand the Problem
2. Break Down the Domain Knowledge
3. Formulate If-Then Statements
4. Construct Logic Rules
```

3.2.4 Prompts for rule mending

If the suggested rules do not produce the expected results, we prompt the LLM to correct them. We leverage feedback from the ASP solver to accomplish this. In particular, if a syntax test fails, we provide the LLM with the error message from the solver and a request to suggest a corrected version. Here is the prompt to achieve this:

```
You must repair the syntax of the prompted Answer Set Programming
rule(s).
The solver has given the following error: {syntax_error}.
```

If the syntax is correct but we do not get the expected answer, we pass the answer from the ASP solver to the LLM and explain that we expected the correct answer instead. The prompt looks as follows:

```
The prompted rules are added to the theory do not produce the
correct answer.
They do not calculate the correct answer, which is: {correct_answer},
but instead result in the following incorrect answer: {solver_answer}.
```

The semantic mending also includes the expected answer and the inferred answer, if any.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use , available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

4 Knowledge distillation experiments

We conduct a series of experiments to evaluate our knowledge distillation method on the GQA, CLEVR, and CLEGR^V datasets to answer the following research questions:[4]

(R1) Given an VQA task and correctly parsed examples of scenes, questions, and answers into ASP, can our approach extend the ASP theory to deal with questions that require operations/steps not yet encoded?
(R2) Which LLMs are suitable for our method?
(R3) How do the individual prompting strategies contribute to the overall performance?
(R4) How do LLMs fare when prompted to compose new predicates from already existing ones?

Before going into the details of our experiments, we discuss the evaluation platform, the selection of distillation and testing instances from the datasets, and our selection of LLMs.

4.1 Evaluation platform

The evaluation platform is a workstation with an Intel Core i7-12700K CPU, 32 GB of RAM, and an NVIDIA GeForce RTX 3080 Ti GPU with 12 GB of video memory. The ASP solver we use is `clingo` Gebser *et al.* (2019), version 5.8.0, from the Potassco solver suite,[5] an state-of-the-art system. All experiments were run 5 times with a timeout of 600 s. For reproducibility, we logged all our parameters, random seeds, input prompts, and resulting rules.

4.2 Instances

We create two sets of examples for each dataset, a *distillation suite* for the distilling process of Algorithm 1, and a *testing suite* to benchmark the distillation results. The suites were created aiming at having proportionally balanced examples of all primitive functions of each dataset.

For GQA, we used about $45k$ instances from its "balanced training set" and then divided this set into a distillation and test suites of about $35k$ and $10k$ instances, respectively. For CLEVR, the splits are taken from its training set and contain $35k$ examples for distillation and $15k$ for testing. For CLEGR^V, they also belong to the training set and contain $2k$ examples for distillation and $1k$ for testing.

Distillation sample. Our preliminary experiments showed that the rules generated during distillation tend to be general, often covering multiple examples at once when a rule is distilled. To control resource usage and demonstrate the efficiency of our approach with limited data, we do not use the complete distillation suite at every step, but instead sample a small, fixed number of examples ($N = 10$) for each distillation attempt. While using larger sample sizes may improve performance by exposing more patterns, it is more costly and can also lead to deterioration due to increased prompt length and the resulting

[4] The code for reproducing our experiments is available in an online repository: https://github.com/pudumagico/KDASP.
[5] https://potassco.org/clingo/

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

strain on the LLM's reasoning capabilities. We have chosen this value empirically after our preliminary experiments, as it was large enough to produce rules without prohibitive cost. On the other hand, for evaluation always the full test suite is used.

4.3 LLM selection

The landscape of LLMs is rapidly evolving, with new models and variants emerging frequently, making it infeasible to cover all available options in our evaluation. We therefore focus on a representative subset of competitive and accessible models.

We ran experiments with several LLMs, both locally hosted and online, API-based ones. Small models that can be easily self-hosted showed unfortunately very poor performance for the considered rule distillation task. We tried in particular GPT4ALL[6] "wizardlm-13b" and Zephyr 7b β,[7] which is a fine-tuned version of the Mistral 7B model developed by the Hugging Face H4 team.

Ultimately, we resort to larger API-based models. Our selection includes GPT-4o[8] from OpenAI, which is the company's most advanced model, notable for its multi-modal capabilities (text, vision, and audio) and high alignment performance, accessible via a paid API service. In addition, we include three open-source models: DeepSeek,[9] a flagship Chinese-English model designed for long-context reasoning and GPT-level performance; Mistral Large,[10] optimized for fast inference and general-purpose usage under an open-weight license (code not public but model weights are); Meta's LLaMA 3 70B-Instruct,[11] which offers strong instruction-following ability and high-quality outputs across tasks; and last, Gemini-3 Pro,[12] which exhibits a new "high reasoning" mode that in practice translates to an embedded preprompt that makes the LLM "think" before answering. In terms of scale, the models range from LLaMA 3 (70 billion parameters), Mistral Large ($\approx$ 123 billion) DeepSeek-V2 (236 billion), and Gemini-3 Pro (estimated 1 trillion), up to GPT-4o, with estimated 1.8 trillion parameters.

The temperature parameter of an LLM regulates the randomness of its output: lower values (e.g. near 0) make the model more deterministic by favoring high-probability tokens, while higher values introduce greater variability. We set a temperature of 0 for the LLMs which theoretically assures deterministic behavior. Yet in practice, there are multiple sources of non-determinism when working with LLMs, such as different tokenizers Ali *et al.* (2024), floating point errors, and versions of systems under the same name which can evolve in the background.

4.4 Theory extension simulation

For each VQA dataset, we start with the complete ASP theory T, and remove all rules that mention a selected predicate P that occurs in some question representation. Intuitively, P represents the ASP implementation of some primitive function, as described

[6] nomic.ai/gpt4all
[7] huggingface.co/HuggingFaceH4/zephyr-7b-beta
[8] https://platform.openai.com/docs/models/gpt-4o
[9] https://huggingface.co/deepseek-ai/DeepSeek-V2-Chat
[10] https://mistral.ai/news/mistral-large/
[11] https://huggingface.co/meta-llama/Meta-Llama-3-70B-Instruct
[12] https://deepmind.google/models/gemini/pro/

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

Table 1. *Restoring the initial theory* T *of GQA after removing all rules mentioning predicate* P

Predicate	GPT-4o	DeepSeek	Mistral	LLaMA3	Gemini-3
query	97.44 ± 2.48	98.01 ± 2.27	98.86 ± 0.08	86.62 ± 21.9	98.92 ± 0.00
exist	100.0 ± 0.00	99.98 ± 0.01	99.38 ± 1.06	89.58 ± 6.01	100.0 ± 0.00
or	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00	94.27 ± 1.67	100.0 ± 0.00
filter	97.24 ± 3.38	98.62 ± 2.76	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00
choose_attr	99.83 ± 0.00	99.79 ± 0.06	98.44 ± 2.76	99.74 ± 0.07	99.83 ± 0.00
verify_rel	98.94 ± 1.23	100.0 ± 0.00	94.70 ± 2.52	96.44 ± 3.00	100.0 ± 0.00
select	100.0 ± 0.00	99.77 ± 0.46	99.53 ± 0.38	32.20 ± 39.1	100.0 ± 0.00
negate	99.42 ± 0.70	100.0 ± 0.00	98.98 ± 0.57	99.04 ± 0.59	100.0 ± 0.00
relate	98.13 ± 2.69	98.76 ± 1.52	99.11 ± 1.54	76.74 ± 12.3	100.0 ± 0.00
two_different	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00	99.55 ± 0.45	100.0 ± 0.00
two_same	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00	98.76 ± 0.00	100.0 ± 0.00

Table 2. *Restoring the initial theory* T *of CLEVR after removing all rules mentioning predicate* P

Predicate	GPT-4o	DeepSeek	Mistral	LLaMA3	Gemini-3
exist	100.0 ± 0.00	95.98 ± 8.04	100.0 ± 0.00	81.18 ± 3.46	100.0 ± 0.00
unique	94.43 ± 10.5	100.0 ± 0.00	—	—	100.0 ± 0.00
count	100.0 ± 0.00	100.0 ± 0.00	98.80 ± 0.62	—	100.0 ± 0.00
equal_integer	100.0 ± 0.00	98.21 ± 3.10	100.0 ± 0.00	—	100.0 ± 0.00
and	100.0 ± 0.00	94.65 ± 2.74	100.0 ± 0.00	95.79 ± 2.98	100.0 ± 0.00
relate_left	100.0 ± 0.00	93.14 ± 8.70	100.0 ± 0.00	89.82 ± 7.20	100.0 ± 0.00
filter_large	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00	—	100.0 ± 0.00
query_shape	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00	88.01 ± 6.92	100.0 ± 0.00
same_color	100.0 ± 0.00	99.68 ± 0.64	100.0 ± 0.00	96.53 ± 2.46	100.0 ± 0.00

in Section 2, which can be encoded in the theory using one or more rules. We take a representative selection from the complete set of predicates and use samples as described in Subsection 4.2 from the distillation suite, where each example selected must mention P. We then iterate over the examples using Algorithm 1 to distill rules that can effectively calculate the expected answer.

The results for the datasets GQA, CLEVR, and CLEGR^V are shown in Tables 1, 2 and 3, respectively. The tables report, for each model and predicate in the dataset, the average number of examples used (n) and the average accuracy (%) achieved over the test suite, along with its standard deviation; the best results are in bold. The column labeled $\mathrm{T} \setminus P$ displays the accuracy of the initial ASP encoding with predicate P removed, which serves as a baseline. The drop in accuracy depends on the number of questions affected and the role of predicate P; for example `select` is used in almost every question of GQA as removing it drops the accuracy to 9.60%. The other columns illustrate the performance of the evaluated LLMs; "—" indicates that there is no improvement, meaning, no new rules where produced.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

Table 3. *Restoring the initial theory* T *of CLEGR^V after removing all rules mentioning predicate P*

Predicate	GPT-4o	DeepSeek	Mistral	LLaMA3	Gemini-3
cycle	99.19 ± 0.00	88.75 ± 0.32	87.44 ± 2.31	—	99.78 ± 0.00
shortest_path	91.73 ± 6.07	85.69 ± 1.93	85.10 ± 1.16	—	100.0 ± 0.00
paths	94.56 ± 2.98	80.74 ± 7.45	—	—	95.56 ± 2.12
count_nodes	91.91 ± 8.52	86.50 ± 0.38	—	—	100.0 ± 0.00
adjacent	100.0 ± 0.00	78.90 ± 7.30	88.99 ± 0.33	81.29 ± 1.96	86.11 ± 23.3
adjacent_to	100.0 ± 0.00	74.74 ± 11.8	—	88.59 ± 0.57	100.0 ± 0.00
same_line	98.76 ± 2.34	64.35 ± 22.7	87.14 ± 0.20	—	100.0 ± 0.00
common	98.60 ± 0.53	96.70 ± 3.70	—	—	98.22 ± 0.50
line_names	100.0 ± 0.00	87.62 ± 4.22	—	—	100.0 ± 0.00
line_count	100.0 ± 0.00	92.11 ± 8.68	—	—	100.0 ± 0.00
exist	95.38 ± 6.17	93.80 ± 3.10	—	—	100.0 ± 0.00

A general overview per dataset allows one to see that for both GQA and CLEVR, LLMs were highly successful when distilling rules for them. On the other hand, CLEGR^V appears to be more challenging, as accuracies are in general inferior to the ones in other datasets, and some models could not produce an answer in some cases. This can be explained by the difficulty of the predicates involved in CLEGR^V, such as `cycle` and `shortest path`, in contrast to the focus on spatial and comparison relationships from GQA and CLEVR, which are more straightforward to implement. The predicates in CLEGR^V are usually larger in terms of number of rules used to encode them but are also more complex, as they involve recursion, while none of the rules for GQA nor CLEVR are recursive.

Analyzing the results of every model reveals consistent patterns across datasets. GPT-4o remained the most reliable model overall, achieving near or perfect accuracy with low deviation, across most predicates and datasets. Notably on CLEVR, it was able to recover or produce semantically equivalent rules to the ones removed. DeepSeek showed strong performance as well, often trailing GPT-4o on GQA and CLEVR, but noticeably worse in CLEGR^V.

Mistral-Large demonstrated competitive results as well, being on par with DeepSeek on GQA and CLEVR (with the exception of the `unique` predicate in CLEVR where no rules were produced). Yet it followed the trend of deteriorating results for CLEGR^V, now some distillation attempts on particular predicates without improvement.

Last, LLama3-70b's performance was acceptable for GQA, producing rules for almost every predicate, but with high variance and a low score on `select`, where the accuracy was only 32.20%. For CLEVR, and in particular CLEGR^V, the results were noticeable worse; no new rules were produced for the distillation of many predicates.

Gemini-3 demonstrated exceptional performance across all datasets, consistently achieving near-perfect accuracy (Tables 1–3). Notably, it achieved 100% accuracy on several complex graph-based predicates in CLEGR^V, where other high-end models showed slight performance drops. We attribute this to the model's high-reasoning mode, which allows an LLM to (i) explore multiple reasoning paths, (ii) using "verification loops" to ensure that their reasoning is correct, and (iii) allocates a significant share of time to

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

Table 4. *Average number of examples used by model and dataset*

Dataset	GPT-4o	DeepSeek	Mistral	LLaMA3	Gemini-3
GQA	4.04 ± 1.09	2.16 ± 0.32	4.02 ± 1.50	4.03 ± 0.67	2.79 ± 0.10
CLEVR	3.22 ± 0.50	2.34 ± 0.16	3.90 ± 1.29	6.48 ± 1.83	2.58 ± 0.32
CLEGR^V	4.02 ± 1.51	2.40 ± 0.32	5.80 ± 0.75	5.02 ± 0.24	1.73 ± 0.53

Table 5. *Average and standard deviation token usage per dataset and model*

	GQA		CLEVR		CLEGR^V	
Model	In	Out	In	Out	In	Out
GPT-4o	11259 ± 762	359 ± 353	6862 ± 743	356 ± 42	3615 ± 336	1931 ± 206
DeepSeek	8096 ± 360	319 ± 24	3092 ± 445	793 ± 130	3321 ± 357	325 ± 458
Mistral-Large	13591 ± 1494	788 ± 635	6551 ± 532	475 ± 540	12604 ± 1188	1331 ± 164
LLaMA3-70B	33565 ± 2494	2587 ± 2048	25935 ± 3007	2942 ± 3708	39716 ± 2574	5166 ± 4701
Gemini-3	8050 ± 205	128 ± 40	3892 ± 311	82 ± 36	4585 ± 278	368 ± 214

this preparatory reasoning step. In our setting, these characteristics are translated into a model that is more precise and robust than the rest.

The findings highlight that the distillation process used few examples and proved considerably successful for GPT-4o, DeepSeek, Mistral-Large and Gemini-3 on GQA and CLEVR, whereas Mistral-Large and LLaMA3-70b struggled on CLEGR^V where complex graph reasoning tasks posed greater difficulty.

Regarding the number of examples used, it always happened that fewer than all available $N = 10$ examples per distillation attempt were used. This is because the LLM may have already produced a set of rules general enough such that it can cover all future examples, or a set of rules was produced that the LLM could not improve upon. In general, the amount was remarkably low, using 3–8 examples to produce a set of rules. While we set an upper limit of $N = 10$ examples for our experiments, in practice, typically we use 1–4 examples per distillation attempt as shown in Table 4. The LLM often reaches a "fixpoint" during the process: once it produces a rule general enough to cover the representative logic of a question type, presenting further examples does not result in additional rules. Gemini-3 was highly efficient, often requiring the fewest number of examples (averaging to only 1.73 for CLEGR^V) to converge on a correct set of rules.

For practitioners, the efficiency and operational cost of the distillation process are primarily determined by the token consumption of the LLMs. While input costs are largely a function of the preprompt size and the number of examples, output costs vary significantly based on the model's verbosity and its ability to adhere to the constraint of returning only ASP rules. Table 5 provides a breakdown of the average input and output tokens per distillation attempt on 100 distillation attempts for each dataset using a random predicate. We observe that the results align with the number of examples

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

Table 6. *Average accuracy by model and ablation condition. For each dataset, best configuration per model is in bold, best model under full configuration is underscored*

Dataset	Model	Full	No mending	No CoT	No multi-prompt
GQA	GPT-4o	$\mathbf{99.18 \pm 1.09}$	97.63 ± 2.28	98.06 ± 1.59	96.84 ± 5.54
	DeepSeek	$\mathbf{99.54 \pm 0.72}$	98.97 ± 1.54	99.08 ± 1.81	99.36 ± 1.32
	Mistral-Large	$\mathbf{99.00 \pm 1.53}$	97.94 ± 2.91	96.06 ± 5.64	98.74 ± 2.19
	LLama3-70b	$\mathbf{88.45 \pm 20.0}$	86.01 ± 23.2	82.98 ± 8.43	86.43 ± 16.5
	Gemini-3	$\mathbf{99.89 \pm 0.32}$	98.89 ± 0.59	99.88 ± 0.52	97.45 ± 0.45
CLEVR	GPT-4o	$\mathbf{99.38 \pm 1.86}$	98.43 ± 4.72	96.01 ± 11.4	96.85 ± 9.44
	DeepSeek	97.96 ± 2.69	98.61 ± 2.59	$\mathbf{99.85 \pm 0.45}$	93.58 ± 18.9
	Mistral-Large	98.74 ± 0.52	95.07 ± 14.0	99.11 ± 1.19	$\mathbf{99.21 \pm 0.51}$
	LLama3-70b	94.07 ± 6.33	$\mathbf{98.94 \pm 0.70}$	98.79 ± 0.95	87.91 ± 22.5
	Gemini-3	$\mathbf{100.00 \pm 0.00}$	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00
CLEGR^V	GPT-4o	$\mathbf{97.28 \pm 3.28}$	86.86 ± 4.14	85.62 ± 4.34	88.34 ± 1.61
	DeepSeek	$\mathbf{88.54 \pm 9.34}$	88.16 ± 1.90	86.98 ± 3.27	88.20 ± 1.58
	Mistral-Large	87.17 ± 0.88	$\mathbf{87.25 \pm 1.44}$	84.25 ± 8.37	87.04 ± 1.83
	LLama3-70b	$\mathbf{86.76 \pm 1.87}$	86.33 ± 3.14	85.28 ± 6.61	85.39 ± 4.10
	Gemini-3	$\mathbf{98.45 \pm 4.22}$	98.45 ± 4.22	97.51 ± 1.31	98.12 ± 4.75

needed per distillation attempt shown in Table 4, as models like DeepSeek and Gemini-3 use fewer tokens than the competitors. This is because their higher precision reduces the number of failed distillation attempts and repeated API calls triggered by the feedback loop. In contrast, other models like LLama often incur higher costs through multiple retries and exhibit high variance.

4.5 Ablation experiments

Our distillation algorithm consists of a series of constituents including syntax mending, semantic mending, multi-prompting, two types of CoT and regression testing. We produce ablation studies focusing on three major elements of our prompting approach, which we refer to as *distillation strategies*:

- **No mending**: When removing both syntactic and semantic mending,
- **No CoT**: When removing both simple and complex chain of thought prompts, and
- **No multiprompt**: When removing multi-prompting from the prompt.

Table 6 reports the average accuracy of each model on each dataset with and without ablation. The best accuracy in each row is shown in bold; the best model without ablation for each dataset is additionally underlined. When a model failed to generate valid rules for a predicate, the accuracy was replaced with that of the theory $\mathrm{T} \setminus P$ as baseline.

Across datasets, GPT-4o and DeepSeek achieve consistently strong results. GPT-4o in particular benefits from the full prompting pipeline, especially on GQA and CLEGR^V: disabling any strategy in these settings leads to clear performance drops of up to 10%. DeepSeek shows similar behavior, though in CLEVR, disabling CoT slightly improves performance, indicating that simpler prompting can sometimes improve performance even on strong models. In CLEGR^V, the impact of these strategies is generally less

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use , available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

pronounced. Mistral-Large and LLama3-70b, which trail behind on GQA and CLEGR^V, perform comparably to larger models on CLEVR, especially when some strategies are deactivated. In these cases, reduced prompting appears to ease the burden on smaller models. For instance, LLama3-70b improves by 5%–10% on several CLEVR predicates when mending is disabled. The same models exhibit high variance and inconsistent behavior when distilling rules. LLama3-70b frequently fails to distill rules for harder tasks in CLEGR^V, regardless of strategy, and Mistral-Large sometimes fails completely for certain GQA or CLEGR^V predicates if strategies like multi-prompting are disabled. The ablation study for Gemini-3 reveals a remarkable stability compared to other models. While most models suffered significantly when mending or multi-prompting were disabled, Gemini-3 maintained an average accuracy of over 93% even without mending. Unlike Mistral or LLaMA3, Gemini-3 was not hindered by complex prompting strategies, showing that its reasoning depth is sufficient to handle the full pipeline without being overburdened.

A more thorough analysis of the experiments is given in Appendix C.

The efficacy of distillation strategies is not uniform but rather highly correlated with the intrinsic reasoning capabilities of the underlying model. We observe three distinct performance archetypes: (i) *Overburdened Models* (Mistral-Large, LLaMA3-70b), where the computational or context overhead of complex strategies like CoT and multi-prompting often outweighs their benefits, leading to better performance when these modules are deactivated; (ii) *Balanced Models* (GPT-4o, DeepSeek), which effectively leverage these strategies to bridge reasoning gaps, showing clear performance gains when all distillation components are active; and (iii) *Saturated Models* (Gemini-3), which possess such high baseline capabilities that they render some strategies redundant.

4.6 Composing new ASP predicates

We explore furthermore the synthesis of new ASP predicates by composing existing operations from a fixed ASP theory T, specifically on the GQA dataset. The primitives (e.g. `select`, `has relation`, `filter`, `verify_rel`) provide the foundational semantics for reasoning over visual scenes. Our goal is to test whether language models can define new predicates compositionally from those in T, in particular those with higher arity and those that use different ASP constructs.

We showcase how to introduce a new predicate, `in_between`, which is a spatial predicate that relates three objects, and evaluate whether LLMs can provide its semantics using only the existing predicates in T.

The process unfolds in several steps:

1. **Manual extension of the theory:** We begin by adding rules to handle a new predicate P to T, such that it implements the desired predicate using predicates previously defined. For example, the predicate `in_between` can be implemented as:

```
state(T0, X) :- in_between(T0, T0, T1, T2),
                state(T1, ID1),
                state(T2, ID2),
                has_rel(X, to_the_right_of, ID1),
                has_rel(X, to_the_left_of, ID2).
```

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

Table 7. *Distillation for producing new predicates removing all rules mentioning predicate P. Best average accuracy per predicate in bold*

P	GPT-4o	DeepSeek	Mistral-Large	LLama3-70b	Gemini-3
in_between	97.39 ± 2.82	$\mathbf{99.23 \pm 1.31}$	93.46 ± 3.26	92.82 ± 3.20	98.85 ± 1.15
connected	$\mathbf{100.0 \pm 0.00}$	99.84 ± 0.32	99.92 ± 0.16	84.20 ± 22.4	$\mathbf{100.0 \pm 0.00}$
isolated	$\mathbf{100.0 \pm 0.00}$	$\mathbf{100.0 \pm 0.00}$	$\mathbf{100.0 \pm 0.00}$	74.09 ± 26.3	$\mathbf{100.0 \pm 0.00}$
count_class	$\mathbf{100.0 \pm 0.00}$	$\mathbf{100.0 \pm 0.00}$	$\mathbf{100.0 \pm 0.00}$	–	$\mathbf{100.0 \pm 0.00}$

2. **Sampling modifiable examples:** We identify examples where the structure of the scene graph allows for a transformation into a query that involves the new predicate. These examples are modifiable into ones that speak about the desired predicate. For example, for the `in_between` predicate, one searches for scenes with three or more objects.
3. **Transforming the examples:** Each selected example is transformed to
   - a new ASP question using the new predicate (e.g. `in_between`),
   - an updated answer, and
   - an optionally modified scene if additional attributes or relations are needed.
4. **Dataset splitting:** We create distillation and testing suites.
5. **Rule distillation:** To obtain new rules, we use Algorithm 1, testing whether the LLM can produce P by composing existing operations from T using the generated examples as guidance.

This method enables us to use a VQA dataset, in this case GQA, to probe the capabilities of LLMs to generate new predicates from existing ones. Besides `in_between`, we construct additional predicates to further evaluate predicate composition (their ASP encodings are given in Appendix D):

- For `connected`, we select two objects from the scene and ensure the presence of a path between them through one or more intermediate objects, that is we synthesize relational chains by adding `has_rel(A, connects, B)` relations (e.g. object A is next to object B, and B is to the left of C) to form paths of length greater than two. The query then tests whether such indirect connections exist.
- For `isolated`, we identify an object and systematically remove all of its relationships from the scene graph, making it structurally disconnected. The associated query asks whether the object is isolated.
- For `count_class`, we leave the scene unchanged and simply construct a query that counts the number of objects belonging to a given class. We note counting is not part of the original GQA primitives.

We use new distillation and test suites following the procedure described before, with 500 examples each.

For each new predicate, we report the average number n of examples used, accuracy, and standard deviation, as shown in Table 7.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

GPT-4o can successfully provide the intended semantics for all novel predicates `connected`, `isolated`, and `count_class` with perfect accuracy, the exception being `in_between`, which achieves ca. 97%.

Deepseek and Mistral both match GPT-4o on `isolated` and `count_class`. The former even surpasses GPT-4o on `in_between`, but at the cost of a lower than perfect score on `connected`. Mistral presents lower scores than DeepSeek on both `in_between` and `connected`. LLama3-70b produced the worst results, in particular no rules were produced for `count_class`. Last, Gemini-3 positions itself again as the best performant model overall, achieving perfect accuracy on most of the predicates.

This experiment demonstrates the potential of LLMs to extend symbolic theories compositionally, a critical capability for scenarios where rules must be dynamically generated or adapted to new domains.

4.7 Assessment of research questions

We now turn to our research questions R1–R4:

(R1) Given a VQA task and correctly parsed examples of scenes, questions, and answers into ASP, can our approach extend the ASP theory to deal with questions that require operations/steps not yet encoded?
LLMs can effectively recover missing ASP rules, with strong performance on GQA and CLEVR. Although performance drops on CLEGR^V, key primitive operations were still captured, promoting the viability of rule distillation across domains.

(R2) Which LLMs are suitable for our method?
Only large-scale models like GPT-4o, DeepSeek and Gemini-3 consistently succeed across tasks. Mistral-Large performs moderately well, while LLama3-70b struggles, showing that both scale and instruction-following capabilities are crucial for robust rule distillation.

(R3) How do the individual prompting strategies contribute to the overall performance?
Distillation strategies (mending, CoT, multi-prompting) substantially improve performance for top models, but can hinder weaker ones. GPT-4o and DeepSeek depend on them for complex cases, while Mistral and LLama3-70b often benefit from simpler prompting.

(R4) How do LLMs fare when prompted to compose new predicates from already existing ones?

GPT-4o, DeepSeek, and Gemini-3 can compose new ASP predicates using primitives from the base theory, even for recursive and aggregation-based rules. Mistral succeeds on simpler constructs, while LLama3-70b remains unreliable for compositional synthesis.

Regarding scalability with respect to the number of rules of the initial theory, we observe that the distillation algorithm operates locally by processing one primitive predicate at a time. Consequently, the total size of the theory has low impact on the effectiveness of it. In fact, larger initial theories may improve performance by providing the LLM with richer context to prevent reasoning drift, whereas extremely sparse theories risk insufficient context. In terms of reasoning capabilities, our experiments across GQA, CLEVR, and CLEGR^V demonstrate robustness in spatial, equality, counting, and

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

recursive graph-related tasks, covering a vast landscape of ASP constructs. A future direction is to extend this approach to temporal datasets, leveraging ASP's native temporal extensions[13] to handle time-dependent predicates within the VQA encoding.

In summary, our results show that LLMs can effectively recover missing ASP rules in the context of VQA and extend symbolic theories with new logical rules. Success depends heavily on both dataset complexity and model capabilities: simpler tasks in GQA and CLEVR are addressed with high accuracy, while structurally demanding domains like CLEGR^V, which involve recursion and aggregation, require larger models and robust prompting strategies.

5 Pruning extracted rules

It happened that many logic programs produced in our experiments include *redundant rules* – that is rules whose removal does not affect the answer of the queried predicates. Redundant rules can arise from overly specific rule instantiations or responses by LLMs where a correct rule is produced alongside other innocuous ones which do not interfere with the correct computation.

Redundancy increases the complexity of rule sets, makes analysis and debugging harder, and may obscure the core reasoning pattern. Furthermore, in neurosymbolic systems where extracted rules are passed back into neural architectures (e.g. for neurosymbolic learning or data augmentation), compactness and non-redundancy are crucial for interpretability and computational efficiency.

We now illustrate on output of our distillation algorithm how pruning a selection of the produced rules may simplify extracted rule sets without loss of accuracy with respect to some distillation data.

Consider an instance of rules distilled for the `select` predicate in the GQA dataset using GPT-4o:

```
state(TO,ID)   :- select(TO, TI, CLASS), state(TI, ID),
                  has_attr(ID, class, CLASS).
state(TO,ID)   :- select(TO, TI, CLASS), unique(TO, TI), state(TI, ID),
                  has_attr(ID, class, CLASS).
is_attr_value(ID, X) :- query(TO, TI, ATTR), state(TI, ID),
                        has_attr(ID, ATTR, X).
state(TO,ID)   :- select(TO, TI, CLASS), state(TI, ID),
                  has_attr(ID, class, CLASS).
```

This example contains multiple rules produced by the LLM during the distillation process that are not useful to solve the *select* query. After eliminating the redundant rules, the set reduces to:

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

The pruned theory achieves 100% accuracy on the GQA distillation suite, equal to the accuracy without pruning

[13] https://github.com/potassco/telingo

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

Algorithm 2 Rule redundancy elimination.

```
1  # T\P: initial theory
2  # R: list of new ASP rules produced by the LLM
3  # pruning_sample: sample of training scenes, questions, and answers
4
5  redundant_rules = []
6
7  for ri in R:
8   Q = R - {ri}
9   if evaluate(T\P + R, pruning_sample) == evaluate(T\P + Q, pruning_sample):
10    redundant_rules.append(ri)
11
12 R = R - set(redundant_rules)
```

5.1 Rule redundancy heuristic

We present our redundancy elimination procedure in Algorithm 2. Assume we have the results of a distillation process for a predicate P: a set R of new ASP rules produced by the LLM, the initial theory $T \setminus P$, and a pruning sample drawn from the distillation suite.

Since rules generated later in Algorithm 1 tend to be more general due to regression testing, as new distilled rules must capture past examples as well, we attempt to eliminate rules from first to last. The algorithm iterates over all rules $r_i \in R$ in the order they were generated (line 7).

For each rule, it temporarily removes r_i to form a reduced set Q (line 8), and calls `evaluate` to compare the behavior of extending T with both sets on the pruning sample (line 9). If the outputs are identical, the rule is marked as redundant (line 10). After all candidates have been checked, the redundant rules are removed from R (line 12).

We use a sample from the examples for distillation also for pruning. If a rule can be removed without affecting correctness on this sample, it is marked as redundant and discarded.

Figure 3 visualizes the outcomes of applying Algorithm 2 on the new rules generated in the main experiments (Section 4.4) for GQA, CLEVR, and CLEGR^V; values in tabular form are presented in Appendix E. For each predicate, the bars indicate the average number of rules pruned across the sets of new rules produced by the LLM. We use a sample of 1000 examples from the corresponding distillation suite of each predicate.

Across all datasets, pruning reveals meaningful differences in how LLMs approach rule generation. GPT-4o consistently produces more extensive rule sets, often leading to high pruning counts – such as 8.8 rules for *unique* in CLEVR – suggesting a thorough but sometimes redundant exploration of possible solutions. In contrast, Mistral-Large and LLama3-70b tend to yield more compact outputs, with near-zero pruning for several predicates, such as *relate* or *select* in GQA. Gemini-3 exhibited the lowest pruning rates. In many cases, it produced zero redundant rules. This indicates that the model "high" reasoning mode allows to target the exact logic required by the VQA primitive as an ASP rule.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684



(a) GQA. (b) CLEVR. (c) CLEGR^V.

Fig. 3. Average number of rules pruned for each predicate across different models on the GQA, CLEVR, and CLEGR^V datasets.

Low pruning, combined with high accuracy and a low number of examples, indicates that the LLM has produced general and useful rules. In contrast, high pruning with many examples suggests exploratory behavior, where the LLM initially generates a broader set of rules that are gradually refined through the distillation process. However, when high pruning occurs despite few examples being used, it may indicate that the LLM is generating superfluous rules – statements that do not affect the outcome.

GQA and CLEVR present the highest pruning rates overall, as their simpler and more compositional predicates encourage LLMs to generate varied candidate rules. In contrast, CLEGR^V shows consistently lower pruning counts across models – not due to more efficient generation, but because its increased complexity makes rule synthesis harder in the first place, often resulting in fewer or no rules to prune. The pruned rules – alongside observed syntactic or semantic errors – offer possibly valuable data that can be potentially incorporated as prompt feedback or leveraged as training data to fine-tune models toward more accurate rule generation.

6 Related work

Our work intersects several areas of research that integrate symbolic reasoning with machine learning. In the following, we review relevant developments in ASP and LLMs, SRL, and modular neurosymbolic approaches to VQA.

6.1 Answer-set programming and LLMs

Recently, LLMs have been explored in the context of ASP. Ishay *et al.* (2023) observed that LLM reasoning capabilities are shallow, but they can serve as a highly effective semantic parser to transform input into ASP representations. These are then used to solve logical puzzles. Furthermore, Bauer *et al.* (2024) recently proposed to use LLMs to parse questions into ASP facts in the context of VQA for images of graphs.

Rajasekharan *et al.* (2023) showed that the combination of LLMs with ASP in their STAR framework produces good results for natural language tasks that require qualitative reasoning, mathematical reasoning, and goal-directed conversation. Santana *et al.* (2024) proposed a two-step system for translating natural language to ASP via a controlled natural language intermediate form, demonstrating accurate rule generation on graph-based tasks. LLASP (Coppolillo *et al.* 2024) is an approach that involves fine-tuning LLMs specifically for ASP, showing that model adaptation can significantly improve rule generation accuracy in logic programming tasks.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

Our work is different as we do not focus on semantic parsing but on the more challenging task of knowledge distillation, where we aim for a system that can produce sound logical rules capturing knowledge about a particular domain.

Recent work by Zhu *et al.* (2023) explores the use of LLMs to learn rules from arithmetic and kinship relationships, yet the rules they learn do not contain variables and their semantics is informal.

6.2 Statistical-relational learning

Similar to our approach, methods from SRL (Raedt and Kersting 2017), in particular from ILP (Muggleton 1991; Muggleton and Raedt 1994; Cropper *et al.* 2022), aim to produce rules from example data and a background theory. SRL has seen great advances in terms of scalability by, for example applying gradient-based boosting (Gutmann and Kersting 2006), and systems like ILASP (Law *et al.* 2020b) and FastLAS (Law *et al.* 2020a) provide means for inductively learning expressive ASP programs. Kareem *et al.* (2024) combine LLMs with ILASP to induce commonsense rules from answer sets for robust QA, requiring only few examples.

SRL adopts a statistical and probabilistic perspective that is fundamentally data-driven. In contrast, our approach does not perform learning. While we also make use of a dataset, its role is to guide a knowledge distillation process by conditioning an LLM, rather than to serve as training data. Notably, only a few examples are sufficient for this purpose.

Traditional ILP uses a search-based approach where the solutions produced are correct and minimal under some criteria. A key aspect of many ILP systems are *mode declarations* that define the syntactic form of allowed rules to restrict the search space of possible programs. They are given in a formal language and tacitly assume an intuition about the form of the solution. The distillation approach does not require this; we only elicit knowledge that is already present in the LLM, and the information in the prompt that instructs what rules we want is informal and in natural language. When prompting LLMs, rules are general by command – while optimality is not enforced, it may happen implicitly. In very recent work, Yang *et al.* (2025) propose to use LLM agents to generate the language bias itself, enabling automated hypothesis generation in ILP.

6.3 Modular neurosymbolic VQA

There are several VQA systems that feature a modular architecture that combines sub-symbolic with symbolic components (Yi *et al.* 2018; Mao *et al.* 2019; Amizadeh *et al.* 2020; Eiter *et al.* 2022; Surís *et al.* 2023; Johnston *et al.* 2023).

Specifically, Yi *et al.* (2018) used a pipeline to extract a *scene graph* (a list of all objects detected in the image with their attributes and positions) from the image. They then translated the provided question into a structured representation of the reasoning steps, called *functional program*, and executed this program on the structural scene representation to obtain an answer. The authors showed excellent results on the popular CLEVR dataset (Johnson *et al.* 2017). This approach has been advanced with logic-based reasoning processes, for example by Differentiable First-Order Logic (∇-FOL) (Amizadeh

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

et al. 2020), or by ASP (Eiter *et al.* 2022). These reasoning processes can consider not just the most probable scene-graph prediction, but rather the entire vector of probabilities as output by the object detection and attribute/relation classifier networks that form the visual perception component of the VQA pipeline.

While our focus is on VQA, related developments exist in textual QA. For instance, Nguyen *et al.* (2025) use LLMs in combination with ASP to detect misleading content by generating logic-based explanations. Their system highlights parts of a text responsible for incorrect or hallucinated answers, illustrating the potential of symbolic reasoning for enhancing transparency of QA pipelines.

Foundational models including Vision-Language Models (VLMs), such as BLIP-2 (Li *et al.* 2023) and SimVLM (Wang *et al.* 2022) have become sufficiently strong through their pre-training regimes to generalize well to multiple different datasets. Approaches that use these VLMs as components are, for example ViperGPT (Surís *et al.* 2023), CodeVQA (Subramanian *et al.* 2023), and PnP-VQA (Tiong *et al.* 2022).

Regarding systems for neurosymbolic learning, it should be mentioned that there are logic-based systems that employ the semantic loss (Xu *et al.* 2018) to use the information produced by the reasoning module to improve the learning tasks of the neural networks involved (Yang *et al.* 2020; Manhaeve *et al.* 2021).

7 Conclusion

We have presented a method for declarative knowledge distillation using large language models (LLMs), which serves to generate rules for the reasoning component of a VQA system. By leveraging examples from a dataset as guidance and relying exclusively on prompting – without requiring fine-tuning – the approach enables an automated extension of answer-set programming (ASP) theories. We have demonstrated the usefulness of the method across diverse datasets; while designed for VQA, it is in principle generic and could be explored in other contexts as well.

Combining logic-based reasoning with neural models provides systems that are interpretable and verifiable, properties that are crucial for trustworthy AI. Logical rules are an intuitive and transparent basis for encoding complex reasoning behavior and can facilitate explainability in neurosymbolic architectures. Our results show that LLMs can effectively synthesize ASP rules to extend reasoning capabilities in VQA, a task that requires understanding and manipulating symbolic representations. Among the evaluated models, GPT-4o and DeepSeek outperform others, particularly when enhanced with distillation strategies such as multi-prompting and chain-of-thought (CoT) prompting. Furthermore, Gemini-3 Pro demonstrated exceptional capability, consistently achieving near-perfect accuracy across all datasets while producing the most compact rule sets. This superior performance is attributed to the model's "high reasoning mode," which allows the model to internally explore and validate logical paths before generating the output ASP rule(s). Notably, the ability of LLMs to propose rule completions and revisions indicates that they could serve as interactive *copilots*, supporting practitioners in the development of logical programs.

Despite its promise, the proposed method has several limitations. It assumes the availability of datasets with decomposable, compositional question semantics, which may

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

restrict its applicability in domains lacking such structure. Prompt sensitivity remains a challenge, though our templates demonstrated robust performance across different datasets. Furthermore, rule correctness is validated empirically rather than formally, although constraints and regression tests mitigate some potential errors.

Future work includes applying prompt tuning to better specialize LLMs for ASP rule generation, aiming to improve accuracy and reduce example requirements. We also plan to extend our distillation method beyond VQA to dynamic domains (e.g. biomedical knowledge, real-time monitoring, learning agents) and structured question answering, where symbolic inputs and multi-step reasoning are central.

Acknowledgements

The authors acknowledge TU Wien Bibliothek for financial support through its Open Access Funding Programme. Additionally, this work was partially funded from the Bosch Center for AI.

References

Abraham, S. S., Alirezaie, M. and Raedt, L. D. 2024. CLEVR-POC: Reasoning-intensive visual question answering in partially observable environments. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC/COLING 2024)*, ELRA and ICCL, 3297–3313.

Ali, M., Fromm, M., Thellmann, K., Rutmann, R., Lübbering, M., Leveling, J., Klug, K., Ebert, J., Doll, N., Buschhoff, J. S., Jain, C., Weber, A. A., Jurkschat, L., Abdelwahab, H., John, C., Suarez, P. O., Ostendorff, M., Weinbach, S., Sifa, R., Kesselheim, S. and Flores-Herr, N. 2024. Tokenizer choice for LLM training: Negligible or crucial? In *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16–21, 2024*, K. Duh, H. Gómez-Adorno and S. Bethard, Eds. Association for Computational Linguistics, 3907–3924. https://doi.org/10.18653/v1/2024.findings-naacl.247.

Alviano, M., Dodaro, C., Faber, W., Leone, N. and Ricca, F. 2013. WASP: A native ASP solver based on constraint learning. In *Logic Programming and Nonmonotonic Reasoning, 12th International Conference, LPNMR 2013, Corunna, Spain, September 15–19, 2013. Proceedings*, P. Cabalar and T. C. Son, Eds. Vol. 8148 of Lecture Notes in Computer Science, Springer, 54–66. https://doi.org/10.1007/978-3-642-40564-8.

Amizadeh, S., Palangi, H., Polozov, A., Huang, Y. and Koishida, K. 2020. Neuro-symbolic visual reasoning: Disentangling "Visual" from "Reasoning". In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, Vol. 119 of Proceedings of Machine Learning Research, PMLR, 279–290.

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L. and Parikh, D. 2015. VQA: Visual question answering. In *2015 IEEE International Conference on Computer Vision (ICCV)*, IEEE Computer Society, 2425–2433. https://doi.org/10.1109/ICCV.2015.279.

Auer, C., Bachmaier, C., Brandenburg, F. J., Gleissner, A. and Reislhuber, J. 2013. Optical graph recognition. In *Graph Drawing*. Springer Berlin Heidelberg, Berlin, Heidelberg, 529–540.

Barbara, V., Guarascio, M., Leone, N., Manco, G., Quarta, A., Ricca, F. and Ritacco, E. 2023. Neuro-symbolic AI for compliance checking of electrical control panels. *Theory and Practice of Logic Programming* 23, 4, 748–764.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

Barra, S., Bisogni, C., Marsico, M. D. and Ricciardi, S. 2021. Visual question answering: Which investigated applications? *Pattern Recognition Letters* 151, 325–331. https://doi.org/10.1016/j.patrec.2021.09.008.

Basu, K., Shakerin, F. and Gupta, G. 2020. AQuA: ASP-based visual question answering. In *22nd International Symposium on Practical Aspects of Declarative Languages (PADL, LNCS 2020)*, Vol. 12007 of LNCS, Springer, 57–72. https://doi.org/10.1007/978-3-030-39197-3_4.

Bauer, J. J., Eiter, T., Ruiz, N. H. and Oetsch, J. 2023. Neuro-symbolic visual graph question answering with LLMs for language parsing. In *Workshop on Trends and Applications of Answer Set Programming (TAASP 2023)*. https://doi.org/10.34726/5462.

Bauer, J. J., Eiter, T., Ruiz, N. H. and Oetsch, J. 2024. Visual graph question answering with ASP and LLMs for language parsing. In *Proceedings 40th International Conference on Logic Programming, ICLP 2024 Technical Communications/Doctoral Consortium, Dallas,USA, October 11–17, 2024*, P. Cabalar, F. Fabiano, M. Gebser, G. Gupta and T. Swift, Eds. Vol. 416 of EPTCS. https://doi.org/10.4204/EPTCS.416.

Borroto, M., Ielo, A., Mazzotta, G. and Ricca, F. 2025. Answer set programming and neurosymbolic AI: Applications and future perspectives (invited talk). In *Hybrid Models for Coupling Deductive and Inductive Reasoning*, P.Bruno, F. Calimeri, F. Cauteruccio and G. Terracina, Eds. Springer Nature Switzerland, Cham, 1–21.

Brewka, G., Eiter, T. and Truszczynski, M. 2011. Answer set programming at a glance. *Communications of the ACM* 54, 12, 92–103. https://doi.org/10.1145/2043174.2043195.

Bruno, P., Calimeri, F., Marte, C. and Manna, M. 2021. Combining deep learning and asp-based models for the semantic segmentation of medical images. In *Proceedings of the 5th International Joint Conference on Rules and Reasoning, RuleML+RR 2021*, Vol. 12851 of Lecture Notes in Computer Science, Springer, 95–110. https://doi.org/10.1007/978-3-030-91167-6_7.

Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Maratea, M., Ricca, F. and Schaub, T. 2020. ASP-core-2 input language format. *Theory and Practice of Logic Programming* 20, 2, 294–309.

Chowdhury, A., Shakarian, P. and Simari, G. 2025. Abduction of domain relationships from data for VQA. In *Electronic Proceedings in Theoretical Computer Science*, Vol. 416, EPTCS, 168–174. Publisher Copyright: © 2025 Open Publishing Association. All rights reserved.; 40th International Conference on Logic Programming, ICLP 2024; Conference date: 14-10-2024 Through 17-10-2024.https://doi.org/10.4204/EPTCS.416.15.

Coppolillo, E., Calimeri, F., Manco, G., Perri, S. and Ricca, F. 2024. LLASP: Fine-tuning large language models for answer set programming. In *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning, KR 2024, Hanoi, Vietnam. November 2–8, 2024*. https://doi.org/10.24963/kr.2024/78.

Cropper, A., Dumancic, S., Evans, R. and Muggleton, S. H. 2022. Inductive logic programming at 30. *Machine Learning* 111, 1, 147–172. https://doi.org/10.1007/S10994-021-06089-1.

Ding, D., Yao, T., Luo, R. and Sun, X. 2025. Visual question answering in robotic surgery: A comprehensive review. *IEEE Access* 13, 9473–9484. https://doi.org/10.1109/ACCESS.2024.3525145.

Eiter, T., Higuera, N., Oetsch, J. and Pritz, M. 2022. A neuro-symbolic ASP pipeline for visual question answering. *Theory and Practice of Logic Programming* 22, 5, 739–754. https://doi.org/10.1017/S1471068422000229.

Eiter, T., Geibinger, T., Higuera, N. and Oetsch, J. 2023. A logic-based approach to contrastive explainability for neurosymbolic visual question answering. In *Proceedings of*

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

the 32nd International Joint Conference on Artificial Intelligence (IJCAI 2023), 3668–3676. https://doi.org/10.24963/IJCAI.2023/408.

Eiter, T., Hadl, J., Higuera, N., Lange, L., Oetsch, J., Scheuvens, B. and Strötgen, J. 2025. 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. To appear.

Faber, W., Leone, N. and Pfeifer, G. 2004. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In *Logics in Artificial Intelligence, 9th European Conference, JELIA 2004, Lisbon, Portugal, September 27–30, 2004, Proceedings*, J. J. Alferes and J. A. Leite, Eds. Vol. 3229 of Lecture Notes in Computer Science, Springer, 200–212. https://doi.org/10.1007/978-3-540-30227-8_19.

Gebser, M., Kaminski, R., Kaufmann, B. and Schaub, T. 2019. Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming* 19, 1, 27–82.

Gelfond, M. and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proceedings of the 5th International Conference and Symposium on Logic Programming*, MIT Press, 1070–1080.

Goyal, Y., Khot, T., Summers-Stay, D., Batra, D. and Parikh, D. 2017. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *2017 IEEE Conference On Computer Vision and Pattern Recognition (CVPR)*, 6325–6334. https://doi.org/10.1109/CVPR.2017.670.

Gutmann, B. and Kersting, K. 2006. TildeCRF: Conditional random fields for logical sequences. In *Proceedings of the 17th European Conference on Machine Learning (ECML 2006)*, Vol. 4212 of Lecture Notes in Computer Science, Springer, 174–185. https://doi.org/10.1007/11871842_20.

Hadl, J. 2023. GS-VQA: Zero-Shot Neural-symbolic Visual Question Answering with Vision-Language Models. Master's thesis, Technische Universität Wien, Wien. reposiTUm. https://doi.org/10.34726/hss.2023.109680.

Hudson, D. A. and Manning, C. D. 2019. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *2019 IEEE/CVF Conference On Computer Vision and Pattern Recognition (CVPR)*, Computer Vision Foundation/IEEE, 6700–6709. https://doi.org/10.1109/CVPR.2019.00686.

Ishay, A., Yang, Z. and Lee, J. 2023. Leveraging large language models to generate answer set programs. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Rhodes, Greece, September 2–8, 2023*, 374–383. https://doi.org/10.24963/KR.2023/37.

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L. and Girshick, R. B. 2017. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, IEEE Computer Society, 1988–1997. https://doi.org/10.1109/CVPR.2017.215.

Johnston, P., Nogueira, K. and Swingler, K. 2023. NS-IL: Neuro-symbolic visual question answering using incrementally learnt, independent probabilistic models for small sample sizes. *IEEE Access* 11, 141406–141420. https://doi.org/10.1109/ACCESS.2023.3341007.

Kareem, I., Gallagher, K., Borroto, M. A., Ricca, F. and Russo, A. 2024. Using learning from answer sets for robust question answering with LLM. In *Logic Programming and Nonmonotonic Reasoning - 17th International Conference, LPNMR 2024, Dallas, TX, USA, October 11–14, 2024*, C. Dodaro, G. Gupta and M. V. Martinez, Eds. Vol. 15245 of Lecture Notes in Computer Science, Springer, 112–125. https://doi.org/10.1007/978-3-031-74209-5_9.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L., Shamma, D. A., Bernstein, M. S. and Fei-Fei, L. 2017. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision* 123, 1, 32–73. https://doi.org/10.1007/s11263-016-0981-7.

Law, M., Russo, A., Bertino, E., Broda, K. and Lobo, J. 2020a. FastLAS: Scalable inductive logic programming incorporating domain-specific optimisation criteria. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7–12, 2020*, AAAI Press, 2877–2885. https://doi.org/10.1609/AAAI.V34I03.5678.

Law, M., Russo, A. and Broda, K. 2020b. The ILASP system for inductive learning of answer set programs. arXiv preprint arXiv:2005.00904. (Also published in: The Association for Logic Programming Newsletter, April 2020. Available at: https://logicprogramming.org/2020/04/the-ilasp-system-for-inductive-learning-of-answer-set-programs/).

Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S. and Scarcello, F. 2006. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic* 7, 3, 499–562. https://doi.org/10.1145/1149114.1149117.

Li, J., Li, D., Savarese, S. and Hoi, S. 2023. 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 2023)*, PMLR, vol. 202, pp. 19730–19742. https://proceedings.mlr.press/v202/li23q.html.

Liang, W., Niu, F., Reganti, A., Thattai, G. and Tur, G. 2020. LRTA: A Transparent Neural-Symbolic Reasoning Framework with Modular Supervision for Visual Question Answering. arXiv:2011.10731.

Lifschitz, V. 2019. *Answer Set Programming*. Springer.

Lin, Z., Zhang, D., Tao, Q., Shi, D., Haffari, G., Wu, Q., He, M. and Ge, Z. 2023. Medical visual question answering: A survey. *Artificial Intelligence in Medicine* 143, 102611. https://doi.org/10.1016/J.ARTMED.2023.102611.

Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T. and Raedt, L. D. 2021. Neural probabilistic logic programming in deepproblog. *Artificial Intelligence* 298, 103504.

Mao, J., Gan, C., Kohli, P., Tenenbaum, J. B. and Wu, J. 2019. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*, OpenReview.net.

Minderer, M., Gritsenko, A. A., Stone, A., Neumann, M., Weissenborn, D., Dosovitskiy, A., Mahendran, A., Arnab, A., Dehghani, M., Shen, Z., Wang, X., Zhai, X., Kipf, T. and Houlsby, N. 2022. Simple open-vocabulary object detection. In *Computer Vision - ECCV 2022 - 17th European Conference*, Vol. 13670 of Lecture Notes in Computer Science, Springer, 728–755. https://doi.org/10.1007/978-3-031-20080-9_42.

Muggleton, S. H. 1991. Inductive logic programming. *New Generation Computing* 8, 4, 295–318. https://doi.org/10.1007/BF03037089.

Muggleton, S. H. and Raedt, L. D. 1994. Inductive logic programming: Theory and methods. *The Journal of Logic Programming* 19–20, 629–679. https://doi.org/10.1016/0743-1066(94)90035-3.

Nguyen, Q., Pham, T., Vuong, T., Trinh, V. and Nguyen, H. 2025. Detecting misleading information with llms and explainable ASP. In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence, ICAART 2025 - Porto, Portugal, February 23–25, 2025*, A. P. Rocha, L. Steels and H. J. van den Herik, Eds. Vol. 3, SCITEPRESS, 1327–1334. https://doi.org/10.5220/0013357400003890.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G. and Sutskever, I. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, Vol. 139 of Proceedings of Machine Learning Research, PMLR, 8748–8763.

Raedt, L. D. and Kersting, K. 2017. Statistical relational learning. In *Encyclopedia of Machine Learning and Data Mining*. Springer, 1177–1187. https://doi.org/10.1007/978-1-4899-7687-1_786.

Rajasekharan, A., Zeng, Y., Padalkar, P. and Gupta, G. 2023. Reliable natural language understanding with large language models and answer set programming. In *Proceedings 39th International Conference on Logic Programming, ICLP 2023, Imperial College London, UK, 9th July 2023 - 15th July 2023*, Vol. 385 of EPTCS, 274–287. https://doi.org/10.4204/EPTCS.385.27.

Sabu, A. M. and Das, A. S. 2018. A survey on various optical character recognition techniques. In *2018 Conference on Emerging Devices and Smart Systems (ICEDSS)*, 152–155. https://doi.org/10.1109/ICEDSS.2018.8544323.

Santana, M. A. B., Kareem, I. and Ricca, F. 2024. Towards automatic composition of ASP programs from natural language specifications. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3–9, 2024*, ijcai.org, 6198–6206. https://www.ijcai.org/proceedings/2024/685.

Subramanian, S., Narasimhan, M., Khangaonkar, K., Yang, K., Nagrani, A., Schmid, C., Zeng, A., Darrell, T. and Klein, D. 2023. Modular visual question answering via code generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Association for Computational Linguistics, 747–761.

Surís, D., Menon, S. and Vondrick, C. 2023. ViperGPT: Visual inference via Python execution for reasoning. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023*, 11854–11864. https://doi.org/10.1109/ICCV51070.2023.01092.

Tiong, A. M. H., Li, J., Li, B., Savarese, S. and Hoi, S. C. H. 2022. Plug-and-play VQA: Zero-shot VQA by conjoining large pretrained models with zero training. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, Association for Computational Linguistics, 951–967.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., u. Kaiser, L. and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, Vol. 30, Curran Associates, Inc.

Wang, Z., Yu, J., Yu, A. W., Dai, Z., Tsvetkov, Y. and Cao, Y. 2022. SimVLM: Simple Visual Language Model Pretraining with Weak Supervision. In *Proceedings of the 10th International Conference on Learning Representations (ICLR 2022)*. OpenReview.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V. and Zhou, D. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022 (NeurIPS 2022)*.

Xu, J., Zhang, Z., Friedman, T., Liang, Y. and den Broeck, G. V. 2018. A semantic loss function for deep learning with symbolic knowledge. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, Vol. 80 of Proceedings of Machine Learning Research, PMLR, 5498–5507.

Yang, Y., Wu, J. and Yue, Y. 2025. Hypothesis generation via LLM-automated language bias for ILP. https://doi.org/10.48550/arXiv.2505.21486.

Yang, Z., Ishay, A. and Lee, J. 2020. NeurASP: Embracing neural networks into answer set programming. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, ijcai.org, 1755–1762. https://doi.org/10.24963/ijcai.2020/243.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684

Yi, K., Wu, J., Gan, C., Torralba, A., Kohli, P. and Tenenbaum, J. 2018. Neural-Symbolic VQA: Disentangling reasoning from vision and language understanding. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*, 1039–1050.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J. and Wen, J. 2026. A survey of large language models. *Frontiers of Computer Science*, 20, 12, article 2012627. https://doi.org/10.1007/s11704-026-60308-3.

Zhu, Z., Xue, Y., Chen, X., Zhou, D., Tang, J., Schuurmans, D. and Dai, H. 2023. Large language models can learn rules. arXiv preprint arXiv:2310.07064.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 15 Nov 2026 at 06:31:35, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1471068426100684