CLEVR-POC: Reasoning-Intensive Visual Question Answering in Partially Observable Environments

Savitha Sam Abraham[*], **Marjan Alirezaie**[†], **Luc De Raedt**[§]
[*]Australian Institute for Machine Learning, The University of Adelaide, Australia
savitha.samabraham@adelaide.edu.au

[†]Flybits Labs. TMU Creative AI Hub, Toronto, Canada
marjan.alirezaie@flybits.com

[§]Örebro University, Centre for Applied Autonomous Sensor Systems(AASS), Örebro, Sweden
Department of Computer Science, KULeuven, Belgium
luc.de-raedt@oru.se

Abstract

The integration of learning and reasoning is high on the research agenda in AI. Nevertheless, there is only a little attention to use existing background knowledge for reasoning about partially observed scenes to answer questions about the scene. Yet, we as humans use such knowledge frequently to infer plausible answers to visual questions (by eliminating all inconsistent ones). Such knowledge often comes in the form of constraints about objects and it tends to be highly domain or environment-specific. We contribute a novel benchmark called CLEVR-POC for reasoning-intensive visual question answering (VQA) in partially observable environments under constraints. In CLEVR-POC, knowledge in the form of logical constraints needs to be leveraged to generate *plausible answers* to questions about a hidden object in a given partial scene. For instance, if one has the knowledge that all cups are colored either red, green or blue and that there is only one green cup, it becomes possible to deduce the color of an occluded cup as either red or blue, provided that all other cups, including the green one, are observed. Through experiments, we observe that the low performance of pre-trained vision language models like CLIP ($\approx$ 22%) and a large language model (LLM) like GPT-4 ($\approx$ 46%) on CLEVR-POC ascertains the necessity for frameworks that can handle reasoning-intensive tasks where environment-specific background knowledge is available and crucial. Furthermore, our demonstration illustrates that a neuro-symbolic model, which integrates an LLM like GPT-4 with a visual perception network and a formal logical reasoner, exhibits exceptional performance on CLEVR-POC.

Keywords: LLM and Reasoning, visual question answering, partial observability, logical constraints

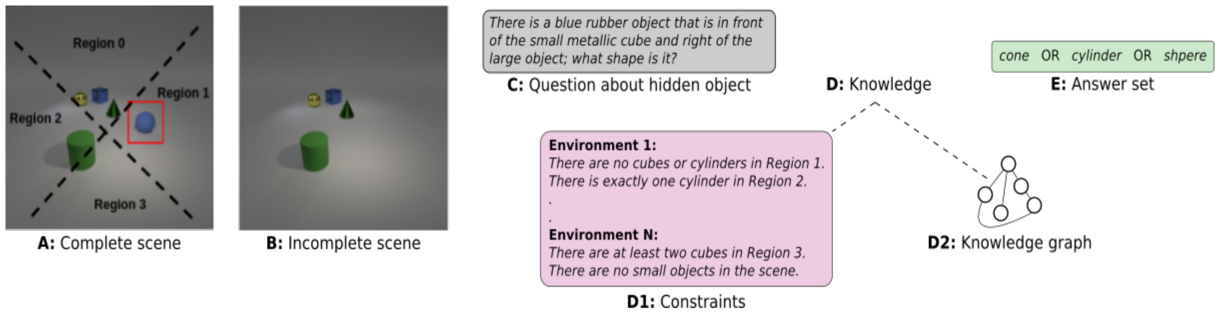
1. Introduction

Visual Question Answering (VQA) has been widely investigated by researchers from various subfields in AI like computer vision and natural language understanding. As a result, we now have access to a vast corpus of VQA datasets coupled with numerous models addressing the task of VQA (Zou and Xie, 2020; Wu et al., 2017).

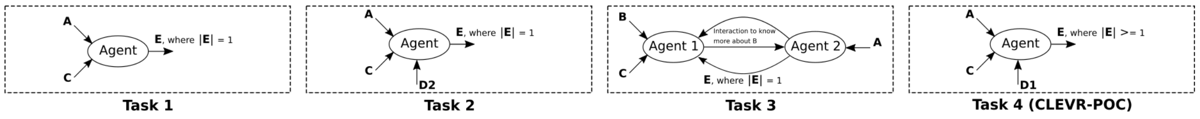
Most existing VQA datasets (Johnson et al., 2017; Antol et al., 2015) have a collection of images paired with questions such that all information required to answer the question is provided in the image, and hence the scene is considered complete. But in real life, we often engage in tasks where scenes may not be completely visible. We instead may have world knowledge about various locations visited by us, acquired over time, that allows us to generate plausible answers to queries about objects we do not see in a scene. For example, in autonomous vehicle scenarios, reasoning is crucial for dealing with partial observability. Comprehensive knowledge of traffic enables the system to interpret limited visual information and make informed decisions, ensuring safe navigation despite occlusions or limited field of view. Furthermore, in factory settings, reasoning combined with background knowledge about the environment can assist teams of robots in dealing with partial observability during navigation and other coordination and cooperation tasks.

In this paper, we introduce a synthetic dataset, CLEVR-POC[1], for a reasoning-intensive VQA task set in partially observable scenarios involving external knowledge in the form of constraints. The dataset consists of pairs of an image, representing a partial scene (**B** in Figure 1a) in some environment (**D1** in Figure 1a where the environment is defined by a set of constraints), and a question in natural language about some hidden/missing object (**C** in Figure 1a) in the scene. Although in the literature, there exist datasets for QA tasks in partially observable environments (e.g., CLEVR-dialog (Kottur et al., 2019), Visual Dialog (Das et al., 2017), Guess What? (De Vries et al., 2017)),

[1]The source code associated with this research project is openly accessible at `https://github.com/savithasam88/CLEVR-POC/tree/master`


Region 0
Region 1
Region 2
Region 3
A: Complete scene
B: Incomplete scene
There is a blue rubber object that is in front of the small metallic cube and right of the large object; what shape is it?
C: Question about hidden object
D: Knowledge
cone OR cylinder OR shpere
E: Answer set
Environment 1:
There are no cubes or cylinders in Region 1.
There is exactly one cylinder in Region 2.
Environment N:
There are at least two cubes in Region 3.
There are no small objects in the scene.
D1: Constraints
D2: Knowledge graph

(a) The different components in VQA tasks.



(b) The different VQA tasks are based on expected inputs and outputs and the number of agents involved.

Figure 1: VQA task components and types of VQA tasks

these do not come with additional background knowledge. The challenge presented in CLEVR-POC necessitates the integration of knowledge and multi-step reasoning involving eliminative induction, into perception systems driven by learning. Given that the knowledge associated with a scene typically varies depending on the specific environment involved, it is not a constant across the dataset. It becomes challenging for a learning system to simply memorize this knowledge during training iterations. Moreover, because this knowledge is environment-specific, employing Large Language Models (LLMs) such as GPT as the source of knowledge, as demonstrated in some of the recent works like (Zhou et al., 2023) and (Shah et al., 2023), does not yield favorable results. We substantiate these assertions through empirical experiments.
The contributions of this paper are as follows:

- We introduce a dataset, CLEVR-POC, that introduces the task of reasoning-intensive VQA - given a *partial scene*, the *constraints* (knowledge) to which the scene conforms and a *question* about a hidden object in the scene, find the set of all plausible answers.

- We evaluate the performance of state-of-the-art pre-trained vision language and large language models on CLEVR-POC.

- We demonstrate that the synergistic use of LLMs alongside a visual perception network and a formal reasoning system with access to external knowledge can efficiently and effectively address the challenges presented by CLEVR-POC.

The organization of the paper is as follows. Section 2 provides an overview of existing work in VQA, focusing on various VQA datasets and briefly discussing LLMs for reasoning. Section 3 delves into the detailed process of generating CLEVR-POC, while Section 4 outlines the research questions explored in this study. Additionally, this section presents the experiments conducted on CLEVR-POC and the corresponding results.

2. Related Work

In this section, we provide an overview of research in two domains - datasets in VQA and LLMs and reasoning.

2.1. Datasets in VQA

A VQA task may involve various combinations of the different components shown in Figure 1a - a complete scene (**A**), a partial scene (**B**), a question (**C**) about the scene, external knowledge in the form of rules/constraints (**D1**), or facts in knowledge graphs (**D2**), and the set of plausible answers to the question (**E**). Each combination results in a different VQA task (see Figure 1b).

2.2. Types of VQA Tasks

2.2.1. Task 1

Given a *complete scene*, and a *question* about an object in the scene, find the answer to the question. Since the scene is complete, the agent can come up with the exact answer implying that the solution set **E** has just one element (|**E**| = 1). DAQUAR (Malinowski and Fritz, 2014), VQA (Antol et al., 2015), CLEVR (Johnson et al., 2017) are datasets in this category.

2.2.2. Task 2

Given a *complete scene*, a *question* about one of the objects in the scene and external knowledge about objects (in the form of triples - **D2**), find the answer to the question leveraging this external knowledge. FVQA (fact-based VQA) (Wang et al., 2017), and KVQA (knowledge aware VQA) (Shah et al., 2019) are datasets in this category.

2.2.3. Task 3

While the above two tasks involve a single agent being posed with a scene and a question, this category of VQA tasks involves more than one agent. One of the agents has access to the complete scene while the other agent is provided with a partial scene and a question. Answering the question requires the agents to interact with each other. CLEVR-dialog (Kottur et al., 2019), Visual Dialog (Das et al., 2017), Guess What? (De Vries et al., 2017) are datasets handling Task 3.

2.2.4. Task 4 (CLEVR-POC)

Given a *partial scene*, *knowledge in the form of rules (constraints)* about the environment to which the scene conforms and a *question* about the hidden object in the scene, find the set of all plausible answers to the question. Since the question is about a hidden object (for example, about the shape of the object), it may not be always possible to provide an exact solution. Answering the question is more about eliminating all cases that are inconsistent with the background knowledge (for example: given the knowledge - *there are no spheres in this environment*) and returning all consistent answers as the solution (*the shape is a cone or a cylinder or a cube*, which is why $|\mathbf{E}| \geq 1$). In contrast to Task 2, where the knowledge graph encompasses general world facts (e.g.,"*cows are herbivores*"), the knowledge in this context is considerably more specific to an environment. While an LLM can be presumed to possess awareness of the former category of knowledge, the same cannot be said for the latter.

2.3. LLMs and Reasoning

In this paper, our emphasis lies on the process of reasoning which depends on a formal system grounded in logical rules and principles. Such a system ensures that all transformations or manipulations of symbols within it, leading to new inferences, adhere to the logical rules and principles governing the system (MacColl, 1897). While LLMs can also be seen as performing symbolic manipulations, these manipulations unlike traditional symbolic reasoning are based on statistical associations or patterns learned from data (Huang and Chang, 2023), because of which it may or may not be logically sound. Despite the progress in the development of large language models (LLMs), many still struggle with a deep understanding of symbols like humans do (Abraham and Alirezaie, 2022; Yao et al., 2022). To address this gap, there are ongoing efforts to create benchmarks (Huang and Chang, 2023), like the proposed CLEVR-POC, to evaluate the reasoning capabilities of LLMs.

In CLEVR-POC, we introduce a VQA task that involves constraint-based reasoning, a form of logical reasoning, where the generated response must satisfy a set of constraints given. These benchmarks are used to assess the capacity of language models in handling symbolic reasoning, contributing to the advancements in the development of more logically sound systems.

3. The CLEVR-POC Dataset

Now we describe in detail the generation of the CLEVR-POC dataset. The dataset, as the name suggests, is based on the CLEVR (Johnson et al., 2017) dataset, which generated scenes with geometrical shapes. Each object is associated with four attributes - color, shape, material, and size. The objects in CLEVR-POC can have one of the four shapes - cone, sphere, cylinder, and cube, three sizes - large, medium, and small, two materials - rubber and metal, and eight colors - red, blue, green, yellow, gray, brown and purple. Besides these four attributes, since a scene is divided into four regions (see Figure 1a), CLEVR-POC also associates an object with the region it is in - $0, 1, 2$ or 3. Each object belongs to exactly one region. Division of a scene into regions enables the specification of constraints at multiple levels.

- **Region-based** constraints - for example, all objects in Region 0 are of shapes cube or cylinder. These constraints must be satisfied by objects in the corresponding region.
- **Across-region** constraints - for example, the total number of objects sharing the same color in regions 1 and 2 is not more than 2. These are constraints specified across two regions.
- **Generic** constraints - for example, there are at least two cubes in the scene. These constraints apply to the whole scene.

One of the major points of distinction in the scene generation process of CLEVR-POC from the original CLEVR is that the scenes in CLEVR-POC are generated such that they conform to a chosen set of constraints. The steps in creating an instance i in the dataset are:

- Generating an environment - $Environment_i$, defined by a set of constraints.
- Generating a complete scene graph, $Complete_i$, that conforms to $Environment_i$.
- Generating the partial scene graph, $Partial_i$ by removing one of the objects, Obj_i, from $Complete_i$.

Template-1 (Value Restriction)
:- object(X),at(X, R'), not hasProperty(X, P1', V1').
Translation All objects at region *R'* have value *V1'* for the property *P1'*.
An instantiation :- object(X),at(X, 0), not hasProperty(X, color, red).
Template-2 (Negation Constraint)
:- object(X), at(X, R'), hasProperty(X, P1', V1').
Translation All objects at region *R'* cannot have value *V1'* for the property *P1'*.
An instantiation :- object(X), at(X, 0), hasProperty(X, material, metal).
Template-3 (Exactly N Constraint)
:- #count{X: hasProperty(X, P1', V1'), object(X), at(X, R')}!=N'
Translation There are exactly *N'* objects at region *R'* with value *V1'* for the property *P1'*.
An instantiation :- #count{X: hasProperty(X, size, small), object(X), at(X, R')}!=2
Template-4 (Atleast N Constraint)
:- #count{X1, X2: sameProperty(X1, X2, P1'), object(X1), object(X2), at(X1, R1'), at(X2, R2')} $<$ N'.
Translation There are at least N' pairs of objects at regions *R1'* and *R2'* that has the same value *V1'* for the property *P1'*.
An instantiation :- #count{X1, X2: sameProperty(X1, X2, shape), object(X1), object(X2), at(X1, 1), at(X2, 2)}$<$1.
Template-5 (OR Constraint)
:- object(X), at(X, R'), not hasProperty(X, P1', V1'), not hasProperty(X, P1', V2').
Translation All objects in region *R'* have value *V1'* for property *P1'* or *V2'* for property *P2'*.
An instantiation :- object(X), at(X, 1), not hasProperty(X, color, yellow), not hasProperty(X, color, blue).

Table 1: A few constraint templates

- Generating a question, Q_i, about the partial scene with object of interest Obj_i.

3.1. Environment Representation

An environment in CLEVR-POC is defined by a set of constraints. We provide a set of 11 constraint templates with CLEVR-POC that are expressed in answer set programming (ASP)[2]. Each environment is created by at most 15 different instantiations of these templates, provided there are at least two constraints associated with each region. A few example constraint templates with their translation in English and an instantiation are shown in Table 1. Around 30 different environments are generated (see Appendix A for an example) and the scenes in the dataset belong to one of these 30 environments - the dataset generation process ensures that the scenes are uniformly distributed across the 30 environments.

3.2. Scene Representation

CLEVR represented a scene in the form of a scene graph whose nodes represented objects annotated with its attributes and edges denoted the spatial relations (left, right, front, behind) between objects. In CLEVR-POC, besides the scene graph representation, we also represent a scene in ASP. Below we show part of the ASP representation of the partial scene in Figure 1a.

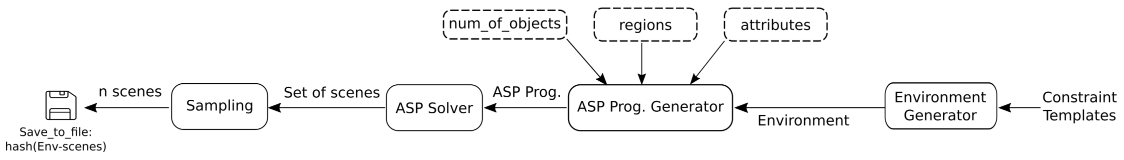
```
%Objects in the scene
object(0). object(1). object(2). object(3).

%Attributes of objects
at(0, 2).
hasProperty(0, color, green).
hasProperty(0, size, large).
hasProperty(0, material, rubber).
hasProperty(0, shape, cylinder).
....
%Spatial relations between objects
front(1, 0).   right(1, 0). ...
```

The predicate `object` is used to define the different objects (denoted using identifiers - 0, 1, ..). `hasProperty(Object, Attribute, Value)` associates a `Value` for an `Attribute` of an `Object`. `at(Object, Region)` represents the region where the object is located. The spatial relations between objects are represented with predicates `left, right, front, behind` - for example, `left(Object1, Object2)` represents that `Object2` is located left of `Object1`.

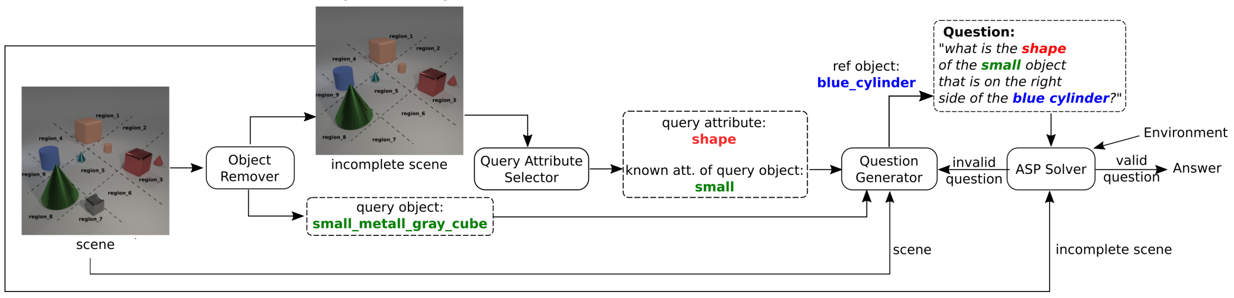
3.3. Image Generation

While the images in CLEVR are generated from a randomly sampled scene graph, CLEVR-POC generates its images from scene graphs known to adhere to constraints defining an environment. Scene graph creation is thus a reasoning problem - given an environment (constraints in ASP) and a desired number of objects (*n*) in the scene, the goal is to assign each object to one of the four regions and propose values to color, size, shape, and material that are consistent with the

[2]ASP is a declarative programming paradigm applied to solve complex search problems (Lifschitz, 2008)



(a) Pipeline for generating *environment* and *complete scenes* in that environment.


scene
Object Remover
incomplete scene
Query Attribute Selector
query attribute: shape
known att. of query object: small
query object: small_metall_gray_cube
ref object: blue_cylinder
Question: "what is the shape of the small object that is on the right side of the blue cylinder?"
Question Generator
invalid question
ASP Solver
valid question
Answer
Environment
scene
incomplete scene

(b) Pipeline for generating partial scenes, and questions and then labeling them with answers.

Figure 2: Two steps in dataset generation process: Figure 2a shows the first step - environment generation from constraint templates and generating complete scenes satisfying these constraints. Figure 2b shows Step 2 - partial scene and question generation from a complete scene.

constraints in the environment. An ASP reasoning engine solves this problem - each answer (a consistent property-value assignment for the *n* objects) in the answer set returned is a scene graph or a possible configuration of the objects in the scene. Since there are many possible configurations - we randomly sample a million of these scene graphs for the subsequent image generation phase. A scene graph is then rendered using Blender[3]. The image representing the partial scene is generated from a partial scene graph constructed from the actual scene graph by randomly removing one of the objects from it. Figure 2a shows the scene graph construction process.

3.4. Question Representation

The questions in CLEVR-POC query about one of the four attributes - color, size, shape, and material of the missing/hidden object in the partial scene. Besides representing the questions using an equivalent functional program as in CLEVR, CLEVR-POC also represents it in ASP. An example question and its ASP form are shown below:

Question:
What is the color of the other cylinder that is the same material as the medium red thing?

```
query(Q):- hasProperty(X,color,Q),
           hasProperty(X,shape,cylinder),
           hasProperty(Y,size,medium),
           hasProperty(Y,color,red),
           same_material(Y,X),
           X!=Y.
```

If the query is about attribute A, $A \in \{color, size, material, shape\}$, the questions are generated such that the cardinality of the set of possible solutions (S) is $1 \leq |S| < |A|$, where $|A|$ is the set of all values for the attribute A (for example $|size|$ = 3 = $|\{large, medium, small\}|$). If the question generated has $|A|$ solutions (for instance, a solution like, '*size is large or small or medium*' is true for any question), it is considered invalid. The questions are balanced across the question types (that depend on the query attribute - see Appendix B for the distribution). It should be noted that the solution space of CLEVR-POC questions is 16 times that of CLEVR as the solutions expected are not always a single value, but a set of values.

[3]https://www.blender.org/

3.5. Question Generation

The question in CLEVR-POC is generated from the question templates available in CLEVR. We avoid the yes/no (existence, comparison) and counting questions and focus on just the attribute querying templates. An example template is as follows:

What shape is the $<\ Z2\ >\ (size)\ <\ C2\ >\ (color)\ <\ M2\ >\ (material)$ [that is] $<\ R\ >\ (relation)$ the $<\ Z\ >\ (size)\ <\ C\ >\ (color)\ <\ M\ >\ (material)\ <\ S\ >\ (shape)$?

Question template instantiation is done based on the complete scene graph of the associated image. The object of interest is always the object that is removed from the complete scene to generate the partial scene graph. The query attribute is picked such that it satisfies the question type balancing requirements. The known attributes of the query object (filling the slots $<\ Z2\ >$ or $<\ C2\ >$ or $<\ M2\ >$ in the above template) are randomly selected. While the filler for the slot $<\ R\ >$ (one

of the left, right, front, behind) is randomly picked, the reference object in the query is picked based on the spatial relations available in the complete scene - picking one of the objects that are in $< R >$ relation of the query object.
The ASP representations of the question, the incomplete scene, and the constraints in the environment are given to an ASP solver to identify the set of possible values for the query attribute. Figure 2b shows the pipeline of question generation. Refer to Appendix A and B for a detailed example and statistics of CLEVR-POC.

4. Experiments

The experiments are designed to answer the following research questions (RQ):

- **RQ1**: How do neural-based vision language models perform on reasoning-intensive VQA tasks (with an emphasis on symbolic knowledge representation and reasoning)?
- **RQ2**: How well do neuro-symbolic vision language architectures handle reasoning-intensive VQA tasks (in the context of mapping raw inputs to symbolic space)?
- **RQ3**: How can we leverage LLMs in reasoning-intensive VQA tasks and what are the challenges associated with it?

In the sections following, we describe the methods implemented to answer these questions.

4.1. Methods

4.1.1. CLIP-based model

CLIP (Contrastive Language Image Pre-training) (Radford et al., 2021) is a vision-language model that is trained to align pairs of text and images to a unified space. We experimented with the CLIP model to investigate RQ1. Figure 3 shows the architecture of a CLIP-based model to solve CLEVR-POC. The pre-trained vision transformer (ViT-B/32) and the text encoders (masked self-attention) in CLIP are leveraged to obtain encodings for the incomplete scene and the question. The encoding for the environment is obtained from its constraints. A pre-trained GPT-2 (Radford et al., 2019) model is used to encode the constraints. As GPT-2 is more language-oriented, we input the natural language version of ASP constraints (while experimenting with ASP-form constraints to assess their impact on performance).
The problem is formulated as a multi-label classification problem where the output is one or more of the following 17 labels - {*red, blue, green, yellow, cyan, brown, gray, purple, rubber, metal, large, small, medium, cone, cube, cylinder, sphere*}. Hence, the three encodings are passed to a multi-label classifier (feed-forward network) which is the only module of the whole model that is trained from scratch. The classifier is trained with a weighted binary cross entropy loss function (Ho and Wookey, 2019) that gives more penalty to the wrong prediction of minority class (as most of the labels in the output are 0, except for the ones in the answer - a false negative is given more penalty). For each of the 17 labels, the weighted cross entropy loss is thus defined as below:

$$WCE\,(y, \hat{y}) = -\,(\beta y \log(\hat{y}) + (1 - y)\log(1 - \hat{y})) \quad (1)$$

β is the weight (is set > 1 to penalize false negatives)[4], y is the ground truth, $\hat{y}$ is the prediction.

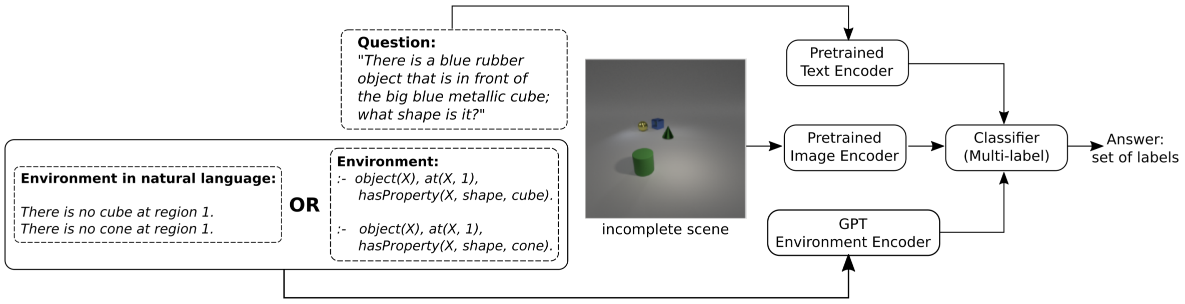
4.1.2. Neuro-Symbolic Visual Question Answering

The architecture for the neuro-symbolic approach to solving CLEVR-POC task is shown in Figure 4. The idea is to convert both the image and the question into a unified space as in CLIP, with the difference that this space is symbolic (scene graph and question in ASP). The architecture is based on the state-of-the-art neuro-symbolic approach on the CLEVR dataset, NS-VQA (Yi et al., 2018) and will be used here to study aspects of RQ2. We modify this architecture to include an ASP solver that takes as input - the scene in ASP, the question in ASP, and the environment constraints in ASP to derive the answer to the question.
The question parser, (a Bidirectional Long Short Term Memory (BiLSTM) sequence to sequence model) is trained as in NS-VQA using REINFORCE - the reward is positive if the ASP program generated by the parser results in the correct answer, else it is 0. The question parser is initially pre-trained in a fully supervised way with a small sample of (question, ASP program) pairs.
The image perception network in NS-VQA is based on Detectron (Girshick et al., 2018) and it was trained independently of the question parser in a supervised way. The ASP solver used is the same as the one used during the dataset generation phase.

4.1.3. LLMs for solving CLEVR-POC

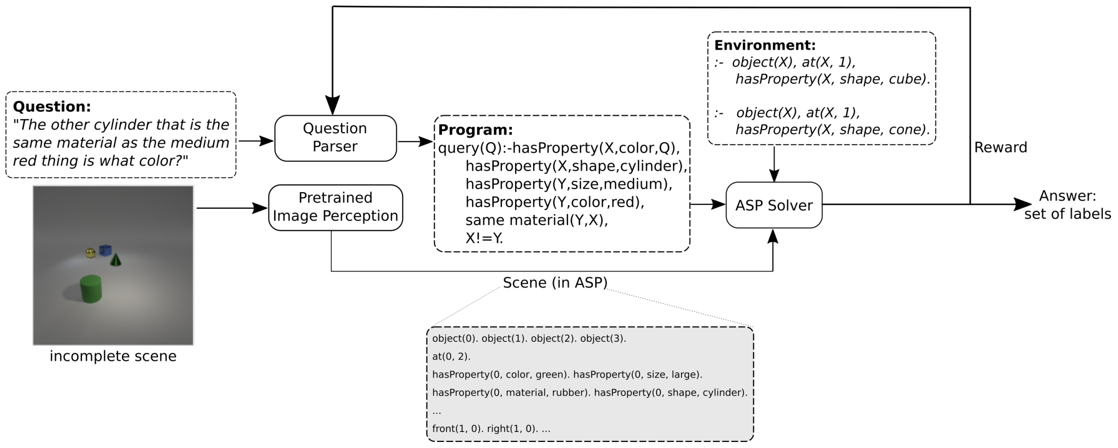
LLMs are leveraged in two ways for solving a reasoning task like CLEVR-POC.
LLM as question parser in NS-VQA: In this approach, we use LLM as a question parser - converting the question into a semantic representation like ASP. The image is converted to a scene graph as done in NS-VQA. Both semantic representations are then passed on to a formal reasoner like an ASP solver to derive solutions consistent with the constraints.
Stand-alone LLM: The second approach is to provide both the image description and the question

[4] The results in Section 4.3 are for $\beta = 5$.



Question:
"There is a blue rubber object that is in front of the big blue metallic cube; what shape is it?"
Environment in natural language:
There is no cube at region 1.
There is no cone at region 1.
OR
Environment:
:- object(X), at(X, 1), hasProperty(X, shape, cube).
:- object(X), at(X, 1), hasProperty(X, shape, cone).
incomplete scene
Pretrained Text Encoder
Pretrained Image Encoder
GPT Environment Encoder
Classifier (Multi-label)
Answer: set of labels

Figure 3: CLIP for CLEVR-POC



Question:
"The other cylinder that is the same material as the medium red thing is what color?"
Question Parser
Pretrained Image Perception
incomplete scene
Program:
query(Q):-hasProperty(X,color,Q), hasProperty(X,shape,cylinder), hasProperty(Y,size,medium), hasProperty(Y,color,red), same material(Y,X), X!=Y.
Environment:
:- object(X), at(X, 1), hasProperty(X, shape, cube).
:- object(X), at(X, 1), hasProperty(X, shape, cone).
ASP Solver
Reward
Answer: set of labels
Scene (in ASP)
object(0). object(1). object(2). object(3).
at(0, 2).
hasProperty(0, color, green). hasProperty(0, size, large).
hasProperty(0, material, rubber). hasProperty(0, shape, cylinder).
...
front(1, 0). right(1, 0). ...

Figure 4: NS-VQA for CLEVR-POC - architecture is updated with an ASP solver

along with the constraints (in NL) as input to LLM and generate as a response the consistent solutions. We, here, assume as done in NS-VQA that the scene graphs are accurate, as our focus is on evaluating LLMs' ability to perform symbolic reasoning. CLEVR-POC, a synthetic dataset where environment-specific knowledge is not fixed, can assess LLMs' symbolic reasoning ability without data contamination (where the dataset becomes unusable once it has been exploited).

The LLM used in the experiments is GPT-4 (OpenAI, 2023) (See Appendix C for details about prompts used).

4.2. Evaluation

Let A = {a1, a2,..} denote the set of values in the actual answer and P = {p1, p2,..} denote the predicted answer set. We evaluate the performance of the two approaches on CLEVR-POC using the two metrics based on accuracy.

Exact Accuracy checks whether the prediction made is exactly accurate, i.e., A is exactly equal to P.

$$Exact\_Accuracy(A, P) = \begin{cases} 1 & \text{if } x \in A \iff x \in P \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Jaccard Index computes the similarity between the actual answer and predicted answer sets as:

$$Jaccard\_Index\,(A, P) = \frac{|A \cap P|}{|A \cup P|} \tag{3}$$

The value of Jaccard_Index is between 0 (no common elements) and 1 (exact match). It gives some credit for partially correct answers as well.

4.3. Results

Tables 2a and 2b show the results for exact and partial answer accuracies respectively for NS-VQA, CLIP-based models, and stand-alone GPT-4 on CLEVR-POC. While NS-VQA (BiLSTM) uses a BiLSTM trained from scratch as the question parser, NS-VQA (GPT-4) uses pre-trained GPT-4 as the question parser. We experimented with varying dataset sizes - 2000, 6000, and then 12000 instances. [5] It can be seen that with a multifold increase in the dataset size, there is an improvement in the answer accuracy, but the performance is not satisfactory.

RQ1 - CLIP-based model analysis: Since the question is not about some object in the scene, and the set of constraints to be satisfied is also not fixed across the instances in the dataset, it is challenging to learn a mapping from the three inputs

[5]The models are trained on Intel® CoreTM i7-12700K, 32GB RAM, and an NVIDIA GeForce RTX 3080 Ti for training.

Dataset	NS-VQA (BiLSTM)	NS-VQA (GPT-4)	CLIP-ASP	CLIP-NL	CLIP (no knowledge)	GPT-4
2000	0.0200	**0.9250**	0.0350	0.0600	0.0500	0.4626
6000	0.1516	**0.9550**	0.1500	0.1700	0.1183	-
12000	0.2308	**0.9441**	0.1800	0.2283	0.1483	-

(a) Exact answer accuracies of CLIP, NS-VQA and GPT-4 models on CLEVR-POC.

Dataset	NS-VQA (Bi-LSTM)	NS-VQA (GPT-4)	CLIP-ASP	CLIP-NL	CLIP (no knowledge)	GPT-4
2000	0.0591	**0.9287**	0.1000	0.1557	0.1412	0.5164
6000	0.3602	**0.9578**	0.3100	0.3403	0.2447	-
12000	0.4331	**0.9496**	0.3600	0.4465	0.2912	-

(b) Jaccard Index of CLIP, NS-VQA and GPT-4 models on CLEVR-POC

Table 2: Exact accuracies and Jaccard index scores of NS-VQA with BiLSTM and GPT-4 as question parsers, CLIP and GPT-4 on CLEVR-POC. CLIP-NL and CLIP-ASP take constraints in natural language and ASP, respectively. CLIP (no knowledge) is the performance of CLIP without constraints.

Sample Size	PA (after pre-training)	PA (after REINFORCE)	PA (GPT-4)
28 (prompt size)	-	-	**0.9250**
$\approx$ 200	0.0512	0	-
$\approx$ 1000	0.4487	0.0366	-
$\approx$ 2000	0.5043	0	-

Table 3: Drop of program accuracies (PA) after REINFORCE and the performance of GPT-4 provided with just 28 (question, ASP program) pairs as prompt.

(the incomplete scene, the natural language question, and the constraints) to the output set of plausible values. Table 2 shows three sets of results for CLIP. The columns CLIP-NL and CLIP-ASP correspond to instances of CLIP where the constraints are given in natural language and ASP respectively. It should be noted that CLIP-NL performs better than CLIP-ASP, suggesting that representing symbolic knowledge in natural language may be ideal while incorporating knowledge into neural frameworks for QA. The performance of CLIP on CLEVR-POC when no external knowledge is provided is shown in the column CLIP (no knowledge). Although without the external knowledge CLIP's performance drops, there is not much of a difference indicating that we need to consider better techniques for incorporating such symbolic constraints into neural frameworks. This points us toward existing neuro-symbolic frameworks.

RQ2 - NS-VQA analysis: While neural models failed in symbolic reasoning and incorporating symbolic knowledge into the network, it can be seen that the major challenge faced by neuro-symbolic architectures lies not in reasoning but in mapping image or question to a symbolic representation in the absence of ground truth semantic representations. In our experiments, we focus on language perception while assuming 100% accuracy in image perception. Tackling both perceptions simultaneously is even more formidable without access to ground truth representations. Hence, the poor performance of NS-VQA (see column NS-VQA (BiLSTM) in Tables 2a and 2b) can be solely attributed to the failure of REINFORCE in learning accurate ASP programs. As mentioned in Section 4.1.2, a BiLSTM is initially pre-trained in a supervised fashion with a few examples. We experimented by varying the number of examples provided for pre-training. Table 3 shows the program accuarcy after pre-training with around 200, 1000 and 2000 pairs of $<$question, ASP program$>$. When these pre-trained models are further trained with REINFORCE, there is a drastic drop in the program accuracy as the focus of the REINFORCE algorithm is on coming up with the correct answer independent of the program's accuracy. This fall is observed even with the original CLEVR dataset. The chances of deriving the correct answer even with a wrong program by a fluke are higher in the case of CLEVR compared to CLEVR-POC considering the larger solution space of CLEVR-POC (see Section 3.4). REINFORCE clearly fails to learn ASP programs through weak supervision even when it initiates its training from a proficient model.

RQ3 - LLM Analysis: In the first experiment we used GPT-4 as a question parser. The BiLSTM-based question parser of NS-VQA is replaced with GPT-4 (the results are shown in column NS-VQA(GPT-4) in Tables 2a and 2b). The model is provided with just 28 (question, ASP program) pairs of examples as prompts. GPT-4 with no fine tuning was able to accurately predict the equivalent ASP programs.

The stand-alone GPT-4 approach gave less than 50% exact accuracy. The evidence indicates that employing GPT-4 as a question parser to translate the question into an ASP program and subsequently utilizing an ASP reasoning engine leads to better results compared to placing the entire burden of symbolic reasoning on GPT-4. It should also be noted that GPT-4 with no data-specific training performed better than CLIP and NS-VQA (BiLSTM). There is still room for improvement with some fine-tuning.

5. Discussion

We now discuss important challenges that our dataset and work point to.

Reasoning and LLM: The experiments showed that the direct application of LLMs is not a good solution for such reasoning-intensive tasks. (Mahowald et al., 2023) also discusses the limitations of LLMs in formal reasoning tasks. Our experiments showed that a more appropriate approach to harnessing LLMs involved relieving them of the task of symbolic reasoning and instead employing them for generating symbolic representations. Progressing even further entails discovering mechanisms for seamlessly incorporating specific knowledge into LLMs and generating responses that are consistent with this knowledge.

Symbolic knowledge in visual perception network: Although the focus of this paper was on language and reasoning, it may be noted that knowledge in the form of constraints in CLEVR-POC can play a significant role during image perception as it can provide hints on what can or cannot be in the image. This is a form of weak supervision which is also required in the absence of ground truth scene graphs to accelerate the learning process. Developing neuro-symbolic models with a stronger feedback mechanism for visual perception, such as DeepProbLog (Manhaeve et al., 2018), NeurASP (Yang et al., 2020), Semantic-Loss (Xu et al., 2018) and LTN (Serafini and Garcez, 2016)), would help in faster convergence. The aforementioned frameworks, however, cannot still be applied to VQA tasks due to scalability issues.

6. Conclusion

Humans often have to interact with the partially observable environment. In light of the need to deal with the inherent uncertainty in knowledge-rich real-world scenarios, this work aimed to establish a benchmark for evaluating reasoning-intensive VQA in partially observable environments. Applying the benchmark to stand-alone LLMs and other vision-language models yielded disappointing results due to their inability to perform symbolic reasoning. We also demonstrated that combining LLM with a visual perception network and a formal reasoner produced positive results.

Future directions involve developing visual perception networks with knowledge-guided supervision, enhancing LLMs' reasoning capabilities, and moving CLEVR-POC to an embodied setup like vision language navigation.

7. Acknowledgements

This research was conducted during the authors' tenure at Örebro University, Sweden and was financially supported by the Wallenberg AI, Autonomous Systems, and Software Program (WASP).

8. Bibliographical References

Savitha Sam Abraham and Marjan Alirezaie. 2022. Compositional generalization and neuro-symbolic architectures. In *Combining Learning and Reasoning: Programming Languages, Formalisms, and Representations*.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *IEEE international conference on computer vision*, pages 2425–2433.

O Blender. 2018. Blender—a 3d modelling and rendering package. *Retrieved. represents the sequence of Constructs1 to*, 4.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Embodied question answering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–10.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. 2017. Visual dialog. In *IEEE conference on computer vision and pattern recognition*, pages 326–335.

Harm De Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron Courville. 2017. Guess what?! visual object discovery through multi-modal dialogue. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5503–5512.

Artur d'Avila Garcez, Marco Gori, Luis C Lamb, Luciano Serafini, Michael Spranger, and Son N Tran. 2019. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *arXiv preprint arXiv:1905.06088*.

Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. 2018. Detectron.

Yaoshiang Ho and Samuel Wookey. 2019. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE access*, 8:4806–4813.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada. Association for Computational Linguistics.

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

Incheol Kim. 2020. Visual experience-based question answering with complex multimodal environments. *Mathematical Problems in Engineering*, 2020.

Satwik Kottur, José MF Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. 2019. Clevr-dialog: A diagnostic dataset for multi-round reasoning in visual dialog. *arXiv preprint arXiv:1903.03166*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *IEEE*, 86(11):2278–2324.

Vladimir Lifschitz. 2008. What is answer set programming? In *23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, page 1594–1597. AAAI Press.

Hugh MacColl. 1897. Symbolic reasoning. *Mind*, 6(24):493–510.

Kyle Mahowald, Anna A Ivanova, Idan A Blank, Nancy Kanwisher, Joshua B Tenenbaum, and Evelina Fedorenko. 2023. Dissociating language and thought in large language models: a cognitive perspective. *arXiv preprint arXiv:2301.06627*.

Mateusz Malinowski and Mario Fritz. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. *Advances in neural information processing systems*, 27.

Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. 2018. Deepproblog: Neural probabilistic logic programming. *advances in neural information processing systems*, 31.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. 2019. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*.

Giuseppe Marra, Sebastijan Dumančić, Robin Manhaeve, and Luc De Raedt. 2021. From statistical relational to neural symbolic artificial intelligence: a survey. *arXiv preprint arXiv:2108.11451*.

Shoya Matsumori, Kosuke Shingyouchi, Yuki Abe, Yosuke Fukuchi, Komei Sugiura, and Michita Imai. 2021. Unified questioner transformer for descriptive question generation in goal-oriented visual dialogue. In *IEEE/CVF International Conference on Computer Vision*, pages 1898–1907.

Rui Da Silva Neves, Jean-François Bonnefon, and Eric Raufaste. 2000. Rationality in human non-monotonic inference.

OpenAI. 2023. Gpt-4 technical report.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Luciano Serafini and Artur d'Avila Garcez. 2016. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. *arXiv preprint arXiv:1606.04422*.

Dhruv Shah, Michael Robert Equi, Błażej Osiński, Fei Xia, Sergey Levine, et al. 2023. Navigation with large language models: Semantic guesswork as a heuristic for planning. In *7th Annual Conference on Robot Learning*.

Sanket Shah, Anand Mishra, Naganand Yadati, and Partha Pratim Talukdar. 2019. Kvqa: Knowledge-aware visual question answering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):8876–8884.

Peng Wang, Qi Wu, Chunhua Shen, Anthony Dick, and Anton Van Den Hengel. 2017. Fvqa: Fact-based visual question answering. *IEEE transactions on pattern analysis and machine intelligence*, 40(10):2413–2427.

Qi Wu, Damien Teney, Peng Wang, Chunhua Shen, Anthony Dick, and Anton Van Den Hengel. 2017. Visual question answering: A survey of methods and datasets. *Computer Vision and Image Understanding*, 163:21–40.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Broeck. 2018. A semantic loss function for deep learning with symbolic knowledge. In *International conference on machine learning*, pages 5502–5511. PMLR.

Zhun Yang, Adam Ishay, and Joohyung Lee. 2020. Neurasp: Embracing neural networks into answer set programming. In *29th International Joint Conference on Artificial Intelligence (IJCAI 2020)*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. 2018. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *Advances in neural information processing systems*, 31.

Gengze Zhou, Yicong Hong, and Qi Wu. 2023. Navgpt: Explicit reasoning in vision-and-language navigation with large language models. *arXiv preprint arXiv:2305.16986*.

Yeyun Zou and Qiyu Xie. 2020. A survey on vqa: Datasets and approaches. In *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*, pages 289–297. IEEE.

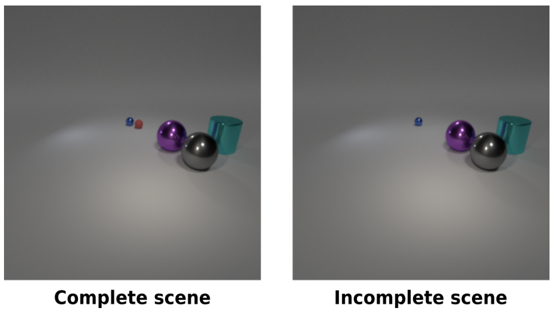


Figure 5: A complete and incomplete scene from CLEVR-POC

A. An example from CLEVR-POC

Complete and incomplete scene: Figure 5 is an example of a complete scene and the incomplete scene generated from it by hiding the *small red rubber sphere*.

Environment Every scene is generated such that it satisfies the constraints in an environment. The following are the general rules shared by all environments in CLEVR-POC.

```
 1. property(color, gray). property(color, red).
 2. property(color, blue). property(color, green).
 3. property(color, brown). property(color, purple).
 4. property(color, cyan). property(color, yellow).
 5. property(shape, cube). property(shape, cylinder).
 6. property(shape, sphere). property(shape, cone).
 7. property(size, small). property(size, medium).
 8. property(size, large).
 9. property(material, rubber).
    property(material, metal).
10. region(0). region(1). region(2). region(3).
11. right_R(0, 0). right_R(0, 1). right_R(0, 2).
    right_R(0, 3).
12. right_R(1, 1). right_R(1, 3).
13. right_R(2, 0). right_R(2, 1). right_R(2, 2).
    right_R(2, 3).
14. right_R(3, 1). right_R(3, 3).
15. left_R(R1, R2) :- right_R(R2, R1).
16. front_R(0, 0). front_R(0, 1). front_R(0, 2).
    front_R(0, 3).
17. front_R(1, 0). front_R(1, 1). front_R(1, 2).
    front_R(1, 3).
18. front_R(2, 2). front_R(2, 3).
19. front_R(3, 2). front_R(3, 3).
20. behind_R(R1, R2) :- front_R(R2, R1).
21. sameProperty(X1, X2, P) :- hasProperty(X1,P,V),}
22. hasProperty(X2,P,V), X1!=X2.
23. same_color(X,Y):- sameProperty(X, Y, color).
24. same_size(X,Y):- sameProperty(X, Y, size).
25. same_shape(X,Y):- sameProperty(X, Y, shape).
26. same_material(X,Y):- sameProperty(X, Y, material).
27. 1{hasProperty(X, color, V) :
28.           property(color, V)}1 :-  object(X).
29. 1{hasProperty(X, material, V) :
30.           property(material, V)}1 :- object(X).
31. 1{hasProperty(X, shape, V) :
32.           property(shape, V)}1 :- object(X).
33. 1{hasProperty(X, size, V) :
34.           property(size, V)}1 :- object(X).
35.1{at(X, R): region(R)}1 :- object(X).
36.:- sameProperty(X1, X2, color),
37.   sameProperty(X1, X2, material),
38.   sameProperty(X1, X2, size)},
39.   sameProperty(X1, X2, shape),
40.   object(X1), object(X2), X1!=X2.
41.exceed_region_capacity(R) :-
42.   #count{X: object(X), at(X, R)} >= 4, region(R).
43.:- exceed_region_capacity(_).
```

Environment's general rules in natural language:

```
1-9. Objects must have 4 properties. They are color,
shape, size, and material.

1-4. Objects can be in one of the 8 colors. It can
be gray, or red, or blue, or green, or brown,
or purple, or cyan,  or yellow.

5-6. Objects can be in one of the 4 shapes.
It can be cube, or a cylinder, or a sphere or cone.

7-8. Objects can be in one of the 3 sizes.
It can be small, medium, or large.

9. Objects can be in one of the 2 materials.
It can be rubber or metal.

10. The scene is divided into 4 regions.
They are named 0, 1, 2, 3.

11. If there are two objects, the first object is
located  in region 0 and the second object is to
the right of the  first object, then the location
of the second object is either in region 0, 1, or
2, or 3.

12. If there are two objects, the first object is
located in region 1 and the second object is to
the right of the first object, then the location
of the second object is either in region 1, or 3.

13. If there are two objects, the first object is
located in region 2 and the second object is to
the right of the first object, then the location
of the second object is either in region 0, 1, 2, or 3.

14. If there are two objects, the first object is
located in region 3 and the second object is to
the right of the first object, then the location
of the second object is either in region 1, or 3.

15. If there are two objects, the first object is
to the right of the second object, then the second
the object is to the left of the first object.

16. If there are two objects, the first object is
located in region 0 and the second object is in
front of the first object, then the location of
the second  object is either  in region 0, 1, or
2, or 3.

17. If there are two objects, the first object is
located in region 1 and the second object is in
front of the first object, then the location of
the second object is either in region 0, 1, or
2, or 3.

18. If there are two objects, the first object is
located in region 2 and the second object is in
front of the first object, then the location of
second object is either in region 2, or 3.

19. If there are two objects, the first object is
located in region 3 and the second object is in
front of the first object, then the location of
the second object is either in region 2, or 3.

20. If there are two objects, the first object is
in front of the second object, then the second
the object is behind the first object.

27-28. Every object must be assigned exactly one
value for color.

29-30. Every object must be assigned exactly one
value for the material.

31-32. Every object must be assigned exactly one
value for shape.

33-34. Every object must be assigned exactly one
value for size.

35. Every object must be assigned exactly one value
for region.

36-40. Two different objects cannot have the same values
```

```
for all the 4 properties.

41-43. Every region can have at most 3 objects.
```

The following constraints in ASP represent the specific environment to which the scene in Figure 5 belongs.

```
44. object(0..4).
45. :- object(X), at(X, 0),
                hasProperty(X, size, large).
46. :- object(X), at(X, 0),
                hasProperty(X, shape, cylinder).
47. :- object(X), at(X, 0),
                hasProperty(X, shape, cone).
48. :- object(X), at(X, 1),
                hasProperty(X, size, small).
49. :- object(X), at(X, 1),
                hasProperty(X, shape, cone).
50. :- object(X), at(X, 1),
                hasProperty(X, material, rubber).
51. :- object(X), at(X, 1),
                hasProperty(X, shape, cube).
52. :- object(X), at(X, 2),
                not hasProperty(X, size, medium).
53. :- object(X), at(X, 2),
                not hasProperty(X, material, metal).
54. :- object(X), at(X, 2),
                hasProperty(X, material, rubber).
55. :- object(X), at(X, 2),
                hasProperty(X, shape, sphere).
56. :- object(X), at(X, 2),
                hasProperty(X, shape, cube).
57. :- object(X), at(X, 3),
                hasProperty(X, size, small).
58  :- object(X), at(X, 3),
                not hasProperty(X, material, metal),
59. not hasProperty(X, color, blue).
60. :- #count{X1, X2: sameProperty(X1, X2, shape),
61.     object(X1), object(X2), at(X1, 3), at(X2, 2),
62.     hasProperty(X1, color, yellow),
63.     hasProperty(X2, color, yellow)} >= 4.
64. :- #count{X1, X2: sameProperty(X1, X2, color),
65.     object(X1), object(X2),
66.     at(X1, 0), at(X2, 3)} >= 2.
```

The following is a natural language interpretation of each line of the preceding rules.

```
44. There are 5 objects in the scene.
45. There are no large size objects in region 0.
46. There are no cylinder shape objects in region 0.
47. There are no cone shape objects in region 0.
48. There are no small size objects in region 1.
49. There are no cone shape objects in region 1.
50. There are no rubber material objects in region 1.
51. There are no cube shape objects in region 1.
52. All objects in region 2 have medium size.
53. All objects in region 2 have metal material.
54. There are no rubber material objects in region 2.
55. There are no sphere shape objects in region 2.
56. There are no cube shape objects in region 2.
57. There are no small size objects in region 3.
58-59. All objects in region 3 have either metal
material or blue color.
60-63. There are at most 1 pairs of color yellow
objects with the same shape in regions 3 and 2
together.
64-66. There are at most 0 pairs of objects with the
same color in regions 0 and 3 together.
```

Question: For each given incomplete scene, we generate one question about (any property) of the missing object. The following is the **question in natural language** that is associated with the incomplete scene in Figure 5:

There is another red rubber object that is the same shape as the big purple object; what size is it?

The following is the same **question represented in ASP**:

```
1.  missing(Q)  :-
2.         hasProperty(X,size,Q),
3.         hasProperty(X,material,rubber),
4.         hasProperty(X,color,red),
5.         hasProperty(Y,color,purple),
6.         hasProperty(Y,size,large),
7.         X!=Y,
8.         same_shape(Y,X).
```

Answer set: The answer set for the above question that satisfies the constraints in the specified environment is:

```
{small, medium}
```

Reasoning Steps: The reasoning involved in deriving the answer set from the question, the incomplete scene, and the constraints in the specified environment is given below.

- Interpreting each line of the question:

```
1. What are the possible values for Q such that:
2. Q is size of the missing object,
3. the missing object's material is rubber,
4. the missing object's color is red,
5. the reference object's color is purple,
6. the reference object's size is large,
7. the missing object is not equal to the
   reference object,
8. the missing object's shape = the
   reference object's shape.
```

- Inferring the missing object's properties: from the scene graph: =>

```
8. the reference object's shape is a sphere.
9. => The missing object's shape is also a sphere.
10. => The missing object is a red rubber sphere.
```

- Inferring the missing object's possible **regions** based on the rules listed as the **Environment's constraints**:

  Among the four regions:

```
A red rubber sphere CAN be located at region 0,
as none of the constraints in lines 45-47 is
violated.

A red rubber sphere CAN'T be located in region
1, as it violates the constraint about the
material at line 50.

A red rubber sphere CAN'T be located in region
2, as it violates the constraints in lines 53,
54, and 55.

A red rubber sphere CAN'T be located in region
3, as it violates the constraints in lines 58-59.

=> The missing red rubber sphere is located at
region 0.
```

- Inferring the possible answer set for the property of interest w.r.t the inferred location of the missing object:

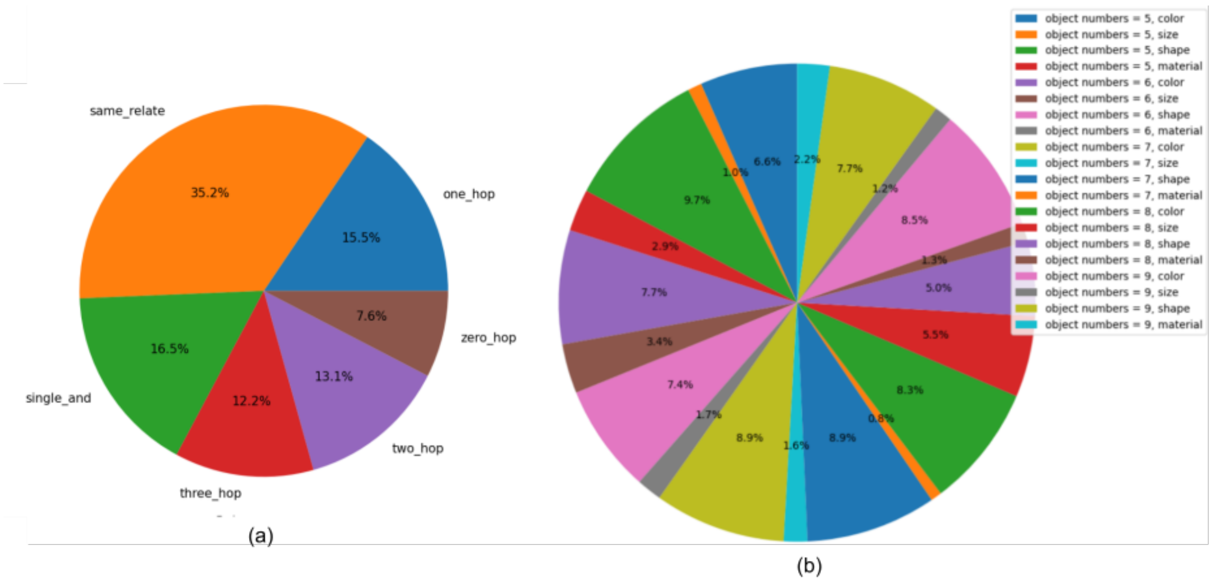


Figure 6: (a) Question templates distribution (b) Distribution of query attributes with object counts between 5 to 9.

```
There are 3 possible values for the size
property:
small, medium, large.

The environment constraint at line 45 discards
the large size for region 0.

=> The possible answer set for Q is:
small, medium.
```

B. Dataset Statistics

Distribution across question templates: Figure 6 (a) shows the distribution of questions across different question templates. Six templates present in the original CLEVR dataset are used in CLEVR-POC.

Distribution of query attributes with number of objects in the scene: Figure 6 (b) shows the distribution of questions of a specific type based on the number of objects in the scene.

Distribution across question types: The type of question asked depends on the attribute of the object that is being inquired about. The generation process enables the user to have control over this distribution. For instance, when generating the specific dataset that was used in the experiments, we established the following criteria: 40% of the questions pertain to the color attribute, another 40% focus on the shape attribute, 10% address the size attribute, and the remaining 10% relate to the material attribute. We made this selection based on the observation that attributes like color and shape encompasses a larger set of values (8 values for color and 4 for shape) in comparison to material (which has just two values). Consequently, the solution space for questions centered around color is more extensive than that for material, resulting in a more diverse solution space for the dataset. Figure 7 (a) displays the question type distribution of the dataset generated based on this setting.

Distribution across solutions: Figure 7 (b), (c), (d), and (e) illustrate the distribution of potential solutions for various question types: size, shape, material, and color, respectively. We aim for a balanced distribution, avoiding a situation where the majority of questions lead to the same set of answers. For instance, when a question pertains to the size of an object, its possible solutions could be one of {large, medium} or {large, small}, or {small, medium} or {large}, or {medium} or {small} as depicted in Figure 7 (b). Since the possible solutions for questions with query attribute *color* are large (as *color* can take 8 values), the entire space is not listed in Figure 7(e). However, it can be seen that the distribution is not favoring any specific solution.

C. Prompts for Language Model

C.1. Stand-alone GPT-4 to solve CLEVR-POC

The format of the prompt provided to GPT-4 when employing it to solve CLEVR-POC is shown below. The prompt contains the task description, the scene description, the constraints or knowledge associated with the scene, the question about the scene, and the answer. The prompt contains two such examples.

Task description: You are a helpful assistant who answers questions about hidden objects based on scene description and the constraints in the scene.

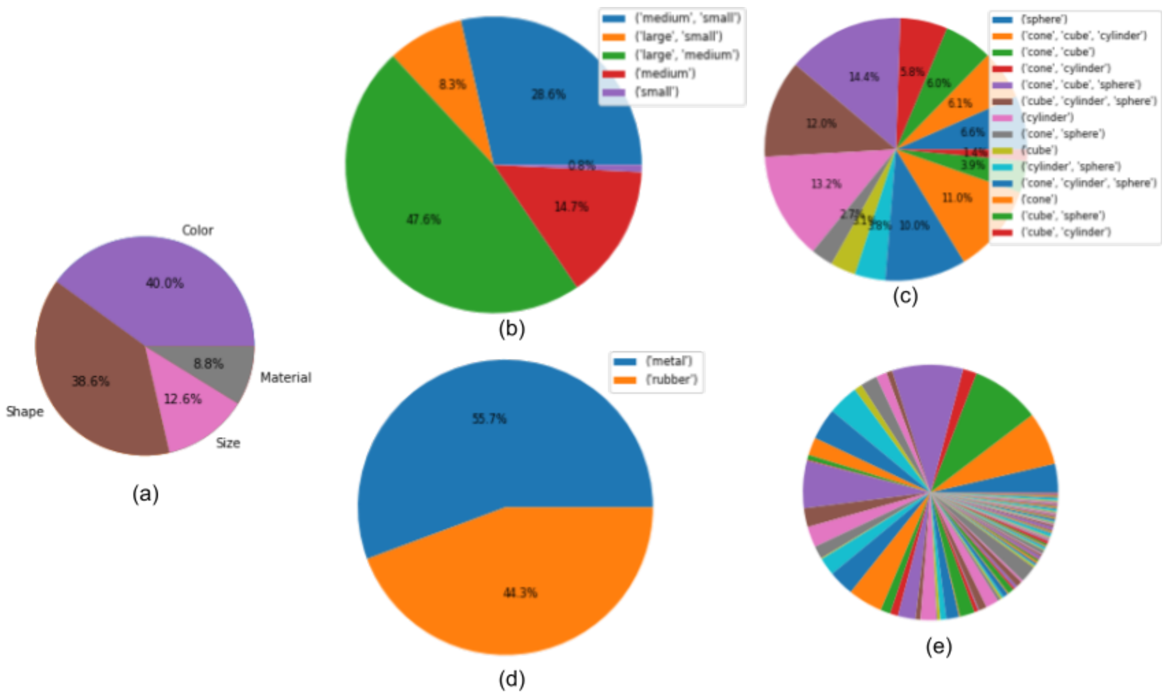


Figure 7: (a) Distribution of question types. (b) Distribution of solutions for questions with query attribute size. (c) Distribution of solutions for questions with query attribute material. (d) Distribution of solutions for questions with query attribute shape. (e) Distribution of solutions for questions with query attribute color. Since the solution space of these questions is larger (> 100), it is not listed here.

The scene graph is in JSON format with the following keys. The key objects contain a list of objects present in the scene. Each object has various attributes like material, color, shape, size, and region. The key relationships hold information about the spatial relationships between objects in the scene. It contains subfields like "front," "right," "left," "behind," etc., each associated with a list of object indices representing objects that have that specific relationship with another object. For example, relationships["front"][0] refers to the objects that are in front of the object at index 0.

Scene Observed: The following is the scene graph:

```
{'objects': [
 {'material': 'metal',  'color': 'red',
  'size': 'medium',  'region': '0',
  'shape': 'cube'
 },
 {'material': 'metal', 'color': 'gray',
  'size': 'medium', 'region': '3',
  'shape': 'sphere'
 },
 {'material': 'metal', 'color': 'brown',
  'size': 'medium', 'region': '1',
  'shape': 'sphere'
 },
 {'material': 'rubber', 'color': 'gray',
  'size': 'medium', 'region': '3',
  'shape': 'sphere'
 },
 {'material': 'metal', 'color': 'red',
  'size': 'medium', 'region': '0',
  'shape': 'sphere'
 },
 {'material': 'rubber', 'color': 'red',
  'size': 'medium', 'region': '2',
  'shape': 'sphere'
 }
],
 'relationships':
  {'left': [[4], [0, 2, 4, 5], [0, 4, 5],  [0,
      1, 2, 4, 5], [], [0, 4]],
   'front': [[1, 3, 4, 5], [5], [0, 1, 3, 4,
       5], [1, 5], [1, 3, 5], []],
   'behind': [[2], [0, 2, 3, 4], [], [0, 2, 4],
        [0, 2], [0, 1, 2, 3, 4]],
   'right': [[1, 2, 3, 5], [3], [1, 3], [], [0,
        1, 2, 3, 5], [1, 2, 3]]
  }
}
```

Constraints: The scene contains several visible objects, and has one additional object that is hidden. Objects must have 4 properties. They are color, shape, size, and material. The scene must conform to the following constraints.

```
Objects can be in one of the 8 colors. It can
     be gray, or red, or blue, or green, or
     brown, or purple, or cyan, or yellow.

Objects can be in one of the 4 shapes. It can
     be a cube, cylinder, sphere, or cone.

Objects can be in one of the 3 sizes. It can be
```

```
      small, medium, or large.

Objects can be in one of the 2 materials. It
     can be rubber or metal.

The scene is divided into 4 regions. They are
     named 0, 1, 2, 3.

If there are two objects and the first object
     is located in region 0 and the second
     object is to the right of the first object
     , then the location of the second object
     is either in region 0, 1, 2, or 3.

If there are two objects and the first object
     is located in region 1 and the second
     object is to the right of the first object
     , then the location of the second object
     is either in region 1, or 3.

If there are two objects and the first object
     is located in region 2 and the second
     object is to the right of the first object
     , then the location of the second object
     is either in region 0, 1, 2, or 3.

If there are two objects, the first object is
     located in region 3 and the second object
     is to the right of the first object, then
     the location of the second object is
     either in region
1, or 3.

If there are two objects, the first object is
     to the right of the second object, then
     the second object is to the left of the
     first object.

If there are two objects, the first object is
     located in region 0 and the second object
     is in front of the first object, then the
     location of the second object is either in
      region 0, 1, 2, or 3.

If there are two objects, the first object is
     located in region 1 and the second object
     is in front of the first object, then the
     location of the second object is either in
      region 0, or 1, or 2, or 3.

If there are two objects, the first object is
     located in region 2 and the second object
     is in front of the first object, then the
     location of the second object is either in
      region 2, or 3.

If there are two objects, the first object is
     located in region 3 and the second object
     is in front of the first object, then the
     location of the second object is either in
      region 2, or 3.

If there are two objects, the first object is
     in front of the second object, then the
     second object is behind the first object.

Every object must be assigned exactly one value
      for color.

Every object must be assigned exactly one value
      for material.

Every object must be assigned exactly one value
      for shape.

Every object must be assigned exactly one value
      for size.

Every object must be assigned exactly one value
      for region.

Two different objects cannot have the same
     values for all the 4 properties.

Every region can have at most 3 objects.

There are 6 objects in the scene.

There are at least 1 pair of color red objects
     with the same size in regions 0 and 2
     together.

There are no small-size objects in region 0.

There are no cone-shaped objects in region 0.

There are no purple color objects in region 0.

There are no blue color objects in region 0.

There are no cylinder shape objects in region
     1.

There are no cyan color objects in region 1.

There are no rubber material objects in region
     1.

There are at least 1 pair of material metal
     objects with the same size in regions 0
     and 3 together.

There are no metal material objects in region
     2.

There are no large-size objects in region 2.

There are at least 1 pair of size medium
     objects with the same shape in regions 1
     and 3 together.

There are no red color objects in region 3.

There are no cube-shaped objects in region 3.

There are at least 1 pair of objects with the
     same material in regions 0 and 1 together.

There are at least 1 pair of color gray objects
      with the same size in regions 3 and 2
     together.
```

Question: Answer the following question about the hidden object. The solution should satisfy the constraints. The other cylinder that is the same material as the medium red thing is what color?

Answer: ###{Gray}###

C.2. GPT-4 as Question Parser

When we use GPT-4 to parse a question to its ASP equivalent, we give as prompt 28 examples of questions in natural language to ASP representation. The prompt with just one Question-ASP pair is shown below.

Task description: You are a helpful assistant that converts questions in English into ASP logic language.

Question: What is the color of the cylinder to the right of the blue sphere?

ASP:

```
###
unknown(Q):-hasProperty(X, color, Q),
             hasProperty(X, shape, cylinder),
             hasProperty(X1, color, blue),
             hasProperty(X1, shape, sphere),
             right(X1, X).
###
```