Relational Programming with Foundation Models

Ziyang Li, Jiani Huang, Jason Liu, Felix Zhu, Eric Zhao, William Dodds, Neelay Velingker, Rajeev Alur, Mayur Naik

University of Pennsylvania

liby99@seas.upenn.edu, jianih@seas.upenn.edu, jasonhl@seas.upenn.edu, zhufelix@seas.upenn.edu, zhaoer@seas.upenn.edu, wdodds@sas.upenn.edu, neelay@seas.upenn.edu, alur@seas.upenn.edu, mhnaik@seas.upenn.edu

Abstract

Foundation models have vast potential to enable diverse AI applications. The powerful yet incomplete nature of these models has spurred a wide range of mechanisms to augment them with capabilities such as in-context learning, information retrieval, and code interpreting. We propose VIEIRA, a declarative framework that unifies these mechanisms in a general solution for programming with foundation models. VIEIRA follows a probabilistic relational paradigm and treats foundation models as stateless functions with relational inputs and outputs. It supports neuro-symbolic applications by enabling the seamless combination of such models with logic programs, as well as complex, multi-modal applications by streamlining the composition of diverse sub-models. We implement VIEIRA by extending the SCALLOP compiler with a foreign interface that supports foundation models as plugins. We implement plugins for 12 foundation models including GPT, CLIP, and SAM. We evaluate VIEIRA on 9 challenging tasks that span language, vision, and structured and vector databases. Our evaluation shows that programs in VIEIRA are concise, can incorporate modern foundation models, and have comparable or better accuracy than competitive baselines.

Introduction

Foundation models are deep neural models that are trained on a very large corpus of data and can be adapted to a wide range of downstream tasks (Bommasani et al. 2021). Exemplars of foundation models include *language models* (LMs) like GPT (Bubeck et al. 2023), *vision models* like Segment Anything (Kirillov et al. 2023), and *multi-modal models* like CLIP (Radford et al. 2021). While foundation models are a fundamental building block, they are inadequate for programming AI applications end-to-end. For example, LMs *hallucinate* and produce nonfactual claims or incorrect reasoning chains (McKenna et al. 2023). Furthermore, they lack the ability to reliably incorporate structured data, which is the dominant form of data in modern databases. Finally, composing different data modalities in custom or complex patterns remains an open problem, despite the advent of multi-modal foundation models such as ViLT (Radford et al. 2021) for visual question answering.


```
@gpt("The height of {{x}} is {{y}} in meters")
type height(bound x: String, y: i32)

// Retrieving height of mountains
rel mount_height(m, h) = mountain(m) and height(m, h)
```

(a) Program **P1**: Extracting knowledge using GPT.

```
@clip(["cat", "dog"])
type classify(bound img: Tensor, label: String)

// Classify each image as cat or dog
rel cat_or_dog(i, l) = image(i, m) and classify(m, l)
```

(b) Program **P2**: Classifying images using CLIP.

mountain

name
Everest
Fuji
K2
Mt.Blanc

P1 ⇨

mount_height

name	height
Everest	8848
Fuji	3776
K2	8611
Mt.Blanc	4808

image

id	img
1	
2	

P2 ⇨

cat_or_dog

prob	id	label
0.02	1	cat
0.98	**1**	**dog**
0.99	**2**	**cat**
0.01	2	dog

(c) Example input-output relations of the programs.

Figure 1: Programs in VIEIRA using foundation models.

Various mechanisms have been proposed to augment foundation models to overcome these limitations. For example, PAL (Gao et al. 2023), WebGPT (Nakano et al. 2021), and Toolformer (Schick et al. 2023) connect LMs with search engines and external tools, expanding their information retrieval and structural reasoning capabilities. LMQL (Beurer-Kellner, Fischer, and Vechev 2022) generalizes pure text prompting in LMs to incorporate scripting. In the domain of computer vision (CV), neuro-symbolic visual reasoning frameworks such as VISPROG (Gupta and Kembhavi 2022) compose diverse vision models with LMs and image processing subroutines. Despite these advances, programmers lack a general solution that systematically incorporates these methods into a single unified framework.

In this paper, we propose VIEIRA, a declarative framework for programming with foundation models. VIEIRA follows a (probabilistic) relational paradigm due to its theoretical and practical versatility. Structured data is commonly stored in relational databases. Relations can also represent structures such as *scene graphs* in vision and *abstract syntax trees* in natural and formal languages. Moreover, extensions for probabilistic and differentiable reasoning enable the integration of relational programming with deep learning in neuro-symbolic frameworks like DeepProbLog (Manhaeve et al. 2018) and SCALLOP (Li, Huang, and Naik 2023).

In VIEIRA, relations form the abstraction layer for interacting with foundation models. Our key insight is that foundation models are ***stateless functions*** *with relational inputs and outputs*. Fig. 1a shows a VIEIRA program which invokes GPT to extract the height of mountains whose names are specified in a structured table. Likewise, the program in Fig. 1b uses the image-text alignment model CLIP to classify images into discrete labels such as `cat` and `dog`. Fig. 1c shows relational input-output examples for the two programs. Notice that the CLIP model also outputs probabilities that allow for probabilistic reasoning.

We implement VIEIRA by extending the SCALLOP compiler with a ***foreign interface*** *that supports foundation models as* ***plugins***. We implement a customizable and extensible plugin library comprising 12 foundation models including GPT, CLIP, and SAM. The resulting unified interface enables a wide spectrum of applications with benefits such as reduced hallucination, retrieval augmentation, and multimodal compositionality. We evaluate VIEIRA on 9 applications that span natural language reasoning, information retrieval, visual question answering, image generation, and image editing. For these applications, we explore diverse methods for programming with foundation models, such as neuro-symbolic reasoning, combining semantic searching with question answering, and modularly composing foundation models. We not only observe on-par or superior performance of our solutions compared to competitive baselines, but also demonstrate their succinctness and ease-of-use.

We summarize our contributions as follows: (1) we introduce a new approach based on relational programming to build applications on top of foundation models; (2) we implement an extensible plugin library of 12 programmable foundation models; and (3) we evaluate VIEIRA on 9 benchmark tasks, and demonstrate comparable or better no-training accuracy than neural-only as well as task-specific baselines. Our framework, plugin library, and evaluations are open-source and available at https://github.com/scallop-lang/scallop.

Related Work

Neuro-symbolic methods. These methods combine the complementary benefits of neural learning and symbolic reasoning. They include domain-specific solutions (Yi et al. 2018; Mao et al. 2019; Li et al. 2020; Wang et al. 2019; Xu et al. 2022; Chen et al. 2020; Minervini et al. 2020) as well as general programming frameworks, such as DeepProbLog (Manhaeve et al. 2018) and SCALLOP (Li, Huang, and Naik 2023). These methods typically concern training or fine-tuning neural models in the presence of logical programs, whereas we target building applications atop foundation models with zero-shot or few-shot examples. Another recent work, the STAR framework (Rajasekharan et al. 2023) also connects a language model (neural) to an answer set programming reasoner (symbolic). It is conceptually similar to VIEIRA but only focuses on natural language understanding and does not support probabilistic reasoning.

Foundation models. These models target different modalities and domains (Touvron et al. 2023; OpenAI 2023; Radford et al. 2021; Kirillov et al. 2023; Radford et al. 2021). Their reasoning capabilities continue to improve with larger context sizes (Ratner et al. 2023), smarter data selection (Adadi 2021), and the discovery of new prompting methods, such as chain-of-thought (Wei et al. 2023; Kojima et al. 2022), self-consistency (Wang et al. 2023), and ReAct (Yao et al. 2023). VIEIRA is orthogonal to these techniques and stands to further enhance the robustness and reliability of foundation models in end-to-end AI applications.

Tools aiding language models. There are many efforts that seek to improve the reasoning abilities of language models (LMs) by incorporating external programs and tools (Gao et al. 2023; Schick et al. 2023; Nakano et al. 2021; Davis and Aaronson 2023). For instance, AutoGPT (Richards 2023) and TaskMatrix.AI (Liang et al. 2023) allows black-box LMs to control symbolic reasoning by invoking commands or calling APIs. On the other hand, many works attempt to extract structured information from LMs for downstream tasks (Gupta and Kembhavi 2022; Beurer-Kellner, Fischer, and Vechev 2022). VIEIRA unifies these two strategies for augmenting model capabilities, and extends them into a glue language for composing multi-modal foundation models.

Language

VIEIRA employs a declarative logic programming language based on Datalog (Abiteboul, Hull, and Vianu 1994). In this section, we present the core language and its *foreign interface* for incorporating diverse foundation models.

Core Language

Relations and data types. The fundamental data type in VIEIRA is set-valued relations comprising tuples of statically-typed primitive values. Besides the standard primitive types such as integers (e.g. `i32`) and string (`String`), VIEIRA introduces two additional types for seamless integration of foundation models: `Tensor` and *Algebraic Data Types* (ADTs). For example, we can declare a relation named `image` to store tuples of image IDs and image `Tensor`s:

```
type image(img_id: i32, img: Tensor)
```

The contents of this relation can be specified via a set of tuples using the built-in *foreign function* `$load_image`:

```
rel image = {(0, $load_image("cat.png")), ...}
```

ADTs in VIEIRA enable the specification of domain specific languages (DSLs) to bridge structured and unstructured data. For example, the following DSL for visual question answering (VQA) describes queries to retrieve scene objects, count objects, and check the existence of objects:

```
type Query = Scene() | Filter(Query, String)
           | Count(Query) | Exists(Query) | ...
// How many balls are there?
const MY_QUERY = Count(Filter(Scene(), "ball"))
```

Logical reasoning. Being based on Datalog, VIEIRA supports defining Horn rules, thereby allowing logical reasoning constructs such as conjunction, disjunction, recursion, stratified negation, and aggregation. Recursion is particularly useful for inductively defining the semantics of a DSL. For example, a (partial) semantics for the above DSL is defined as follows, where `eval_o` and `eval_n` are recursively defined to evaluate objects and numbers, respectively:

```
// Scene returns all objects
rel eval_o(e, o) = case e is Scene() and obj(o)
// Filter applies filter using attributes
rel eval_o(e, o) = case e is Filter(f, a)
    and eval_o(f, o) and attr(o, a)
// Count returns the number of evaluated objects
rel eval_n(e, n) = n := count(o: eval_o(e1, o)
    where e1: case e is Count(e1))
... // other cases of ‘e’
```

Note that the `case-is` operator matches patterns of the ADT and the `count` aggregator counts the number of entities. When combined with foundation models, principled reasoning semantics in this style can compensate for individual foundation models' lack of reasoning capability.

Probabilistic soft logic. Tuples can be tagged with probabilities. The example below shows hard-coded probabilities, suggesting that the entity is more likely a dog than a cat:

```
rel animal = {0.1::(1,"cat"), 0.9::(1,"dog")}
```

Soft-logic operations produce probabilities as well. For instance, the *soft-eq* operator ($\simeq$) on `Tensor`s derives cosine-similarity between tensors, enabling features like *soft-join* and applications like *semantic search*. In the following example, we compute similarity scores between distinct documents by performing *soft-join* on their embeddings:

```
type doc(id: i32, embed: Tensor) // embed docs
rel sim(i, j) = doc(i, v) and doc(j, v) and i!=j
// equiv: sim(i, j) = doc(i, v1) and doc(j, v2)
     and i!=j and v1~=v2
```

Notice that in the above rule, a join on a tensor value `v` is desugared into a soft-eq on two individual variables (denoted `v1` and `v2`). Internally, with the provenance framework provided by SCALLOP (Li, Huang, and Naik 2023), we use the top-k-proofs semiring (Huang et al. 2021) for scalable probabilistic reasoning, thus enabling features such as ranking and uncertainty estimation.

Foreign Interface

In order to incorporate foundation models, we design a foreign interface with two main programming constructs, called *foreign predicate* and *foreign attribute*. They can be defined externally in languages like Python and imported into VIEIRA for application.

Foreign Predicate (FP). Foreign predicates can be used in rules just like other relations. However, instead of grounding relational facts from a table, FPs ground facts by invoking external functions. The syntax for defining FPs is as follows:

```
extern type PRED([bound|free]? ARG: TYPE, ...)
```

In addition to the type, each argument is specified either as a *bounded* argument (using the keyword `bound`) or a *free* argument (using `free` or omitted for brevity). Semantically, FPs are functions that take in a tuple of bounded arguments and return a list of tuples of free arguments. The runtime of VIEIRA performs memoization on FP results to avoid redundant computation. Optionally, FPs can tag a probability to each returned tuple for further probabilistic reasoning.

```python
@foreign_attribute
def clip(pred: Predicate, labels: List[str]):
  # Sanity checks for predicate and labels...
  assert pred.args[0].ty == Tensor and ...

  @foreign_predicate(name=pred.name)
  def run_clip(img: Tensor) -> Facts[str]:
    # Invoke CLIP to classify image into labels
    probs = clip_model(img, labels)
    # Each result is tagged by a probability
    for (prob, label) in zip(probs, labels):
      yield (prob, (label,)) # prob::(label,)

  return run_clip
```

Figure 2: Snippet of Python implementation of the foreign attribute `clip` which uses the CLIP model for image classification. Notice that the FA `clip` returns the FP `run_clip`.

Foreign Attribute (FA). In VIEIRA, attributes can be used to *decorate* declarations of predicates. They are higher-order functions that take in the provided arguments and the decorated predicate to return a new predicate. The syntax for using an attribute to decorate a predicate is:

```
@ATTR(POS_ARG, ..., KEY=KW_ARG, ...)
type PRED([bound|free]? ARG: TYPE, ...)
```

The attribute is applied prior to the compilation of VIEIRA programs. For interfacing with foundation models, the positional and keyword arguments are particularly helpful in configuring the underlying model, hiding low-level details. Fig. 2 illustrates one succinct implementation of the FA that enables the use of the CLIP model shown in Fig. 1b.

Foundation Models

VIEIRA provides an extensible *plugin framework* that adapts to the evolving landscape of foundation models. In this work, we have implemented 7 plugins, covering 12 foundation models, all through the foreign interface. Our design principle for the interface is three-fold: simplicity, configurability, and compositionality. In this section, we present several representative predicates and attributes which substantially support the applicability of VIEIRA to diverse machine learning tasks.

Text completion. In VIEIRA, language models like GPT (OpenAI 2023) and LLaMA (Touvron et al. 2023) can be used as basic foreign predicates for text completion:

```
extern type gpt(bound p: String, a: String)
rel ans(a) = gpt("population of NY is", a)
```

In this case, `gpt` is an arity-2 FP that takes in a `String` as the prompt and produces a `String` as the response. It uses the model `gpt-3.5-turbo` by default. To make the interface more relational and structural, we provide an FA:

```
@gpt("the population of {{loc}} is {{num}}",
  examples=[("NY", 8468000), ...])
type population(bound loc: String, num: u32)
```

Here, we declare a relation named `population` which produces a population number (`num`) given a location (`loc`) as input. Notice that structured few-shot examples are provided through the argument `examples`.

Semantic parsing. One can directly configure language models to perform *semantic parsing*. For instance, the semantic parser for the simple `Query` DSL (partially defined in the Language section) can be declared as follows:

```
@gpt_semantic_parse(
  "Please semantically parse questions...",
  examples=[("How many red things are there?",
    "Count(Filter(Scene(), 'red'))"), ...])
type parse_query(bound x: String, y: Query)
```

Internally, the language model is expected to generate a fully structured `Query` in its string form. Then, VIEIRA attempts to parse the string to construct actual ADT values. In practice, the success of semantic parsing depends heavily on the design of the DSL, involving factors like intuitiveness (e.g., names and arguments of ADT variants) and complexity (e.g., number of possible ADT variants).

Relational data extraction. Structural relational knowledge available in free-form textual data can be extracted by language models. We introduce a foreign attribute `@gpt_extract_relation` for this purpose. For instance, the following declared predicate takes in a context and produces (subject, object, relation) triplets:

```
@gpt_extract_relation(
  "Extract the implied kinship relations",
  examples=[("Alice and her son Bob went to...",
    [("alice", "bob", "son"), ...])])
type extract_kinship(bound ctx: String,
  sub: String, obj: String, rela: String)
```

This attribute differs from the text completion attribute in that it can extract an arbitrary number of facts. The underlying implementation prompts LMs to respond with JSON-formatted strings, allowing structured facts to be parsed.

Language models for textual embedding. Textual embeddings are useful in performing tasks such as information retrieval. The following example declares an FP encapsulating a cross-encoder (Nogueira and Cho 2019):

```
@cross_encoder("nli-deberta-v3-xsmall")
type enc(bound input: String, embed: Tensor)
rel sim() = enc("cat", e) and enc("neko", e)
```

In the last line, we compute the cosine-similarity of the encoded embeddings using a soft-join on the variable `e`. As a result, we obtain a probabilistic fact like `0.9::sim()` whose probability encodes the cosine-similarity between the textual embeddings of `"cat"` and `"neko"`.

Image classification models. Image-text alignment models, such as CLIP (Radford et al. 2021), can naturally be used as zero-shot image classification models. Fig. 1b shows an example usage of the `@clip` attribute. We also note that dynamically-generated classification labels can be provided to CLIP via a bounded argument in the predicate.

Image segmentation models. OWL-ViT (Minderer et al. 2022), Segment Anything Model (SAM) (Kirillov et al. 2023), and DSFD (Li et al. 2018) are included in VIEIRA as image segmentation (IS) and object localization (LOC) models. IS and LOC models can provide many outputs, such as bounding boxes, classified labels, masks, and cropped images. For instance, the OWL-ViT model can be used and configured as follows:

```
@owl_vit(["human face", "rocket"])
type find_obj(bound img: Tensor,
  id: u32, label: String, cropped_image: Tensor)
```

Here, the `find_obj` predicate takes in an image, and finds image segments containing "human face" or "rocket". According to the names of the arguments, the model extracts 3 values per segment: ID, label, and cropped image. Note that each produced fact will be associated with a probability, representing the confidence from the model.

Image generation models. Visual generative models such as Stable Diffusion (Rombach et al. 2022) and DALL-E (Ramesh et al. 2021) can be regarded as relations as well. The following example shows the declaration of the `gen_image` predicate, which encapsulates a diffusion model:

```
@stable_diffusion("stable-diffusion-v1-4")
type gen_image(bound txt: String, img: Tensor)
```

As can be seen from the signature, it takes in a `String` text as input and produces a `Tensor` image as output. Optional arguments such as the desired image resolution and the number of inference steps can be supplied to dictate the granularity of the generated image.

Tasks and Solutions

We apply VIEIRA to solve 9 benchmark tasks depicted in Fig. 3. Table 1 summarizes the datasets, evaluation metrics, and the foundation models used in our solutions. We elaborate upon the evaluation settings and our solutions below.

Date reasoning (DR). In this task adapted from BIG-bench (Srivastava et al. 2023), the model is given a context and asked to compute a date. The questions test the model's temporal and numerical reasoning skills, as well as its grasp of common knowledge. Unlike BIG-bench where multiple-choice answers are given, we require the model to directly produce its answer in MM/DD/YYYY form.

Our solution leverages GPT-4 (5-shot[1]) for extracting 3 relations: mentioned dates, duration between date labels, and the target date label. From here, our relational program iterates through durations to compute dates for all date labels. Lastly, the date of the target label is returned as the output.

Tracking shuffled objects (TSO). In this task from BIG-bench, a textual description of pairwise object swaps among people is given, and the model needs to track and derive which object is in a specified person's possession at the end.

[1]In this work, k in "k-shot" means the number of examples provided to the LM component within the full solution. Each example is a ground-truth input-output pair for the LM.

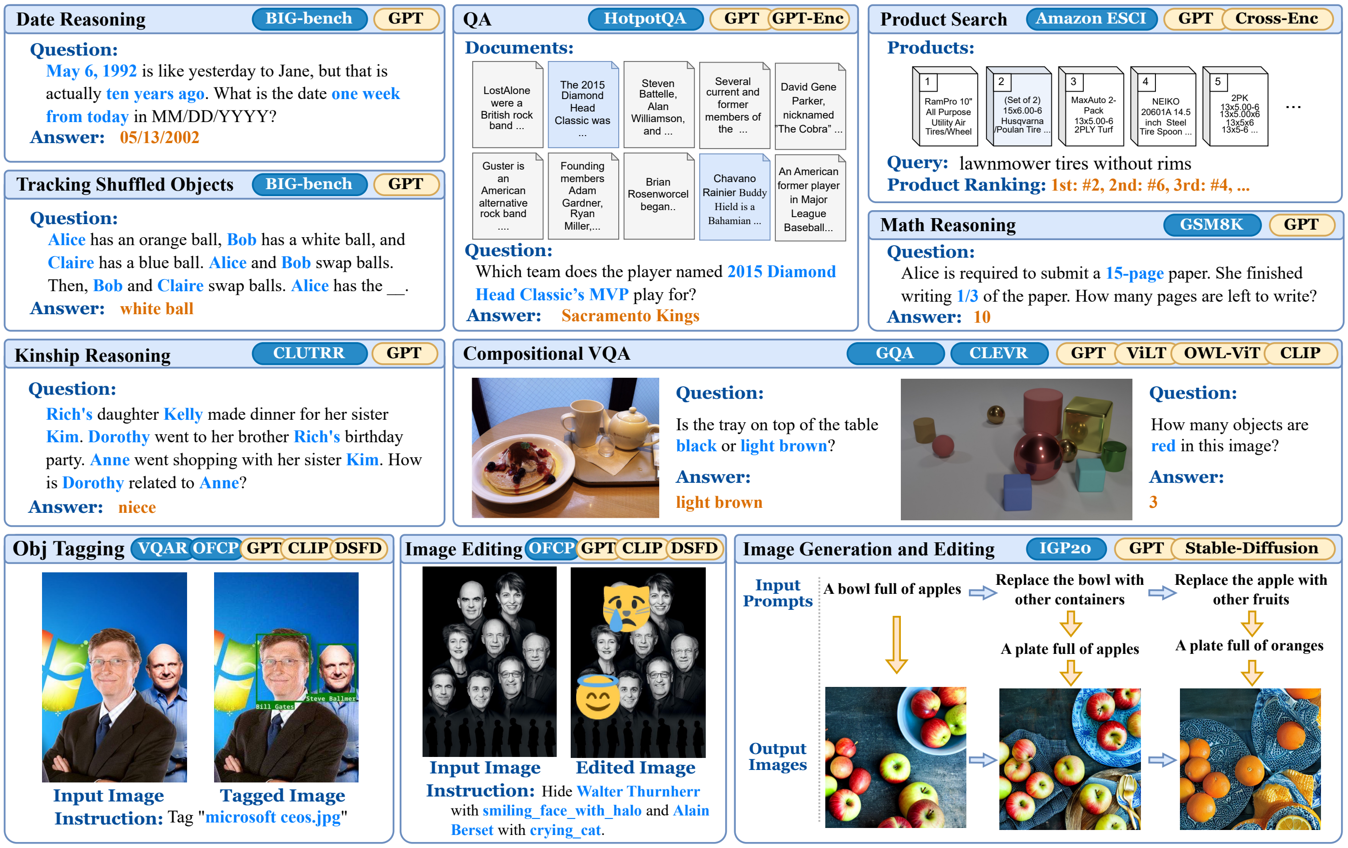


Figure 3: Benchmark tasks. The top of each box lists the dataset(s) and the foundation models used in our solutions.

There are three difficulty levels depending on the number of objects to track, denoted by $n \in \{3, 5, 7\}$.

Our solution for tracking shuffled objects relies on GPT-4 (1-shot) to extract 3 relations: initial possessions, swaps, and the target person whose final possessed object is expected as the answer. Our reasoning program iterates through all the swaps starting from the initial state and retrieves the last possessed object associated with the target.

Kinship reasoning (KR). CLUTRR (Sinha et al. 2019) is a kinship reasoning dataset of stories which indicate the kinship between characters, and requires the model to infer the relationship between two specified characters. The questions have different difficulty levels based on the length of the reasoning chain, denoted by $k \in \{2 \ldots 10\}$.

Our solution for kinship reasoning invokes GPT-4 (2-shot) to extract the kinship graph from the context. We also provide an external common-sense knowledge base for rules like "mother's mother is grandmother". Our program then uses the rules to derive other kinship relations. Lastly, we retrieve the kinship between the specified pair of people.

Math reasoning (MR). This task is drawn from the GSM8K dataset of arithmetic word problems (Cobbe et al. 2021). The questions involve grade school math word problems created by human problem writers, and the model is asked to produce a number as the result. Since the output can be fractional, we allow a small delta when comparing the derived result with the ground truth.

Our solution to this task prompts GPT-4 (2-shot) to produce step-by-step expressions, which can contain constants, variables, and simple arithmetic operations. We evaluate all the expressions through a DSL, and the result associated with the goal variable is returned. By focusing the LM's responsibility solely on semantic parsing, our relational program can then achieve faithful numerical computation via DSL evaluation.

Question answering with information retrieval (QA). We choose HotpotQA (Yang et al. 2018), a Wikipedia-based question answering (QA) dataset under the "distractor" setting. Here, the model takes in 2 parts of inputs: 1) a question, and 2) 10 Wikipedia paragraphs as the context for answering the question. Among the 10 Wikipedia pages, at most 2 are relevant to the answer, while the others are distractors.

Our solution is an adaptation of FE2H (Li, Lei, and Yang 2022), which is a 2-stage procedure. First, we turn the 10 documents into a vector database by embedding each document. We then use the embedding of the question to retrieve the 2 most related documents, which are then fed to a language model to do QA. In this case, the QA model does not have to process all 10 documents, leading to less distraction.

Product search (PS). We use Amazon's ESCI Product Search dataset (Reddy et al. 2022). The model is provided with a natural language (NL) query and a list of products (23

Task	Dataset	#Test Samples	Metric	Foundation Models Used
DR	DR	369	EM	GPT-4
TSO	TSO	150	EM	GPT-4
KR	CLUTRR	1146	EM	GPT-4
MR	GSM8K	1319	EM	GPT-4
QA	Hotpot QA	1000	EM	GPT-4
				ada-002
PS	Amazon ESCI	1000	nDCG	GPT-4
				ada-002
VQA	CLEVR	480	Recall@1 Recall@3	GPT-4
				OWL-ViT
	GQA	500		VilT
				CLIP
VOT	VQAR	100	MI	OWL-ViT
				VilT
				GPT-4
	OFCP	50		DSFD
				CLIP
IGE	OFCP	50	MI	DFSD
				CLIP
	IGP20	20		GPT-4
				Diffusion

Table 1: Characteristics of benchmark tasks including the dataset used, its size, and evaluation metrics. Metrics include exact match (EM), normalized discounted cumulative gain (nDCG), and manual inspection (MI). We also denote the foundation models used in our solution for each task.

products on average). The goal is to rank the products that best match the query. In the dataset, for each pair of query and product, a label among E (exact match), S (substitute), C (complementary), and I (irrelevant) is provided. The metric we use to evaluate the performance is nDCG. The gains are set to be 1.0 for E, 0.1 for S, 0.01 for C, and 0.0 for I.

One challenge of this dataset is that many queries contain negative statements. For example, in the query "#1 treadmill without remote", the "remote" is undesirable. Therefore, instead of computing the embedding of the full query, we decompose the query into positive and negative parts. We then perform semantic search by maximizing the similarity of the positive part while minimizing that of the negative part.

Compositional visual question answering (VQA). We choose two compositional VQA datasets, GQA (Hudson and Manning 2019) and CLEVR (Johnson et al. 2016). In this task, the model is given an image and a question, and needs to answer the question. For GQA, the majority of questions expect yes/no answers, while CLEVR's questions demand features like counting and spatial reasoning. We uniformly sample 500 and 480 examples from GQA and CLEVR datasets respectively. Following VQA conventions (Kim, Son, and Kim 2021), we use Recall@k where $k \in \{1, 3\}$ as the evaluation metrics.

Our solution for GQA is an adaptation of VISPROG (Gupta and Kembhavi 2022). We create a DSL for invoking vision modules such as ViLT and OWL-ViT, and use GPT-4 for converting questions into programs in this DSL. Our solution for CLEVR is similar, directly replicating the DSL provided by the original work. OWL-ViT and CLIP are used to detect objects and infer attributes, while the spatial relations are directly computed using the bounding box data.

Visual object tagging (VOT). We evaluate on two datasets, VQAR (Huang et al. 2021) and OFCP. For VQAR, the model is given an image and a programmatic query, and is asked to produce bounding boxes of the queried objects in the image. Our solution composes a relational knowledge base, defining entity names and relationships, with object retrieval (OWL-ViT) and visual QA (ViLT) models.

Online Faces of Celebrities and Politicians (OFCP) is a self-curated dataset of images from Wikimedia Commons among other sources. For this dataset, the model is given an image with a descriptive NL filename, and needs to detect faces relevant to the description and tag them with their names. Our solution obtains a set of possible names from GPT-4 and candidate faces from DSFD. These are provided to CLIP for object classification, after which probabilistic reasoning filters the most relevant face-name pairs.

Language-guided image generation and editing (IGE). We adopt the task of image editing from (Gupta and Kembhavi 2022). In this task, the instruction for image editing is provided through NL, and can invoke operations such as blurring background, popping color, and overlaying emojis. Due to the absence of an existing dataset, we repurpose the OFCP dataset by introducing 50 NL image editing prompts. Our solution for this task is centered around a DSL for image editing. We incorporate GPT-4 for semantic parsing, DSFD for face detection, and CLIP for entity classification. Modules for image editing operations are implemented as individual foreign functions.

For free-form generation and editing of images, we curate IGP20, a set of 20 prompts for image generation and editing. Instead of using the full prompt, we employ an LM to decompose complex NL instructions into simpler steps. We define a DSL with high-level operators such as generate, reweight, refine, replace, and negate. We use a combination of GPT-4, Prompt-to-Prompt (Hertz et al. 2022), and diffusion model (Rombach et al. 2022) to implement the semantics of our DSL. We highlight our capability of grounding positive terms from negative phrases, which enables handling prompts like "replace apple with other fruits" (Fig. 3).

Experiments and Analysis

We aim to answer the following research questions:

RQ1. Is VIEIRA programmable enough to be applicable to a diverse range of applications with minimal effort?

RQ2. How do solutions using VIEIRA compare to other baseline methods in the no-training setting?

RQ1: Programmability

While a user study for VIEIRA's programmability is out of scope in this paper, we qualitatively evaluate its programmability on three aspects. First, we summarize the lines-of-code (LoC) for each of our solutions in Table 2. The programs

Dataset	LoC	Prompt LoC	Dataset	LoC	Prompt LoC
DR	69	48	CLEVR	178	45
TSO	34	16	GQA	82	36
CLUTRR	61	45	VQAR	53	11
GSM8K	47	28	OFCP (VOT)	33	2
HotpotQA	47	24	OFCP (IGE)	117	44
ESCI	32	7	IGP20	50	12

Table 2: The lines-of-code (LoC) numbers of our solutions for each dataset. The LoC includes empty lines, comments, natural language prompts, and DSL definitions. We note specifically the LoC of prompts in the table.

Method	DR	TSO	CLUTRR	GSM8K
GPT-4	71.00 (0-shot)	30.00 (0-shot)	43.10 (3-shot)	87.10 (0-shot)
GPT-4 (CoT)	87.26 (0-shot)	84.00 (0-shot)	24.17 (3-shot)	**92.00** (5-shot)
Ours	**92.41**	**100.00**	**72.50**	90.60

Table 3: The performance on the natural language reasoning datasets. Numbers are in percentage (%).

HotpotQA			Amazon ESCI		
Method	Fine-tuned	EM	Method	Fine-tuned	nDCG
C2FM	✓	72.07%	BERT	✓	0.830
FE2H	✓	71.89%	CE-MPNet	✓	0.857
—	—	—	MIPS	✗	0.797
Ours	✗	67.3%	Ours	✗	0.798

Table 4: The performance on the HotpotQA and Amazon ESCI. We also include performance numbers from methods which are fine-tuned on the corresponding dataset.

are concise, as most are under 100 lines. Notably, natural language prompts (including few-shot examples) take up a significant portion of each solution. Secondly, 8 out of 10 solutions are coded by undergraduate students with no background in logic and relational programming, providing further evidence of VIEIRA's user-friendliness. Last but not least, our solutions are interpretable and thus offer debuggability. Specifically, all the intermediate relations are available for inspection, allowing systematic error analysis.

RQ2: Baselines and Comparisons

We compare the performance of our solutions to existing baselines under the no-training setting. In particular, our solutions achieve better performance than comparable baselines on 6 out of 8 studied datasets with baselines. Below, we classify the tasks into 4 categories and discuss the respective performance and comparisons.

Natural language reasoning. For the tasks of DR, TSO, CLUTRR, and GSM8K, we pick a generic baseline of GPT-

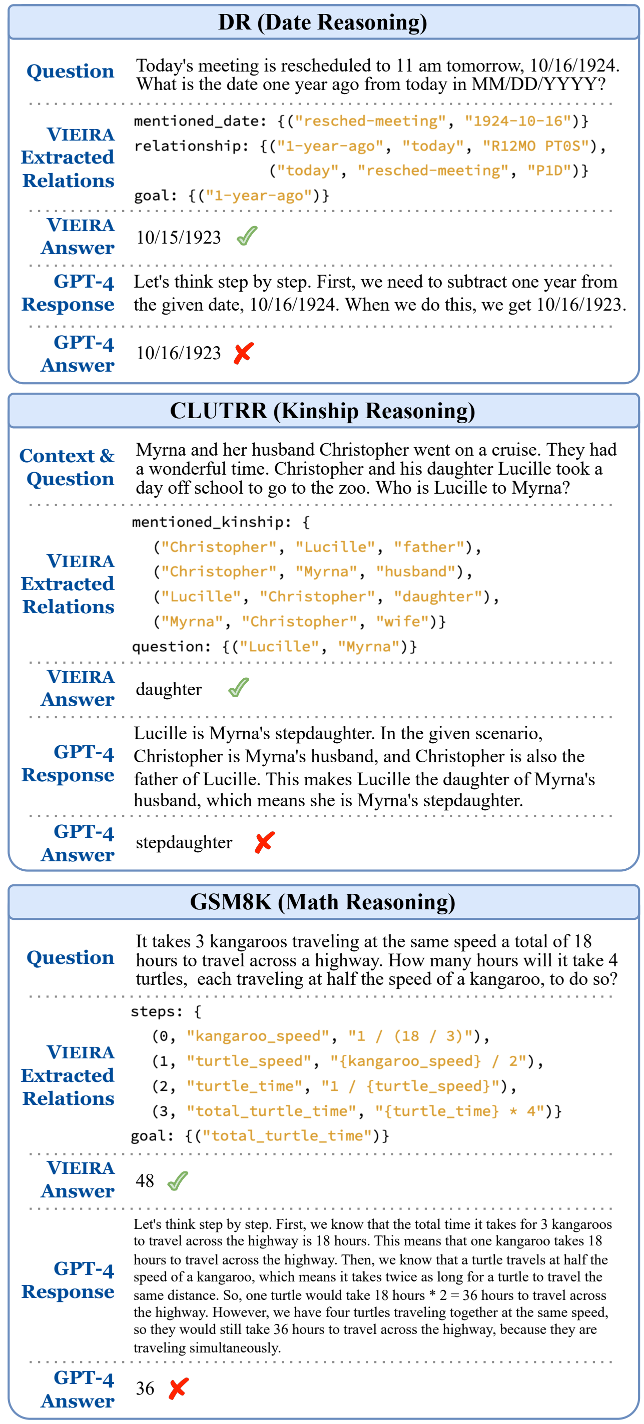
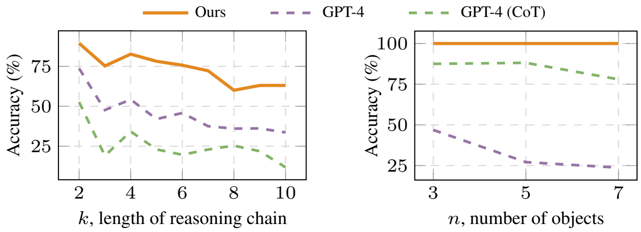


Figure 4: Illustrative comparisons between our solution and GPT-4 (zero-shot CoT) on selected questions from DR, CLUTRR, and GSM8K datasets. We also include the extracted relations used for subsequent reasoning.

4 under zero-shot, few-shot, and chain-of-thought (CoT) settings. All our solutions also rely on GPT-4 (few-shot), but we note that our shots only include extracted facts, and not the final answer or any reasoning chains. The data in Ta-



(a) CLUTRR (b) TSO

Figure 5: Systematic generalizability comparisons on the CLUTRR and TSO datasets.

Method	GQA		CLEVR	
	Recall@1	Recall@3	Recall@1	Recall@3
ViLT-VQA	0.049	0.462	0.241	0.523
PNP-VQA	0.419	—	—	—
Ours	**0.579**	**0.665**	**0.463**	**0.638**

Table 5: Quantitative results on the VQA datasets.

ble 3 indicates that our method can significantly enhance reasoning performance and reduce hallucination, exemplified by achieving a flawless 100% accuracy on the TSO dataset. Note that on GSM8K, our method scores slightly lower than the baseline; we conjecture that our solution demands more from GPT-4 itself to extract structured computation steps. On CLUTRR, our solution even outperforms fCoT (Lyu et al. 2023), a special prompting technique with external tool use, by 0.6%. In Fig. 5 we illustrate the systematic generalizability of our methods. The performance of our solutions remains relatively consistent even when the problems become harder. We provide illustrative examples in Fig. 4 showing comparisons between our method and GPT-4 (zero-shot CoT).

Retrieval augmentation and semantic search. For the HotpotQA dataset, our solution is an adaptation of FE2H (Li, Lei, and Yang 2022), a retrieval-augmented question answering approach. As seen in Table 4, with no fine-tuning, our method scores only a few percentages lower than fine-tuned methods C2FM (Yin et al. 2022) and FE2H. For the Amazon ESCI dataset, our solution performs semantic search for product ranking. While performing slightly lower than the fine-tuned methods (Reddy et al. 2022; Song et al. 2020), our solution outperforms maximum inner product search (MIPS) based on GPT text encoder (`text-embedding-ada-002`).

Compositional multi-modal reasoning. For VQA, we pick ViLT-VQA (Kim, Son, and Kim 2021) (a pre-trained foundation model) and PNP-VQA (Tiong et al. 2022) (a zero-shot VQA method) as baselines. As shown in Table 5, our method significantly outperforms the baseline model on both datasets. Compared to the neural-only baseline, our approach that combines DSL and logical reasoning more effectively handles intricate logical operations such as counting and numerical comparisons. On GQA, out method outperforms previous zero-shot state-of-the-art, PNP-VQA, by 0.16 (0.42 to 0.58). For object and face tagging, without training or fine-tuning, our method achieves 67.61% and 60.82% semantic correctness rates (Table 6).


```
Instruction: Replace the bowl with something
else, and change the apples to other fruits.
```

Figure 6: Qualitative comparison of image editing. Compared to InstructPix2Pix, our image editing method follows the instructed edits better, as it successfully changed the bowl into plate and apples to oranges.

Method	Visual Object Tagging		Image Editing
	VQAR	OFCP	OFCP
Ours	67.61%	60.82%	74.00%

Table 6: Quantitative results on object tagging and image editing tasks. We manually evaluate the tagged entities and the edited images for semantic correctness rates.

Image generation and editing. For image generation and editing, we apply our technique to the OFCP and IGP20 datasets. We rely on manual inspection for evaluating our performance on the OFCP dataset, and we observe 37 correctly edited images out of the 50 evaluated ones, resulting in a 74% semantic correctness rate (Table 6). For IGP20, we choose as the baseline a diffusion model, InstructPix2Pix (Brooks, Holynski, and Efros 2023), which also combines GPT-3 with image editing. We show one example baseline comparison illustrated in Figure 6.

Conclusion

We introduced VIEIRA, a declarative framework designed for relational programming with foundation models. VIEIRA brings together foundation models from diverse domains, providing a unified interface for composition and the ability to perform probabilistic logical reasoning. This results in solutions with comparable and often superior performance than neural-based baselines. In the future, we aim to extend the capabilities of VIEIRA beyond the current in-context learning settings to weakly-supervised training and fine-tuning of foundation models in an end-to-end manner.

Acknowledgements

We thank the anonymous reviewers for useful feedback. This research was supported by NSF grant #2313010 and DARPA grant #FA8750-23-C-0080. Ziyang Li was supported by an Amazon Fellowship.

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1994. *Foundations of Databases: The Logical Level*. Pearson, 1st edition.

Adadi, A. 2021. A survey on data-efficient algorithms in big data era. *Journal of Big Data*, 8(1): 24.

Beurer-Kellner, L.; Fischer, M.; and Vechev, M. 2022. Prompting Is Programming: A Query Language For Large Language Models. In *PLDI*.

Bommasani, R.; Hudson, D. A.; Adeli, E.; Altman, R. B.; Arora, S.; von Arx, S.; Bernstein, M. S.; Bohg, J.; Bosselut, A.; Brunskill, E.; and et al. 2021. On the Opportunities and Risks of Foundation Models. arXiv:2108.07258.

Brooks, T.; Holynski, A.; and Efros, A. A. 2023. InstructPix2Pix: Learning to Follow Image Editing Instructions. arXiv:2211.09800.

Bubeck, S.; Chandrasekaran, V.; Eldan, R.; Gehrke, J.; Horvitz, E.; Kamar, E.; Lee, P.; Lee, Y. T.; Li, Y.; Lundberg, S.; et al. 2023. Sparks of artificial general intelligence: Early experiments with GPT-4. arXiv:2303.12712.

Chen, X.; Liang, C.; Yu, A. W.; Zhou, D.; Song, D.; and Le, Q. V. 2020. Neural Symbolic Reader: Scalable Integration of Distributed and Symbolic Representations for Reading Comprehension. In *ICLR*.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training Verifiers to Solve Math Word Problems. arXiv:2110.14168.

Davis, E.; and Aaronson, S. 2023. Testing GPT-4 with Wolfram Alpha and Code Interpreter plug-ins on math and science problems. arXiv:2308.05713.

Gao, L.; Madaan, A.; Zhou, S.; Alon, U.; Liu, P.; Yang, Y.; Callan, J.; and Neubig, G. 2023. PAL: Program-aided Language Models. arXiv:2211.10435.

Gupta, T.; and Kembhavi, A. 2022. Visual Programming: Compositional visual reasoning without training. arXiv:2211.11559.

Hertz, A.; Mokady, R.; Tenenbaum, J.; Aberman, K.; Pritch, Y.; and Cohen-Or, D. 2022. Prompt-to-Prompt Image Editing with Cross Attention Control. arXiv:2208.01626.

Huang, J.; Li, Z.; Chen, B.; Samel, K.; Naik, M.; Song, L.; and Si, X. 2021. Scallop: From Probabilistic Deductive Databases to Scalable Differentiable Reasoning. In *NeurIPS*.

Hudson, D. A.; and Manning, C. D. 2019. GQA: a new dataset for compositional question answering over real-world images. arXiv:1902.09506.

Johnson, J.; Hariharan, B.; van der Maaten, L.; Fei-Fei, L.; Zitnick, C. L.; and Girshick, R. B. 2016. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. arXiv:1612.06890.

Kim, W.; Son, B.; and Kim, I. 2021. ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision. arXiv:2102.03334.

Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A. C.; Lo, W.-Y.; et al. 2023. Segment Anything. arXiv:2304.02643.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022. Large language models are zero-shot reasoners. In *NeurIPS*.

Li, J.; Wang, Y.; Wang, C.; Tai, Y.; Qian, J.; Yang, J.; Wang, C.; Li, J.; and Huang, F. 2018. DSFD: Dual Shot Face Detector. arXiv:1810.10220.

Li, Q.; Huang, S.; Hong, Y.; Chen, Y.; Wu, Y. N.; and Zhu, S.-C. 2020. Closed Loop Neural-Symbolic Learning via Integrating Neural Perception, Grammar Parsing, and Symbolic Reasoning. In *ICML*.

Li, X.-Y.; Lei, W.-J.; and Yang, Y.-B. 2022. From Easy to Hard: Two-stage Selector and Reader for Multi-hop Question Answering. arXiv:2205.11729.

Li, Z.; Huang, J.; and Naik, M. 2023. Scallop: A Language for Neurosymbolic Programming. In *PLDI*.

Liang, Y.; Wu, C.; Song, T.; Wu, W.; Xia, Y.; Liu, Y.; Ou, Y.; Lu, S.; Ji, L.; Mao, S.; Wang, Y.; Shou, L.; Gong, M.; and Duan, N. 2023. TaskMatrix.AI: Completing Tasks by Connecting Foundation Models with Millions of APIs. arXiv:2303.16434.

Lyu, Q.; Havaldar, S.; Stein, A.; Zhang, L.; Rao, D.; Wong, E.; Apidianaki, M.; and Callison-Burch, C. 2023. Faithful Chain-of-Thought Reasoning. arXiv:2301.13379.

Manhaeve, R.; Dumancic, S.; Kimmig, A.; Demeester, T.; and Raedt, L. D. 2018. DeepProbLog: Neural Probabilistic Logic Programming. arXiv:1805.10872.

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. arXiv:1904.12584.

McKenna, N.; Li, T.; Cheng, L.; Hosseini, M. J.; Johnson, M.; and Steedman, M. 2023. Sources of Hallucination by Large Language Models on Inference Tasks. arXiv:2305.14552.

Minderer, M.; Gritsenko, 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 with Vision Transformers. arXiv:2205.06230.

Minervini, P.; Riedel, S.; Stenetorp, P.; Grefenstette, E.; and Rocktäschel, T. 2020. Learning Reasoning Strategies in End-to-End Differentiable Proving. In *ICML*.

Nakano, R.; Hilton, J.; Balaji, S.; Wu, J.; Ouyang, L.; Kim, C.; Hesse, C.; Jain, S.; Kosaraju, V.; Saunders, W.; Jiang, X.; Cobbe, K.; Eloundou, T.; Krueger, G.; Button, K.; Knight, M.; Chess, B.; and Schulman, J. 2021. WebGPT: Browser-assisted question-answering with human feedback. arXiv:2112.09332.

Nogueira, R.; and Cho, K. 2019. Passage Re-ranking with BERT. arXiv:1901.04085.

OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774.

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. arXiv:2103.00020.

Rajasekharan, A.; Zeng, Y.; Padalkar, P.; and Gupta, G. 2023. Reliable Natural Language Understanding with Large Language Models and Answer Set Programming. In *International Conference on Logic Programming*.

Ramesh, A.; Pavlov, M.; Goh, G.; Gray, S.; Voss, C.; Radford, A.; Chen, M.; and Sutskever, I. 2021. Zero-shot text-to-image generation. In *ICML*.

Ratner, N.; Levine, Y.; Belinkov, Y.; Ram, O.; Magar, I.; Abend, O.; Karpas, E.; Shashua, A.; Leyton-Brown, K.; and Shoham, Y. 2023. Parallel Context Windows for Large Language Models. In *Proceedings of the ACL*.

Reddy, C. K.; Màrquez, L.; Valero, F.; Rao, N.; Zaragoza, H.; Bandyopadhyay, S.; Biswas, A.; Xing, A.; and Subbian, K. 2022. Shopping Queries Dataset: A Large-Scale ESCI Benchmark for Improving Product Search. arXiv:2206.06588.

Richards, T. B. 2023. AutoGPT. https://github.com/Significant-Gravitas/AutoGPT. Accessed: 2024-02-12.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-Resolution Image Synthesis With Latent Diffusion Models. In *CVPR*.

Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Zettlemoyer, L.; Cancedda, N.; and Scialom, T. 2023. Toolformer: Language Models Can Teach Themselves to Use Tools. arXiv:2302.04761.

Sinha, K.; Sodhani, S.; Dong, J.; Pineau, J.; and Hamilton, W. L. 2019. CLUTRR: A Diagnostic Benchmark for Inductive Reasoning from Text. arXiv:1908.06177.

Song, K.; Tan, X.; Qin, T.; Lu, J.; and Liu, T.-Y. 2020. MPNet: Masked and Permuted Pre-training for Language Understanding. arXiv:2004.09297.

Srivastava, A.; Rastogi, A.; Rao, A.; Shoeb, A. A. M.; Abid, A.; Fisch, A.; Brown, A. R.; Santoro, A.; Gupta, A.; Garriga-Alonso, A.; and et al. 2023. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. arXiv:2206.04615.

Tiong, A. M. H.; Li, J.; Li, B.; Savarese, S.; and Hoi, S. C. 2022. Plug-and-Play VQA: Zero-shot VQA by Conjoining Large Pretrained Models with Zero Training. In Goldberg, Y.; Kozareva, Z.; and Zhang, Y., eds., *Findings of the ACL: EMNLP*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288.

Wang, P.-W.; Donti, P. L.; Wilder, B.; and Kolter, Z. 2019. SATNet: Bridging Deep Learning and Logical Reasoning Using a Differentiable Satisfiability Solver. In *ICML*.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. arXiv:2203.11171.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; and Zhou, D. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903.

Xu, Z.; Rawat, Y. S.; Wong, Y.; Kankanhalli, M.; and Shah, M. 2022. Don't Pour Cereal into Coffee: Differentiable Temporal Logic for Temporal Action Segmentation. In *NeurIPS*.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W. W.; Salakhutdinov, R.; and Manning, C. D. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. arXiv:1809.09600.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629.

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 *NeurIPS*.

Yin, Z.; Wang, Y.; Wu, Y.; Yan, H.; Hu, X.; Zhang, X.; Cao, Z.; Huang, X.; and Qiu, X. 2022. Rethinking Label Smoothing on Multi-hop Question Answering. arXiv:2212.09512.