Coupling Large Language Models with Logic Programming for Robust and General Reasoning from Text

Zhun Yang, Adam Ishay [1]
[1] Arizona State University
{zyang90,aishay}@asu.edu

Joohyung Lee [1,2]
[2] Samsung Research
joolee@asu.edu

Abstract

While large language models (LLMs), such as GPT-3, appear to be robust and general, their reasoning ability is not at a level to compete with the best models trained for specific natural language reasoning problems. In this work, we observe that a large language model could serve as a highly effective few-shot semantic parser that turns natural language sentences into a logical form that can be used as input to answer set programs, a logic-based declarative knowledge representation formalism. The combination leads to a robust and general system that works across multiple QA tasks without the need to retrain for new tasks. It requires only a few examples to direct an LLM to tune to an individual task, along with ASP knowledge modules that can be reused over multiple tasks. We demonstrate that this method achieves state-of-the-art performance on several NLP benchmarks, such as bAbI, StepGame, CLUTRR, and gSCAN, and also handles robot planning tasks that an LLM alone fails to solve.

1 Introduction

A typical way to handle a question-answering task is to train a neural network model on large training data and test it on similar data. Such models work well with linguistic variability and ambiguity, but often learn statistical features and correlations rather than true reasoning (Ruder, 2021), which makes them not robust, lack generalization, and difficult to interpret.

Alternatively, transformer-based large language models (LLMs) have recently shown wide success on many downstream tasks, demonstrating general reasoning capability on diverse tasks without being retrained. However, when we restrict our attention to individual NLP reasoning benchmarks, they usually do not perform as well as state-of-the-art models despite various efforts to improve accuracy through prompt engineering (Wei et al., 2022; Zhou et al., 2022).

Similarly, LLMs gained attention for plan generation for robots due to the rich semantic knowledge they acquired about the world (Ahn et al., 2022; Huang et al., 2022; Zeng et al., 2022). However, LLMs are known to perform shallow reasoning and cannot find complex plans (Valmeekam et al., 2022).

In another context, Nye et al. (2021) note that LLMs are good at generating system 1-like sequences, which are often inconsistent and incoherent. This is because LLMs are trained to predict subsequent words in a sequence and do not appear to have a deep understanding of concepts such as cause and effect, logic, and probability, which are important for reasoning.

Nevertheless, we note that the rich semantic knowledge that LLMs possess makes them effective general-purpose few-shot semantic parsers that can convert linguistically variable natural language sentences into atomic facts that serve as input to logic programs. We also note that the fully declarative nature of answer set programs (Lifschitz, 2008; Brewka et al., 2011) makes them a good pair with the LLM semantic parsers, providing interpretable and explainable reasoning on the parsed result of the LLMs using background knowledge. Combining large language models and answer set programs leads to an attractive dual-process, neuro-symbolic reasoning that works across multiple QA tasks without retraining for individual tasks.

We tested this idea with several NLP benchmarks, bAbI (Weston et al., 2016), StepGame (Shi et al., 2022), CLUTRR (Sinha et al., 2019), and gSCAN (Ruis et al., 2020), by applying the same dual-system model and achieved state-of-the-art performance in all of them. Furthermore, the high accuracy and transparency allow us to easily identify the source of errors, making our system a useful data set validation tool as well. In particular, we found a significant amount of errors in the original CLUTRR dataset that are hard to detect manually.

While the new version of GPT-3 (Brown et al.,

2020) (text-davinci-003) shows improvement over its predecessors, we observe that it also retains critical limitations. In the process, we develop prompt methods for semantic parsing to overcome some of them.

2 Preliminaries

2.1 Semantic Parsing and LLMs

Semantic parsing involves converting a natural language query or statement into a structured representation that a computer can understand and manipulate. Statistical methods have increased in popularity (Zelle and Mooney, 1996; Miller et al., 1996; Zettlemoyer and Collins, 2005; Wong and Mooney, 2007), and encoder-decoder models in particular have been widely used (Dong and Lapata, 2016; Jia and Liang, 2016; Kočiskỳ et al., 2016). However, these statistical methods require annotated input and output pairs. Furthermore, machine learning models too often fail to compositionally generalize to unseen data (Lake and Baroni, 2018).

More recently, pre-trained language models have been applied to semantic parsing tasks (Liu et al., 2021), such as generating SQL queries, SPARQL queries, logical forms, or programs, from natural language, together with fine-tuning or prompt-tuning on pre-trained models, such as BART, RoBERTa and GPT-2 (Chen et al., 2020a; Shin et al., 2021; Schucher et al., 2022). With larger pre-trained networks, such as GPT-3, prompting appears to yield a reasonable semantic parser without the need for fine-tuning (Shin et al., 2021; Drozdov et al., 2022).

Another line of related work is to apply pre-trained language models to relation extraction, the task of extracting semantic relationships from a text given two or more entities (Liu et al., 2021). Wang et al. (2022) do zero-shot relation extraction with pre-trained language models from the BERT family and GPT-2 variants. Zhou and Chen (2022) fine-tune BERT and RoBERTa models for the extraction of sentence-level relations. Chen et al. (2022) apply prompt-tuning to RoBERT_LARGE for relation extraction. Similar to ours, Agrawal et al. (2022) use a few-shot prompt with GPT-3 for the extraction of clinical relations.

2.2 Dual-System Model

There is increasing interest in combining neural and symbolic systems (Marcus, 2018; Lamb et al., 2020; Sarker et al., 2021). Such dual-system models achieved new state-of-the-art results in visual QA (Goldman et al., 2018; Sampat and Lee, 2018; Yi et al., 2019; Chen et al., 2020b; Ding et al., 2021). In the case of textual problems, to improve LLMs to generate more consistent and coherent sentences, Nye et al. (2021) suggest that generation be decomposed into two parts: candidate sentence generation by an LLM (system 1 thinking) and a logical pruning process (system 2 thinking) implemented via a separate symbolic module. They show that this neuro-symbolic, dual-process model requires fewer data to learn and achieves higher accuracy and better generalization. However, the main limitation of their work is that the symbolic module is manually constructed in Python code for the specific given task, which requires nontrivial efforts. Furthermore, their Python symbolic module is not readily reusable or composable. Also, the main results focused on the consistent text generation problem rather than evaluating the method on the datasets and comparing it with existing models. This is because writing the world models in Python is not a scalable method.

We follow the idea from (Nye et al., 2021) but adopt logic programming in place of the System 2 process. We argue that this combination is much more appealing than the one from (Nye et al., 2021), achieving the results that were only promised in (Nye et al., 2021) but not shown.

2.3 Answer Set Programming

Answer Set Programming (ASP) (Lifschitz, 2008; Brewka et al., 2011) is a declarative logic programming paradigm that has been shown to be effective in knowledge-intensive applications. It is based on the stable model (a.k.a. answer set) semantics of logic programs (Gelfond and Lifschitz, 1988), which could express causal reasoning, default reasoning, aggregates, and various other constraints. There are several efficient solvers, such as CLINGO, DLV, and WASP.

It is also known that classical logic-based action formalisms, such as the situation calculus (McCarthy and Hayes, 1969; Reiter, 2001) and the event calculus (Shanahan, 1995), can be formulated as answer set programs. For example, the following is one of the axioms in Discrete Event Calculus stating the commonsense law of inertia, saying that fluent F holds at the next time if there is no action affecting it.

```
% (DEC5)
holds_at(F,T+1) :- timepoint(T), fluent(F),
    holds_at(F,T), -released_at(F,T+1),
    not terminated(F,T).
```

Such a rule is universal and applies to almost all objects.

Answer set programs are also known to be elaboration tolerant (McCarthy, 1998). There has been work on modularizing knowledge bases in ASP, such as module theorem (Oikarinen and Janhunen, 2006; Babb and Lee, 2012) and knowledge modules (Baral et al., 2006). While ASP has been widely applied to many reasoning problems, it has not been considered as much in reasoning with natural language text because its input is expected to be strictly in a logical form, giving little flexibility in accepting diverse forms of natural language input.

We use CLINGO v5.6.0 as the answer set solver. For the language of CLINGO, we refer the reader to the textbook (Lifschitz, 2019) or the CLINGO manual.[1]

3 Our Method

We call our framework *[LLM]+ASP* where [LLM] denotes a large pre-trained network such as GPT-3, which we use as a semantic parser to generate input to the ASP reasoner. More specifically, we assume data instances of the form $\langle S, q, a\rangle$, where S is a context story in natural language, q is a natural language query associated with S, and a is the answer. We use an LLM to convert a problem description (that is, context S and query q) into atomic facts, which are inputted into the ASP solver, together with background knowledge encoded as ASP rules. The output of the ASP solver is interpreted as the prediction to this data instance. Figure 1 illustrates the inference flow in the context of StepGame. The pipeline is simple but general enough to apply to various tasks without retraining; it only requires replacing few-shot prompts to the LLM and the ASP background knowledge with those appropriate for the new tasks.

Combining LLMs and ASP this way allows symbolic reasoning to be robust to varying and raw textual input; the ASP knowledge modules are not affected by the various forms of text input that express the same facts. Our method does not require training datasets. Instead, a few examples that turn natural language sentences into atomic facts are sufficient to build a semantic parser thanks to learned representations in LLMs. ASP knowledge modules can be reused for different tasks.

[1] https://github.com/potassco/guide/releases.

3.1 Prompts for Fact Extraction

We use GPT-3 to extract atomic facts from the story and query. Most of the time, giving several examples yields accurate semantic parsing. The following is an example prompt for bAbI.

```
Please parse the following statements into facts
    . The available keywords are: pickup, drop,
    and go.
Sentence: Max journeyed to the bathroom.
Semantic parse: go(Max, bathroom).

Sentence: Mary grabbed the football there.
Semantic parse: pickup(Mary, football).
...
```

We find that GPT-3 is highly tolerable to linguistic variability. For example, in StepGame, GPT-3 can turn various sentences below into the same atomic fact `top_right("C","D").`

```
C is to the top right of D.
C is to the right and above D at an angle of
    about 45 degrees.
C is at a 45 degree angle to D, in the upper
    righthand corner.
C is directly north east of D.
C is below D at 2 o'clock.
```

In the experiments to follow, we find that the following strategy works well for fact extraction.

1. In general, we find that if the information in a story (or query) can be extracted independently, parsing each sentence separately (using the same prompt multiple times) typically works better than parsing the whole story.
2. There is certain commonsense knowledge that GPT-3 is not able to leverage from the examples in the prompt. In this case, detailing the missing knowledge in the prompt could work. For example, in StepGame, clock numbers are used to denote cardinal directions, but GPT-3 couldn't translate correctly even with a few examples in the prompt. It works after enumerating all cases ("12 denotes top, 1 and 2 denote top_right, 3 denotes right, . . . ") in the prompt.
3. Semantic parsing tends to work better if we instruct GPT-3 to use a predicate name that better reflects the intended meaning of the sentence. For example, "A is there and B is at the 5 position of a clock face" is better to be turned into `down_right(B,A)` than `top_left(A,B)` although, logically speaking, the relations are symmetric.

The complete set of prompts for semantic parsing is given in Appendix **??**.

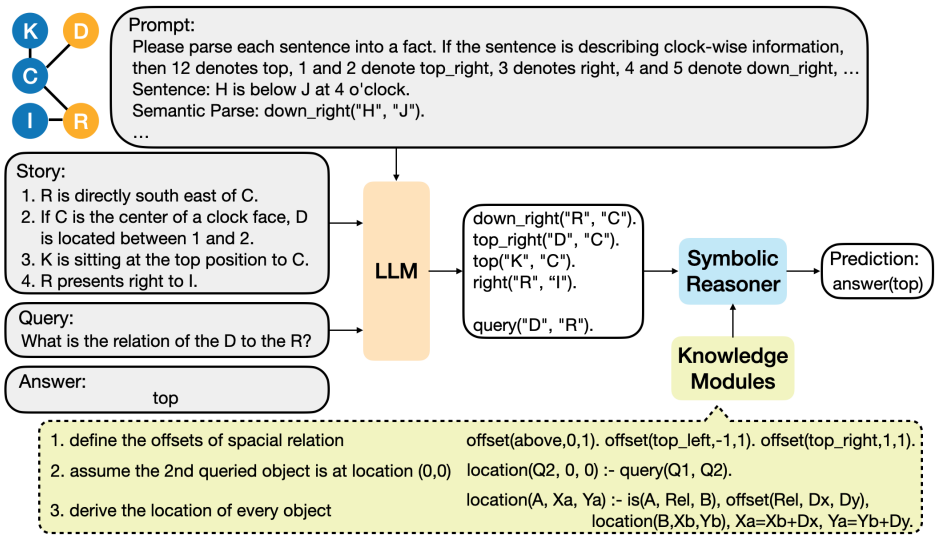

Figure 1: The GPT-3+ASP pipeline for the StepGame dataset.

3.2 Knowledge Modules

Instead of constructing a minimal world model for each task in Python code (Nye et al., 2021), we use ASP knowledge modules. While some knowledge could be lengthy to be described in English, it could be concisely expressed in ASP. For example, the **location** module contains rules for spatial reasoning in a 2D grid space and is used for bAbI, StepGame, and gSCAN. Below is the main rule in the **location** module that computes the location `(Xa,Ya)` of object `A` from the location `(Xb,Yb)` of object `B` by adding the offsets `(Dx,Dy)` defined by the spatial relation `R` between `A` and `B`.

```
location(A, Xa, Ya) :- location(B, Xb, Yb),
    is(A, R, B), offset(R, Dx, Dy),
    Xa=Xb+Dx, Ya=Yb+Dy.
```

The **location** module also includes 9 predefined offsets, e.g., `offset(left,-1,0)`, that can be used to model multi-hop spatial relations of objects or effects of a robot's moving in a 2D space. For example, queries in StepGame are about the spatial relation `R` of object `A` to `B`. Using the **location** module, one can fix `B`'s location to be `(0,0)` and compute the spatial relation `R` based on the location of `A` as follows.

```
location(B, 0, 0) :- query(A, B).
answer(R) :- query(A, B), location(A, X, Y),
    offset(R, Dx, Dy),
    Dx=-1: X<0; Dx=0: X=0; Dx=1: X>0;
    Dy=-1: Y<0; Dy=0: Y=0; Dy=1: Y>0.
```

The second rule above contains six *conditional literals* among which `Dx=-1:X<0` says that "`Dx` must be -1 if `X<0`." For example, if `A`'s location `(X,Y)` is `(-3,0)`, then `(Dx,Dy)` is `(-1,0)` and the answer `R` is `left`. Similar rules can also be applied to bAbI task 17, which asks if `A` is `R` of `B`.

In the above rules, the relation `R` in, e.g., `is(A,R,B)`, is a variable and can be substituted by any binary relation. Such high-order representation turns out to be quite general and applicable to many tasks that query relation or its arguments.

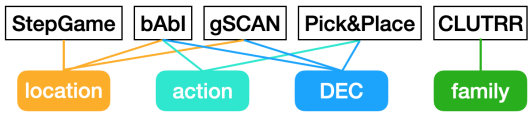

Figure 2: The knowledge modules at the bottom are used in each task on the top.

Figure 2 shows the knowledge modules used in this paper, where **DEC** denotes the Discrete Event Calculus axioms from (Mueller, 2006; Lee and Palla, 2012). In this section, we explained the main rules in the **location** module. The complete ASP knowledge modules are given in Appendix **??**.

4 Experiments

We apply the method in the previous section to four datasets.[2] Recall that we do few-shot in-context learning, but do not use the training set included in these datasets. We use the same pipeline as in Figure 1 with different prompts and knowledge modules for each dataset. More details about experiment settings are available in the appendix. The code is submitted as supplementary material.

[2]Due to space restriction, we put the experiments about Pick&Place in Appendix **??**.

Task	GPT-3(d3) Few-Shot	GPT-3(d3) CoT	GPT-3(d3) +ASP	STM(Le et al., 2020) (10k train)	QRN(Seo et al., 2017) (10k train)	(1k train)
1: Single supporting fact	98.4	97.3	100.0	100.0 ± 0.0	100.0	100.0
2: Two supporting facts	60.8	72.2	100.0	99.79 ± 0.23	100.0	99.3
3: Three supporting facts	39.6	54.1	100.0	97.87 ± 1.14	100.0	94.3
4: Two arg relations	60.4	72.7	100.0	100.0 ± 0.0	100.0	100.0
5: Three arg relations	88.2	89.1	99.8	99.43 ± 0.18	100.0	98.9
6: Yes/no questions	97.4	97.3	100.0	100.0 ± 0.0	100.0	99.1
7: Counting	90.6	88.6	100.0	99.19 ± 0.27	100.0	90.4
8: Lists/sets	96.2	97.1	100.0	99.88 ± 0.07	99.6	94.4
9 : Simple negation	98.4	98.2	100.0	100.0 ± 0.0	100.0	100.0
10: Indefinite knowledge	93.6	92.4	100.0	99.97 ± 0.06	100.0	100.0
11: Basic coreference	93.6	99.2	100.0	99.99 ± 0.03	100.0	100.0
12: Conjunction	88.6	88.8	100.0	99.96 ± 0.05	100.0	100.0
13: Compound coreference	98.4	97.3	100.0	99.99 ± 0.03	100.0	100.0
14: Time reasoning	78.0	91.5	100.0	99.84 ± 0.17	99.9	99.2
15: Basic deduction	57.0	95.0	100.0	100.0 ± 0.0	100.0	100.0
16: Basic induction	90.8	97.5	100.0	99.71 ± 0.15	100.0	47.0
17: Positional reasoning	66.0	70.8	100.0	98.82 ± 1.07	95.9	65.6
18: Size reasoning	89.8	97.1	100.0	99.73 ± 0.28	99.3	92.1
19: Path finding	21.0	28.7	100.0	97.94 ± 2.79	99.9	21.3
20: Agents motivations	100.0	100.0	100.0	100.0 ± 0.0	100.0	99.8
Average	80.34	86.18	**99.99**	99.85	99.70	90.1

Table 1: Test accuracy on 20 tasks in bAbI data

4.1 bAbI

The bAbI dataset (Weston et al., 2016) is a collection of 20 QA tasks that have been widely applied to test various natural language reasoning problems, such as deduction, path-finding, spatial reasoning, and counting. State-of-the-art models, such as self-attentive associative-based two-memory model (STM) (Le et al., 2020) and Query-Reduction networks (QRN) (Seo et al., 2017) achieve close to 100% accuracy after training with 10k instances while QRN's accuracy drops to 90% with 1k training instances.

We first designed two GPT-3 baselines, one with few shot prompts (containing a few example questions and answers) and the other with Chain-of-Thought (CoT) prompts (Wei et al., 2022), which state the relevant information to derive the answer.

We also apply GPT-3+ASP. For example, we use GPT-3 to turn "the kitchen is south of the bathroom" into an atomic fact `is(kitchen, southOf, bathroom)` by giving a few examples of the same kind. Regarding knowledge modules, Tasks 1–3, 6–9, 10–14, and 19 are about events over time and use the **DEC** knowledge module. Tasks 4, 17, and 19 require various domain knowledge modules such as **location** and **action** knowledge modules. The remaining tasks do not require domain knowledge and rely only on simple rules to extract answers from parsed facts.

Table 1 compares our method with the two GPT-3 baselines, as well as two state-of-the-art methods on bAbI datasets, STM and QRN. Interestingly, the new GPT-3, text-davinci-003 (denoted GPT-3 (d3)), with basic few-shot prompting achieves 80.34% accuracy, while CoT improves it to 86.18%. GPT-3(d3)+ASP achieves state-of-the-art performance on bAbI with 99.99% average performance among all tasks, producing only two answers that disagree with the labels in the dataset. It turns out that the two questions are malformed since the answers are ambiguous, and our model's answers can be considered correct.[3]

4.2 StepGame

Although bAbI has been extensively tested, it has several problems. Shi et al. (2022) note data leakage between the train and the test sets where named entities are fixed and only a small number of relations are used. Palm et al. (2018) point out that models do not need multi-hop reasoning to solve the bAbI dataset. To address the issues, Shi et al. (2022) propose the StepGame dataset. It is a contextual QA dataset in which the system is required to interpret a story S about spatial relationships among several entities and answers a query q about the relative position of two of those entities, as illustrated in Figure 1. Unlike the bAbI dataset, StepGame uses a large number of named entities, and requires multi-hop reasoning up to as many as 10 reasoning steps.

In the basic form of the StepGame dataset, each

[3] See Appendix **??** for the examples.

story consists of k sentences that describe k spatial relationships between $k+1$ entities in a chain-like shape. In this paper, we evaluate the StepGame dataset with noise, where the original chain is extended with noise statements by branching out with new entities and relations.

Similarly to bAbI, we designed two GPT-3 baselines and applied our method to the StepGame data set. More details on the prompts are available in Appendix **??**.

Method	k=1	k=2	k=3	k=4	k=5
RN	22.6	17.1	15.1	12.8	11.5
RRN	24.1	20.0	16.0	13.2	12.3
UT	45.1	28.4	17.4	14.1	13.5
STM	53.4	36.0	23.0	18.5	15.1
TPR-RNN	70.3	46.0	36.1	26.8	24.8
TP-MANN	85.8	60.3	50.2	37.5	31.3
SynSup	**98.6**	**95.0**	**92.0**	79.1	70.3
Few-Shot (d3)	55.0	37.0	25.0	30.0	32.0
CoT (d3)	61.0	45.0	30.0	35.0	35.0
GPT-3(c1)+ASP	44.7	38.8	40.5	58.8	62.4
GPT-3(d2)+ASP	92.6	89.9	89.1	**93.8**	**92.9**

Method	k=6	k=7	k=8	k=9	k=10
RN	11.1	11.5	11.2	11.1	11.3
RRN	11.6	11.4	11.8	11.2	11.7
UT	12.7	12.1	11.4	11.4	11.7
STM	13.8	12.6	11.5	11.3	11.8
TPR-RNN	22.3	19.9	15.5	13.0	12.7
TP-MANN	28.5	26.5	23.7	22.5	21.5
SynSup	63.4	58.7	52.1	48.4	45.7
Few-Shot (d3)	29.0	21.0	22.0	34.0	31.0
CoT (d3)	27.0	22.0	24.0	23.0	25.0
GPT-3(c1)+ASP	57.4	56.2	58.0	56.5	54.1
GPT-3(d2)+ASP	**91.6**	**91.2**	**90.4**	**89.0**	**88.3**

Table 2: Test accuracy on the StepGame test dataset, where (c1), (d2), and (d3) denote text-curie-001, text-davinci-002, and text-davinci-003 models, respectively

For each $k \in \{1, \ldots, 10\}$, the StepGame dataset with noise consists of 30,000 training samples, 1000 validation samples, and 10,000 test samples. To save the API cost for GPT-3, we only evaluated the two GPT-3 baselines on the first 100 test samples and evaluated our method on the first 1,000 test samples for each $k \in \{1, \ldots, 10\}$. Table 2 compares the accuracy of our method with the two baselines of GPT-3 and the current methods, i.e. RN (Santoro et al., 2017), RRN (Palm et al., 2018), UT (Dehghani et al., 2018), STM (Le et al., 2020), TPR-RNN (Schlag and Schmidhuber, 2018), TP-MANN (Shi et al., 2022), and SynSup (with pre-training on the SPARTUN dataset) (Mirzaee and Kordjamshidi, 2022). Surprisingly, the GPT-3 baselines could achieve accuracy comparable to other models (except for SynSup) for large k values. CoT does not always help and decreases the accuracy with big ks. This may be because there is a higher chance of making a mistake in a long chain of thought. GPT-3(d2)+ASP outperforms all state-of-the-art methods and the GPT-3 baselines by a large margin for $k = 4, \ldots, 10$. Although SynSup achieves a higher accuracy for $k = 1, 2, 3$, this is misleading due to errors in the dataset. As we analyze below, about 10.7% labels in the data are wrong. The SynSup training makes the model learn to make the same mistakes over the test dataset, which is why its performance looks better than ours.

The modular design of GPT-3+ASP enables us to analyze the reasons behind its wrong predictions. We collected the first 100 data instances for each $k \in \{1, \ldots, 10\}$ and manually analyzed the predictions on them.

Among 1000 predictions of GPT-3(d2)+ASP, 108 of them disagree with the dataset labels, and we found that 107 of those have errors in the labels. For example, given the story and question "*J and Y are horizontal and J is to the right of Y. What is the relation of the agent Y with the agent J?*", the label in the dataset is "right" while the correct relation should be "left".[4] Recall that our method is interpretable, so we could easily identify the source of errors.

4.3 CLUTRR

CLUTRR (Sinha et al., 2019) is a contextual QA dataset that requires inferring family relationships from a story. Sentences in CLUTRR are generated using 6k template narratives written by Amazon Mechanical Turk crowd-workers, and thus are more realistic and complex compared to those in bAbI and StepGame.

CLUTRR consists of two subtasks, *systematic generalization* that evaluates stories containing unseen combinations of logical rules (Minervini et al., 2020; Bergen et al., 2021) and *robust reasoning* that evaluates stories with noisy descriptions (Tian et al., 2021). Since we use ASP for logical reasoning, which easily works for any combination of logical rules, we focus on the robust reasoning task.

Table 3 compares our method with RN (Santoro

[4]The remaining disagreeing case is due to text-davinci-002's mistake. For the sentence, "*if E is the center of a clock face, H is located between 2 and 3*." text-davinci-002 turns it into "right(H, E)" whereas text-davinci-003 turns it into "top-right(H, E)" correctly. To save API cost for GPT-3, we did not re-run the whole experiments with text-davinci-003.

Method	CLU.	clean	supp.	irre.	disc.
RN	1.0	49	68	50	45
MAC	1.0	63	65	56	40
Bi-att	1.0	58	67	51	57
GSM	1.0	**68.5**	48.6	62.9	52.8
GPT-3(d3)+ASP	1.0	**68.5**	**82.8**	**74.8**	**67.4**
GPT-3(d3)+ASP	1.3	97.0	84.0	92.0	90.0

Table 3: Test accuracy on 4 categories in CLUTRR 1.0 and CLUTRR 1.3 datasets

et al., 2017), MAC (Hudson and Manning, 2018), BiLSTM-attention (Sinha et al., 2019), and GSM (Tian et al., 2021) on the original CLUTRR dataset, namely CLUTRR 1.0, in four categories of data instances: clean, supporting, irrelevant, and disconnected. Except for our method, all other models are trained on the corresponding category of CLUTRR training data. Although our method achieves similar or higher accuracies in all categories, they are still much lower than we expected.

We found that such low accuracy is due to the clear errors in CLUTRR, originating mostly from errors in the template narratives or the generated family graphs that violate common sense. The authors of CLUTRR recently published CLUTRR 1.3 codes to partially resolve this issue. [5] With the new code, we created a new dataset, namely CLUTRR 1.3, consisting of 400 data instances with 100 for each of the four categories. The last row in Table 3 shows that our method actually performs well on realistic sentences in CLUTRR. Indeed, with our method (text-davinci-003) on CLUTRR 1.3 dataset, 363 out of 400 predictions are correct, 16 are still wrong due to data mistakes (e.g., the label says "Maryann has an uncle Bruno" while the noise sentence added to the story is "Maryann told her son Bruno to give the dog a bath"), and 21 are wrong due to GPT-3's parsing mistakes (e.g., GPT-3 turned the sentence "Watt and Celestine asked their mother, if they could go play in the pool" into `mother("Watt", "Celestine")`. Since the sentences in CLUTRR 1.3 are more realistic than those of bAbI and StepGame, GPT-3 makes more mistakes even after reasonable efforts of prompt engineering. More details on data errors and GPT-3 errors are available in Appendix **??** and Appendix **??**.

We also evaluated our method on a simpler and cleaner variant of the CLUTRR data set, namely CLUTRR-S, that was used as a benchmark problem for a state-of-the-art neuro-symbolic approach DeepProbLog (Manhaeve et al., 2021). Table 4 compares the accuracy of our method and DeepProbLog in all 4 categories of test data. GPT-3(d3)+ASP achieves 100% accuracy, outperforming DeepProbLog without the need for training.

Method	clean	supp.	irre.	disc.
DeepProbLog	100	100	100	94
GPT-3(d2)+ASP	100	100	97	97
GPT-3(d3)+ASP	100	100	100	100

Table 4: Test accuracy on CLUTRR-S dataset

Remark: Due to the modular structure, our method could serve as a data set validation tool to detect errors in a dataset. We detected 107 wrong data instances in the first 1000 data in StepGame and 16 wrong data instances in the 400 data in CLUTRR 1.3.

4.4 gSCAN

The gSCAN dataset (Ruis et al., 2020) poses a task in which an agent must execute action sequences to achieve a goal (specified by a command in a natural language sentence) in a grid-based visual navigation environment. The dataset consists of two tasks, and we evaluate our method on the data splits from the compositional generalization task. There is one shared training set, one test set (split A) randomly sampled from the same distribution of the training set, and seven test sets (splits B to H) with only held-out data instances (i.e., not appearing in the training set) in different ways.

In the gSCAN dataset, each data instance is a tuple $\langle G, q, a \rangle$ where G is the grid configuration (in JSON format) describing the size of the gird, the location and direction of the agent, and the location and features of each object in the grid; q is a query (e.g., "pull a yellow small cylinder hesitantly"); and a is the answer in the form of a sequence of actions (e.g., "turn right, walk, stay, pull, stay, pull, stay"). For each data instance, we (i) use a Python script to extract atomic facts (e.g., `pos(agent,(2,3))`) from the grid configuration G; (ii) extract atomic facts from query q into atomic facts (e.g., `query(pull)`, `queryDesc(yellow)`, `while(hesitantly)`) using GPT-3; and (iii) predict the sequence of actions for this query using ASP. The details of the prompts are given in Appendix **??**.

Table 5 compares the accuracy of our method and the state-of-the-art methods, i.e., GECA (Ruis et al., 2020), DualSys (Nye et al., 2021) and Vil-

[5] https://github.com/facebookresearch/clutrr/tree/develop

Method	A	B	C	D
GECA	87.60	34.92	78.77	0.00
DualSys	74.7	81.3	78.1	0.01
Vilbert+CMA	99.95	99.90	99.25	0.00
GPT-3(c1)+ASP	98.30	**100**	**100**	**100**
GPT-3(d2)+ASP	**100**	**100**	**100**	**100**

Method	E	F	G	H
GECA	33.19	85.99	0.00	11.83
DualSys	53.6	76.2	0.0	21.8
Vilbert+CMA	99.02	99.98	0.00	22.16
GPT-3(c1)+ASP	**100**	**100**	**100**	**100**
GPT-3(d2)+ASP	**100**	**100**	**100**	**100**

Table 5: Test accuracy on the gSCAN dataset

bert+CMA (Qiu et al., 2021), on the gSCAN test dataset in eight splits. To save API cost for GPT-3, we only evaluated the first 1000 data instances of each split. With text-davinci-002, our method achieves 100% accuracy. GPT-3+ASP with text-curie-001, the accuracy is slightly lower, making 17 errors in split A. The errors are of two kinds. The language model fails to extract adverbs in the correct format for 11 data instances (e.g., GPT-3 responded `queryDesc(while spinning)` instead of `while(spinning)`) and didn't ground the last word in a query for 6 data instances (e.g., for query `walk to a small square`, GPT-3 missed an atomic fact `queryDesc(square)`). Once the parsed results are correct, ASP does not make a mistake in producing plans.

4.5 Findings

The following summarizes the findings of the experimental evaluation.

- Our experiments confirm that LLMs like GPT-3 are still not good at multi-step reasoning despite various prompts we tried. Chain-of-Thought is less likely to improve accuracy when a long chain of thought is required.
- On the other hand, LLMs are surprisingly good at turning a variety of expressions into a "canonical form" of information extraction. This in turn allows ASP knowledge modules to be isolated from linguistic variability in the input.
- Even for generating simple atomic facts, larger models tend to perform better. For example, in StepGame and gSCAN, text-curie-001 performs significantly worse compared to text-davinci-002 (Tables 2 and 5).
- The total amount of knowledge that needs to be encoded for all of the above datasets is not too large. This is in part due to the fact that GPT-3 "normalized" various forms of input sentences for ASP to process and that knowledge modules could be reused across different datasets.
- The modular design of our approach makes it possible to locate the root cause of each failed prediction in the training data and improve upon it. There are three sources of errors: semantic parsing in LLMs, symbolic constraints, and the dataset itself, and we can resolve the first two issues by improving the prompts and updating the constraints, respectively.
- Our framework could serve as a few-shot dataset justifier and corrector. Among all predictions by our method that do not align with the labels, almost all of them (with only a few exceptions discussed in the paper) are due to errors in the dataset.

5 Conclusion

Symbolic logic programming was considered to be limited in reasoning from text due to the inability to handle various and ambiguous linguistic expressions. Combining it with a large language model that learned distributed representations alleviates the problem. The method not only leads to the best accuracy but also the results are interpretable because the source of the errors can be identified. It is also general; by using pre-trained networks with few-shot prompts and reusable knowledge modules, adapting to a new domain does not require massive training.

The knowledge modules used in our experiments are reusable. For the above experiments, the modules are relatively simple to write, as are the prompts for parsing natural language for LLMs. However, acquiring this kind of knowledge on a massive scale is also an important line of research (Liu and Singh, 2004; Bosselut et al., 2019; Hwang et al., 2021) that needs to be combined. In addition, it is possible to use LLM's code generation capability (Chen et al., 2021) to generate logic program rules, which we leave for future work.

One may think that the logic rules are too rigid. However, there are many weighted or probabilistic rules that can be defeated (Richardson and Domingos, 2006; Fierens et al., 2013; Lee and Wang, 2018). They could be used for more realistic settings, but for the benchmark problems above, they were not needed.

6 Ethical Considerations

All datasets used in this paper are publicly available. For CLUTRR dataset, the gender information is essential to tell if, e.g., A is B's uncle or niece. We used GPT-3 to predict the genders of persons in each story. Since each story is systematically generated using sampled common first names and sampled sentence templates, it does not reveal any identity. As mentioned, the original CLUTRR dataset had some errors, and we describe carefully the codes and settings of the generated CLUTRR 1.3 dataset in Appendix **??**.

7 Limitations

The current work requires that knowledge modules be written by hand. Commonly used axioms, such as general knowledge like the commonsense law of inertia expressed by event calculus, can be reused easily, but there are vast amounts of other commonsense knowledge that are not easy to obtain. LLMs could be used to supply this information, but we have not tried. Knowledge graphs, such as ConceptNet (Liu and Singh, 2004), COMET (Bosselut et al., 2019) and ATOMIC (Hwang et al., 2021), can be utilized to populate ASP rules. Like code models, we expect that LLMs could generate ASP code, which we leave for future work.

Also, using large language models, despite various efforts, sometimes it is not understandable why GPT-3 does not behave as expected. And although not encountered, our model may have degraded performance when parsing more complex text passages.

References

Monica Agrawal, Stefan Hegselmann, Hunter Lang, Yoon Kim, and David Sontag. 2022. Large language models are few-shot clinical information extractors. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, page 1998–2022. Association for Computational Linguistics.

Michael Ahn, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. 2022. Do as I can, not as I say: Grounding language in robotic affordances. In *6th Annual Conference on Robot Learning*.

Joseph Babb and Joohyung Lee. 2012. Module theorem for the general theory of stable models. *Theory and Practice of Logic Programming*, 12(4-5):719–735.

Chitta Baral, Juraj Dzifcak, and Hiro Takahashi. 2006. Macros, macro calls and use of ensembles in modular answer set programming. In *International Conference on Logic Programming*, pages 376–390. Springer.

Leon Bergen, Timothy O'Donnell, and Dzmitry Bahdanau. 2021. Systematic generalization with edge transformers. *Advances in Neural Information Processing Systems*, 34:1390–1402.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for knowledge graph construction. In *Association for Computational Linguistics (ACL)*.

Gerhard Brewka, Ilkka Niemelä, and Miroslaw Truszczynski. 2011. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103.

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.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2022. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In *Proceedings of the ACM Web Conference 2022*, pages 2778–2788.

Xilun Chen, Asish Ghoshal, Yashar Mehdad, Luke Zettlemoyer, and Sonal Gupta. 2020a. Low-resource domain adaptation for compositional task-oriented semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5090–5100.

Zhenfang Chen, Jiayuan Mao, Jiajun Wu, Kwan-Yee Kenneth Wong, Joshua B Tenenbaum, and Chuang Gan. 2020b. Grounding physical concepts of objects and events through dynamic visual reasoning. In *International Conference on Learning Representations*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2018. Universal transformers. In *International Conference on Learning Representations*.

Mingyu Ding, Zhenfang Chen, Tao Du, Ping Luo, Josh Tenenbaum, and Chuang Gan. 2021. Dynamic visual reasoning by learning differentiable physics models from video and language. *Advances in Neural Information Processing Systems*, 34.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *54th Annual Meeting of the Association for Computational Linguistics*, pages 33–43. Association for Computational Linguistics (ACL).

Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. 2022. Compositional semantic parsing with large language models. *arXiv preprint arXiv:2209.15003*.

Daan Fierens, Guy Van den Broeck, Joris Renkens, Dimitar Shterionov, Bernd Gutmann, Ingo Thon, Gerda Janssens, and Luc De Raedt. 2013. Inference and learning in probabilistic logic programs using weighted boolean formulas. *Theory and Practice of Logic Programming*, pages 1–44.

Michael Gelfond and Vladimir Lifschitz. 1988. The stable model semantics for logic programming. In *Proceedings of International Logic Programming Conference and Symposium*, pages 1070–1080. MIT Press.

Omer Goldman, Veronica Latcinnik, Ehud Nave, Amir Globerson, and Jonathan Berant. 2018. Weakly supervised semantic parsing with abstract examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1809–1819.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and brian ichter. 2022. Inner monologue: Embodied reasoning through planning with language models. In *6th Annual Conference on Robot Learning*.

Drew A Hudson and Christopher D Manning. 2018. Compositional attention networks for machine reasoning. In *International Conference on Learning Representations*.

Jena D Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2021. (comet-) atomic 2020: On symbolic and neural commonsense knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6384–6392.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22.

Tomáš Kočiskỳ, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1078–1087.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882. PMLR.

Luis C Lamb, Artur Garcez, Marco Gori, Marcelo Prates, Pedro Avelar, and Moshe Vardi. 2020. Graph neural networks meet neural-symbolic computing: A survey and perspective. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4877–4884.

Hung Le, Truyen Tran, and Svetha Venkatesh. 2020. Self-attentive associative memory. In *International Conference on Machine Learning*, pages 5682–5691. PMLR.

Joohyung Lee and Ravi Palla. 2012. Reformulating the situation calculus and the event calculus in the general theory of stable models and in answer set programming. *Journal of Artificial Inteligence Research (JAIR)*, 43:571–620.

Joohyung Lee and Yi Wang. 2018. Weight learning in a probabilistic extension of answer set programs. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 22–31.

Vladimir Lifschitz. 2008. What is answer set programming? In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1594–1597. MIT Press.

Vladimir Lifschitz. 2019. *Answer set programming*. Springer Heidelberg.

Hugo Liu and Push Singh. 2004. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys (CSUR)*.

Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. 2021. Neural probabilistic logic programming in deepproblog. *Artificial Intelligence*, 298:103504.

Gary Marcus. 2018. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.

John McCarthy. 1998. Elaboration tolerance. In *Working Papers of the Fourth Symposium on Logical Formalizations of Commonsense Reasoning*.

John McCarthy and Patrick Hayes. 1969. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, Edinburgh.

Scott Miller, David Stallard, Robert Bobrow, and Richard Schwartz. 1996. A fully statistical approach to natural language interfaces. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 55–61.

Pasquale Minervini, Sebastian Riedel, Pontus Stenetorp, Edward Grefenstette, and Tim Rocktäschel. 2020. Learning reasoning strategies in end-to-end differentiable proving. In *International Conference on Machine Learning*, pages 6938–6949. PMLR.

Roshanak Mirzaee and Parisa Kordjamshidi. 2022. Transfer learning with synthetic corpora for spatial role labeling and reasoning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, page 6148–6165. Association for Computational Linguistics.

Erik Mueller. 2006. *Commonsense reasoning*. Elsevier.

Maxwell Nye, Michael Tessler, Josh Tenenbaum, and Brenden M Lake. 2021. Improving coherence and consistency in neural sequence models with dual-system, neuro-symbolic reasoning. *Advances in Neural Information Processing Systems*, 34:25192–25204.

Emilia Oikarinen and Tomi Janhunen. 2006. Modular equivalence for normal logic programs. In *17th European Conference on Artificial Intelligence(ECAI)*, pages 412–416.

Rasmus Palm, Ulrich Paquet, and Ole Winther. 2018. Recurrent relational networks. In *Proceedings of Advances in Neural Information Processing Systems*, pages 3368–3378.

Linlu Qiu, Hexiang Hu, Bowen Zhang, Peter Shaw, and Fei Sha. 2021. Systematic generalization on gscan: What is nearly solved and what is next? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2180–2188.

Raymond Reiter. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning*, 62(1-2):107–136.

Sebastian Ruder. 2021. Challenges and Opportunities in NLP Benchmarking. http://ruder.io/nlp-benchmarking.

Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M Lake. 2020. A benchmark for systematic generalization in grounded language understanding. *Advances in neural information processing systems*, 33:19861–19872.

Shailaja Sampat and Joohyung Lee. 2018. A model-based approach to visual reasoning on cnlvr dataset. In *Sixteenth International Conference on Principles of Knowledge Representation and Reasoning*.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976.

Md Kamruzzaman Sarker, Lu Zhou, Aaron Eberhart, and Pascal Hitzler. 2021. Neuro-symbolic artificial intelligence. *AI Communications*, pages 1–13.

Imanol Schlag and Jürgen Schmidhuber. 2018. Learning to reason with third order tensor products. *Advances in neural information processing systems*, 31.

Nathan Schucher, Siva Reddy, and Harm de Vries. 2022. The power of prompt tuning for low-resource semantic parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 148–156.

Min Joon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Query-reduction networks for question answering. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Murray Shanahan. 1995. A circumscriptive calculus of events. *Artif. Intell.*, 77(2):249–284.

Zhengxiang Shi, Qiang Zhang, and Aldo Lipani. 2022. Stepgame: A new benchmark for robust multi-hop spatial reasoning in texts. *Association for the Advancement of Artificial Intelligence*.

Richard Shin, Christopher Lin, Sam Thomson, Charles Chen Jr, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7699–7715.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L Hamilton. 2019. Clutrr: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4506–4515.

Jidong Tian, Yitian Li, Wenqing Chen, HE Hao, and Yaohui Jin. 2021. A generative-symbolic model for logical reasoning in nlu. In *Is Neuro-Symbolic SOTA still a myth for Natural Language Inference? The first workshop*.

Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2022. Large language models still can't plan (a benchmark for LLMs on planning and reasoning about change). In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

Chenguang Wang, Xiao Liu, and Dawn Song. 2022. Ielm: An open information extraction benchmark for pre-trained language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, page 8417–8437. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomás Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967.

Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum. 2019. CLEVRER: Collision events for video representation and reasoning. In *ICLR*.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

Andy Zeng, Adrian Wong, Stefan Welker, Krzysztof Choromanski, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, et al. 2022. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 658–666.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Chi. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Wenxuan Zhou and Muhao Chen. 2022. An improved baseline for sentence-level relation extraction. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 161–168, Online only. Association for Computational Linguistics.