Detecting Misleading Information with LLMs and Explainable ASP

Quang-Anh Nguyen[1], Thu-Trang Pham[1], Thi-Hai-yen Vuong[1], Van-Giang Trinh[2] and Nguyen Ha Thanh[3,4]

[1]*VNU University of Engineering and Technology, Hanoi, Vietnam*

[2]*Inria Saclay, EP Lifeware, Palaiseau, France*

[3]*Center for Juris-Informatics, ROIS-DS, Tokyo, Japan*

[4]*Research and Development Center for Large Language Models, NII, Tokyo, Japan*

{*21020116, 21020248, yenvth*}*@vnu.edu.vn, van-giang.trinh@inria.fr, nguyenhathanh@nii.ac.jp*

Keywords: LLM, ASP, Explainability, Misleading Information Detection.

Abstract: Answer Set Programming (ASP) is traditionally constrained by predefined rule sets and domains, which limits the scalability of ASP systems. While Large Language Models (LLMs) exhibit remarkable capabilities in linguistic comprehension and information representation, they are limited in logical reasoning which is the notable strength of ASP. Hence, there is growing research interest in integrating LLMs with ASP to leverage these abilities. Although many models combining LLMs and ASP have demonstrated competitive results, issues related to misleading input information which directly affect the incorrect solutions produced by these models have not been adequately addressed. In this study, we propose a method integrating LLMs with explainable ASP to trace back and identify misleading segments in the provided input. Experiments conducted on the CLUTRR dataset show promising results, laying a foundation for future research on error correction to enhance the accuracy of question-answering models. Furthermore, we discuss current challenges, potential advancements, and issues related to the utilization of hybrid AI systems.

1 INTRODUCTION

Answer Set Programming (ASP) is a declarative programming paradigm, primarily utilized in solving combinatorial problems (Gelfond and Lifschitz, 1988; Eiter et al., 2009). It is rooted in logic programming and nonmonotonic reasoning, offering a powerful framework for knowledge representation and reasoning. The origin of ASP can be traced back to the stable model semantics of logic programming, introduced by Gelfond and Lifschitz in the late 1980s. Over years, it has evolved through several significant stages, adapting and integrating more sophisticated solving techniques from the fields of SAT solving and constraint programming. This convergence of methodologies has not only enhanced the efficiency and scalability of ASP solvers but also broadened their applicability across various domains including AI, bioinformatics, and complex systems analysis (Erdem et al., 2016; Trinh et al., 2024).

Large Language Models (LLMs), such as GPT-4 (Achiam et al., 2023), demonstrate exceptional capabilities in understanding and generating natural language, enabling them to act as versatile tools across various applications by converting ambiguous human language into structured, computable formats. This potential is particularly impactful for Answer Set Programming (ASP), a declarative programming paradigm designed to tackle complex combinatorial problems. In particular, integrating LLMs into ASP systems allows for automatic rule generation and dynamic fact integration, enhancing the flexibility and efficiency of ASP solvers (Ishay et al., 2023). Furthermore, LLMs serve as an interface between natural language and ASP's logical reasoning framework, facilitating the translation of real-world problems into formal representations and enabling ASP to adapt dynamically to contextual requirements (Nguyen et al., 2023a; Rajasekharan et al., 2023). This innovation paves the way for more intelligent, interactive, and autonomous computational logic systems, significantly expanding ASP's applicability in fields like legal reasoning, healthcare, and beyond.

Although the integration of LLMs with ASP has significantly improved accuracy and applicability across various problem domains, erroneous or ambiguous input information also requires attention, as it can directly lead to incorrect solutions. This phe-

Nguyen, Q.-A., Pham, T.-T., Vuong, T.-H.-Y., Trinh, V.-G. and Thanh, N. H.
Detecting Misleading Information with LLMs and Explainable ASP.
DOI: 10.5220/0013357400003890

In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 3*, pages 1327-1334
ISBN: 978-989-758-737-5; ISSN: 2184-433X

nomenon may arise from initial inaccuracies or the intrinsic complexity of natural language. Identifying and addressing such flawed information is significant for the derivation of an accurate and reliable solution, particularly in domains that demand high levels of precision, such as law, healthcare, and so on.

Motivated by such problems, in this paper, we aim to leverage the language comprehension capabilities of LLMs, combined with the inference power and the explanatory features of explainable ASP to address the challenges of detecting incorrect information. We introduce our framework, which integrates LLMs with ASP to perform two main steps: recognizing questions containing conflict answers and analysing these questions to identify misleading information.

The remainder of the paper is organized as follows. Section 2 recalls the foundational background. The related work section follows, examining existing literature on ASP, LLMs, and their intersections, identifying the gaps our study addresses. The core of our work, presented in Section 3, introduces our framework, highlighting our novel approach for the incorrect information detection. Section 4 shows the experimental results demonstrating the efficiency of our approach. Subsequent discussions critique limitations and implications, as well as suggest solutions to overcome them. Finally, Section 6 concludes the paper.

2 PRELIMINARIES

The fundamental knowledge on which our system relies will be presented in detail, including ASP, LLMs and their combinations.

2.1 Answer Set Programming

Answer Set Programming (ASP) has a rich heritage in logic programming, drawing from various declarative programming paradigms. ASP is conceptually connected to earlier forms of logic programming and computational logic, integrating principles from Constraint Logic Programming (CLP) and Abductive Logic Programming (ALP), among others (Eiter et al., 2009).

Early foundational work by (McCarthy, 1959) introduced concepts that would later influence the design of logic programming languages like Prolog, paving the way for declarative programming paradigms. Constraint Logic Programming was further developed as a powerful paradigm, enabling programming with explicit constraints. Marriott and Stuckey's book (Marriott and Stuckey, 1998) provides a comprehensive introduction to this field, explaining key principles and applications that overlap with ASP concerning solving combinatorial problems.

The developments and practical applications of ASP have been showcased in several ASP competitions, which compare system capabilities and performance across a variety of benchmark problems. (Calimeri et al., 2014) details the Third Open Answer Set Programming Competition, emphasizing the growing maturity and capability of ASP systems in handling complex, real-world problems across multiple domains.

Each of these contributions underscores different facets of ASP, reflecting its theoretical depth and versatility as a paradigm for declarative programming. As indicated by (Calimeri et al., 2014), ASP continues to evolve, interfacing with related fields and expanding its applicative reach, thereby confirming its significance and potential within the broader landscape of computational logic and artificial intelligence.

2.2 Large Language Models

Large Language Models (LLMs) have achieved significant milestones since the introduction of the Transformer architecture, which is known for its reliance on attention mechanisms (Vaswani et al., 2017). This transformative approach moved away from recurrent and convolutional structures, demonstrating that attention alone could yield high performance in sequence transduction tasks like translation. In addition to sequence transduction, LLMs have also shown remarkable capabilities in language understanding. This is evident in tasks such as classification (Liga and Robaldo, 2023), sentiment analysis (Kheiri and Karimi, 2023), and information extraction (Zin et al., 2023), where the models can accurately comprehend text and extract relevant information. Moreover, LLMs have proven effective in question answering systems, where they can understand a question posed in natural language and provide a concise and accurate answer (Phi et al., 2020).

Subsequent advancements in LLMs have shown remarkable effectiveness across a range of benchmarks. These include GPT-3, which demonstrated the power of autoregressive language models in few-shot settings (Brown, 2020), and its successor GPT-4, which has expanded capabilities to handle multimodal inputs and achieved human-level performance on various professional exams (Achiam et al., 2023). Despite these advances, logical reasoning remains a challenging area for LLMs, often referred to as their "Achilles' heel." The core issue is that these models

typically generate outputs based on patterns learned from extensive data, which may not always align with the rigorous logical reasoning required in domains like law, medicine, and science. While there are attempts to tailor LLMs to better handle logical reasoning through specific training strategies (Nguyen et al., 2023a), the inherent limitations in their ability to genuinely deduce or reason abstractly remain evident. Studies have shown that even though LLMs perform well with legal text processing tasks, they struggle significantly with tasks demanding high logical inference such as abductive reasoning (Nguyen et al., 2023b).

2.3 Combining Large Language Models and Answer Set Programming

The integration of LLMs with ASP represents a promising avenue in the development of advanced computational logic systems that leverage both symbolic and sub-symbolic approaches. This neuro-symbolic integration seeks to harness the vast background knowledge and natural language processing capabilities of LLMs alongside the rigorous, rule-based reasoning power of ASP.

(Yang et al., 2023) present NeurASP that marries neural network outputs with ASP's symbolic reasoning capabilities. By treating outputs from neural networks as probabilistic facts within ASP, NeurASP facilitates a seamless blend, enabling more nuanced decision-making processes that incorporate both learned patterns and explicitly defined rules. This approach enhances the interpretability and reliability of neural network predictions by tethering them to logical constraints and reasoning processes in ASP.

Similarly, the work of (Bauer et al., 2023) exemplifies the application of this hybrid approach in the context of Visual Question Answering (VQA). Their neuro-symbolic system combines neural networks for image processing and Optical Character Recognition (OCR) with ASP for high-level reasoning about graph-structured data. The integration proves particularly effective in domains where input data are complex and require both perceptual understanding and logical inference, achieving substantial accuracy improvements and showcasing the adaptability of LLMs when guided by symbolic logic frameworks.

The STAR framework presented by (Rajasekharan et al., 2023) represents another significant step in this integration. STAR leverages LLMs for extracting structured knowledge from natural language, which is then reasoned over using goal-directed ASP to yield reliable conclusions that are explainable in nature. This method particularly enhances performance in Natural Language Understanding (NLU) tasks requiring deep reasoning-areas where LLMs typically falter.

3 METHOD

Our goal in this research concentrates on detecting contradictory or ambiguous information in natural language stories that requires logical reasoning to take into account. We propose a method integrating LLMs with ASP and their explainable extension. This approach combines the power of language understanding from LLMs and the inference ability along with the explication of explainable ASP to tackle logical problems.

Figure 1 presents the overview of our framework to highlight problematic components and point out errorneous reasons from a human language story in CLUTRR dataset (Sinha et al., 2019). In particular, Yang et al proposed applying LLMs to convert the relations between people in the story into symbolic expressions. GPT-4 (Achiam et al., 2023) is utilized to complete the task of converting verbal context into semantic parse that can cooperate with foundation rules for ASP solvers. After that, an ASP solver will return an answer set that contains all possible relationships of people that can be inferred. We then classify whether conflicts appear in interpersonal relationships, or in other words, whether there is more than one relation between two distinct individuals. Subsequently, the semantic parse of this story is combined with a set of constructed rules based on foundational rules to collect explanations from the explainable ASP component, which traces back the rules and logical symbols used in the judgment. With these explanations, not only are the conflicting pairs identified, but also the reasons leading to them, particularly the elements in the semantic parse, are located. Afterwards, using natural language processing techniques, misleading information in the given story, as well as the types of errors causing these issues, are identified.

The incorporation of LLMs into the proposed framework offers significant flexibility, as it eliminates the need for pre-training datasets. By simply providing a few examples as few-shot prompts to the LLM, the system can effectively adjust to different reasoning contexts without extensive reconfiguration. This methodology not only broadens the applicability of the framework but also simplifies its deployment across diverse scenarios. Comprehensive details about the components of this framework, including its setup and operational mechanisms, will be elaborated in the subsequent sections.

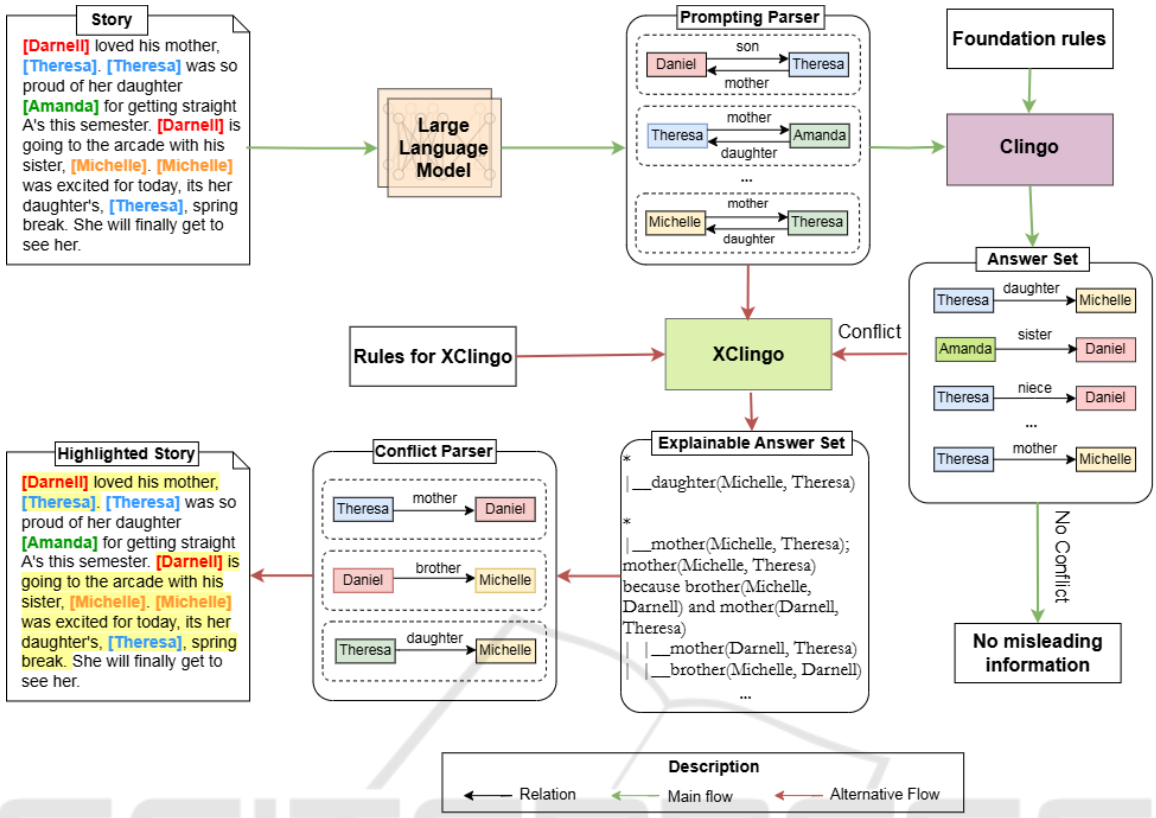

Figure 1: Overview of our framework.

3.1 Semantic Parsing with LLM

Before operating the ASP solvers (`Clingo` and `XClingo`), we first extract atomic facts from a given story to build a foundation for advanced decision-making processes. To facilitate this, we deploy an innovative approach we describe as the **Prompting Parser** which integrates with GPT-4 (Achiam et al., 2023) in a few-shot learning framework. Principally, this parser is structured into two core components: the narrative story which provides a rich, contextual background in natural language, and the precise, structured semantic parse which aims to distill the narrative into formal logical statements.

Specifically, the process begins by feeding a descriptive story into GPT-4, equipped with the capability to comprehend and interpret nuanced language through its advanced language models. Following the natural language input, our task is to enable the model to transform this narrative into a set of succinct, logical relations and entities which represent the core facts and relationships embedded within the story.

Prompting Parser:

```
% Sample 1
Story:[Michelle] was excited for today, its
her daughter's, [Theresa], spring break. She
will finally get to see her. [Michael] was
busy and sent his wife, [Marlene], instead.
[Kristen] loved to care for her newborn child
[Ronald]. [Eric]'s son is [Arthur].
Semantic Parse: daughter("Michelle",
"Theresa"). mother("Theresa", "Michelle").
wife("Michael", "Marlene"). hus-
band("Marlene", "Michael") child("Kristen",
"Ronald"), mother("Kristen", "Ronald").
son("Eric", "Arthur"), father("Arthur",
"Eric").

% Sample 2
Story:[Vernon] was present in the delivery
room when his daughter [Raquel] was born,
but when his daughter [Constance] was born
he was too sick. [Vernon] and his daughter
[Margaret] went to the movies.[Constance],
[Margaret]'s sister, had to stay home as she
was sick.
Semantic Parse: daughter("Vernon",
"Raquel"). father("Raquel", "Ver-
non"). daughter("Vernon", "Con-
stance"). father("Constance", "Ver-
non"). daughter("Vernon", "Margaret").
father("Margaret", "Vernon"). sis-
ter("Margaret", "Constance").
...
```

The story serves as the input context written in natural language. The semantic parser represents the logical outcome that we aim for LLMs to generate. For example, a logical form *daughter("Michelle", "Theresa")* means that Michelle has a daughter called Theresa. Throughout the experimental process, we have observed that LLMs demonstrate proficiency in understanding natural language and converting it into logical representations effectively. However, alongside these strengths, it still exhibits some errors in the process of generating logical representations, such as syntactic errors and missing or redundant information in each component. For cases where the parser generates incorrect logical results, we iterate through this step until we obtain the correct logical representations.

3.2 Collecting Answers with ASP

After obtaining the logical representations from the "**Semantic Parse**", we feed them into the ASP solver `Clingo` (Gebser et al., 2019) built upon the predefined correct rule set (called the set of foundation rules) to infer a comprehensive list of all potential relationships between individuals. A rule is constructed from a head and two predicates with the form *relation("first people", "second people")*. For instances, a rule *son(A, C) :- son(A, B), brother(B, C), A!=C.* indicates that if A has a son B, B has a brother C and A is different from C then A also has a son C. Typically, for stories with accurate, clear, and complete information, there is only one unique relationship between two individuals. However, if the story contains misleading information, multiple relationships between the two individuals may exist. Based on this idea, we identify all questions with conflicting answers that involve more than one relation between two distinct individuals. These questions are then analyzed to trace the incorrect information within the story.

3.3 Using Explainable ASP for Error Detection

In order to receive clear clarifications for the answers responded by the ASP solver `Clingo`, an explainable system for ASP with the name `XClingo` (Cabalar et al., 2020) is used. To take advantage of this extension, a rule set based on the original foundation rules for `XClingo` is designed. Specifically, trackable rules are set with *!trace_rule* while trackable atoms are combined with *!track* to write particular explanations. The above information will be recorded during the reasoning process to provide the answer. The answer is then generated with *!show_trace* command. For example, the rule *son(A, C) :- son(A, B), brother(B, C), A!=C.* is converted to *%!trace_rule {{"son(%, %) because son(%, %) and brother(%, %)", A, C, A, B, B, C}} son(A, C) :- son(A, B), brother(B, C), A!=C.* while the atom *son(A, C).* is turned into *%!trace "son(%, %)", A, C son(A, C).*. Then *%!show_trace son(A, C).* is applied to ensure that the answers with relation between 2 people is son are presented. The integration of `XClingo` with the logical forms in the semantic parser and the constructed rules results in output with a list of trees. An example of a list of tree output from `XClingo` can be seen below.

An example output of `XClingo`

```
*
|__daughter(Michelle, Theresa)

*
|__mother(Michelle, Theresa);mother(Michelle,
Theresa) because brother(Michelle, Darnell)
and mother(Darnell, Theresa)
| |__mother(Darnell, Theresa)
| |__brother(Michelle, Darnell)

*
|__mother(Michelle, Theresa);mother(Michelle,
Theresa) because sister(Michelle, Amanda) and
mother(Amanda, Theresa)
| |__mother(Amanda, Theresa)
| |__sister(Michelle, Amanda);sister(Michelle,
Amanda) because brother(Michelle, Darnell)
and sister(Darnell, Amanda)
| | |__sister(Darnell, Amanda);sister(Darnell,
Amanda) because mother(Darnell, Theresa) and
daughter(Theresa, Amanda)
| | | |__daughter(Theresa, Amanda)
| | | |__mother(Darnell, Theresa)
| | |__brother(Michelle, Darnell)
...
```

As can be seen, the root of tree contains the answers after a process of inference using rules and symbols from the semantic parse. A node can be divided into two types: *inferred* node and *original* node. An inferred node consists of two parts: the main answer with relation between people and the explanation for generating this answer. This node also comprises two child nodes as the materials of creation. The other type of node, or simply a *leave*, has no child node and only has the main answer part. In particular, this node is directly obtained from the semantic parse. Generally, all nodes are collected after the progression of leaves, hence the work of tracking back to leaves can be implemented to explore the elements that yield friction. Therefore, we propose an algorithm (see Algorithm 1) to find leaves that raise conflicted main answers at root of trees in an attempt to emphasize inaccurate components.

Algorithm 1: Procedure for collecting conflict leaves and main answers.

Data: List of trees $\mathcal{T}$ received from `XClingo`; *max_length*;
Result: List of conflict leaves $\mathcal{L}$;
List of conflict main answer $\mathcal{A}$;
List of conflict leaves $\mathcal{L} \leftarrow \emptyset$;
List of conflict main answer $\mathcal{A} \leftarrow \emptyset$;
$\mathcal{T} \leftarrow$ sorted trees $\mathcal{T}$ by height
if *length of* $\mathcal{T} \geq$ *max_length* **then**
  find the first two distinct trees $\mathcal{T}[i]$, $\mathcal{T}[j]$ with conflict main answers at root
  $\mathcal{L} \leftarrow$ leaves of $\mathcal{T}[i]$ and $\mathcal{T}[j]$
  $\mathcal{A} \leftarrow$ main answers at root of $\mathcal{T}[i]$ and $\mathcal{T}[j]$
else
  find all two distinct trees $\mathcal{T}[i]$, $\mathcal{T}[j]$ with conflict main answers at root
  $\mathcal{L} \leftarrow$ leaves of $\mathcal{T}[i]$ and $\mathcal{T}[j]$
  $\mathcal{A} \leftarrow$ main answers at root of $\mathcal{T}[i]$ and $\mathcal{T}[j]$
end

In Algorithm 1, a hyperparameter *max_length* is defined to prevent the process of methods from consuming a large amount of time and overloaded with acquired data. In our experiments, this parameter is set to 5000 to ensure that conflicts are gathered, as well as duplicates and corruption are avoided.

After achieving a list of leaves causing opposing main answer at root of trees, it is noticable that the logical forms in the semantic parse which bring failure are also located. In addition, the symbols that do not appear in the list of trackable entities for the explainable system are also gathered. We merge both types of mentioned in the semantic parse to spot misleading information in the given story. Indeed, the story is split into an array of sentences, afterwards sentences which contain both two people in a mistaken elements of the semantic parse are appended. In case only one person appears in a sentence, the following sentences are examined until the remaining person is found. For each set of leaves, a set of corresponding sentences is obtained. This set of sentences is added to the final set of conflict sentences only if there are no subsets (except an empty set) of this set in the current array of conflict sentences. Moreover, with the collection of conflict main answers at root, the type of errors in the story can also be considered such as conflict in relation between people, ambiguous information or unknown gender.

4 EXPERIMENT

We tested the proposed method on the CLUTRR dataset (Sinha et al., 2019). The following subsections illustrate the setting and results of our experiment, respectively.

4.1 Experimental Setting

We use the CLUTRR dataset that consists of stories related to hypothetical family relationships, aiming to identify the connection between two family members not directly mentioned (Sinha et al., 2019). Addressing this problem involves finding the contrary relations between people and exploring misunderstood information in stories.

To perform these tasks, we chose the CLUTRR dataset's supporting category where the stories contain additional data beside main information for reasoning and predicting the relation between two distinct members, thus resulting in some cases of uncertainty. The training data of this category comprises 5,107 samples which are thoroughly investigated to realize any failures or errors in stories. Throughout the experimental process, our method only applies few-shot prompting and does not involve training. Starting with the natural language to logic conversion phase, the GPT-4 model is leveraged with *max_tokens* of 4096 to recognize verbal context and generate structured output from few-shot learning. In the next step, a state-of-the-art ASP solver `Clingo` (Gebser et al., 2019) (version 5.7.1) is employed with default reasoning mode to optimize the inference process. Since the results of `Clingo` are plain answers of relation between family members, an external system `XClingo` (version 2.0b12) is used to trace the path of reasoning. To prevent `Xclingo`'s explanations from falling into an infinite loop, we are currently limiting the number of trees returned in output to 100000. All experiments were conducted in a `Conda` environment in `Ubuntu` for convenient and efficient implementation.

4.2 Experimental Results

Since all the reasons leading to an answer must be returned, runtime when using `Xclingo` increases significantly compared to utilizing `Clingo` alone. In the worst case with the above settings, the time to complete processing a sample is about 10 minutes, significantly slower when using only `Clingo` at 0.08 seconds. After performing several experiments, misleading information in the story is identified with red highlights while their types of errors are also divided. In

the 5,107 samples, 815 were found to have conflicts in relation between individuals in an answer set responded by `Clingo`. Some instances and their related errors are illustrated below.

Conflict in relations between family members

Story: [Darnell] loved his mother, [Theresa]. [Theresa] was so proud of her daughter [Amanda] for getting straight A's this semester. [Darnell] is going to the arcade with his sister, [Michelle]. [Michelle] was excited for today, its her daughter's, [Theresa], spring break. She will finally get to see her.
Conflict Parse: mother(Darnell, Theresa), brother(Michelle, Darnell), daughter(Michelle, Theresa)
Conflict Answer: mother(Michelle, Theresa), daughter(Michelle, Theresa)

In this case, it is clearly observed that Darnell has a mother whose name is Theresa and a sister called Michelle. With simple inference, we can conclude that Michelle is the daughter of Theresa. However, the given context points out that Theresa is Michelle's daughter giving contrast in the relation between two family members.

Unclear gender

Story: [Vernon] took his brother [Henry] out to get drinks after a long work week. [Vernon] and his sister [Robin] went to brunch today at the new diner. [Henry] is taking his daughter [Verdie] out for lunch at her favorite restaurant. [Robin] was playing in the sandbox with her brother [Henry].
Conflict Parse: sister(Vernon, Robin), brother(Henry, Vernon), sister(Robin, Vernon)
Conflict Answer: brother(Robin, Vernon), sister(Robin, Vernon)

In this instance, it is noticable that the issue of unknown gender happened with Vernon. As a result, during the parsing and reasoning phases, dividing this person to male or female is unclear. The conflict answer yields the result that Robin can also have a sister or a brother whose name is Vernon, which does not meet the requirement to find only one connection between two different family members.

Reflexive relationship

Story: [Eddie] and his sister [Amanda] got their mother [Amanda] a new computer for her birthday. She really liked it. [Amanda]'s father is named [Henry]. [Henry] is taking his son [Eddie] on a camping trip for the weekend. [Henry] went to the store with his brother [Vernon].
Conflict Parse: brother(Eddie, Amanda), mother(Eddie, Amanda)
Conflict Answer: brother(Eddie, Amanda), mother(Eddie, Amanda)

The story contains a fatal error when Amanda has a mother-daughter relationship with herself. This mistake also leads to misunderstanding and failure when convert natural language to logical forms. In particular, the generative model has encountered conference and inferred that Amanda is a brother of Eddie, which is opposite from the correct relation.

Ambiguous information

Story: [Ronald] was visiting his grandparents' house and saw [Karen] first. [Kristen], [Patty]'s mother, was eager to plan a trip with her so she asked her brother, [Ronald], for advice. [Kristen] liked to play hide and seek with her son [Ronald].
Conflict Parse: mother(Ronald, Kristen), sister(Ronald, Kristen)
Conflict Answer: mother(Ronald, Kristen)', 'sister(Ronald, Kristen)

In this sample, the author has already emphasized that Ronald is Kristen's son. Similarly, Kristen is also the mother of Patty in the given context. Therefore, Ronald is the brother of Patty. However, with ambiguous illustration in the story, Ronald becomes the brother of Kristen rather than Patty, causing a serious conflict in the relationship between distinct people.

5 LIMITATIONS AND DISCUSSIONS

This study proposes a novel framework coupling LLMs with ASP to identify misleading information. While the results are promising and show relevant improvements in certain issues, there are several limitations and concerns needed to be considered for future research and applications.

Performance and Efficiency. As the input data become more complicated and numerous, the computational cost for the parsing phase using LLMs may exponentially increase. Similarly, the results from the ASP solver and its explainable extension can be overloaded, leading to waste of resources, decrease in program quality. Besides, recursive rules can cause an infinite number of tree construction making the program less efficient and on the verge of collapse. Alternative strategies to manage computational resources should be made to prevent the failure of system.

Restricted Rule Set. One problem regarding the scability of this system is that foundation rules for the

ASP solver and also its explainable extension are defined depending on specific input data and expected output. Therefore, for different datasets with seperated targets, compatible rule sets have to be reconstructed. In general, this work still demands on the control of human and has not yet been automated.

Perspectives. Further research is needed to address the above limitations comprehensively varying from applying optimization techniques to handle the usage of computational resources to building generative rules system. Moreover, based on the proposal of this study, new development in solving logical reasoning tasks in natural language can be conversed. One possible work is to eliminate errors in verbal context with the advancements of LLMs and ASP.

6 CONCLUSION

This study introduces an innovative methodology for detecting misleading information by integrating Large Language Models (LLMs) with explainable Answer Set Programming (ASP). The synergy between the contextual understanding capabilities of LLMs and the reasoning and explanatory potential of explainable ASP has demonstrated the effectiveness in identifying misleading information that can cause confusion and significantly affect the accuracy of responses. It makes a substantial contribution to the advancement and refinement of reliable AI question-answering systems.

REFERENCES

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Bauer, J. J., Eiter, T., Ruiz, N. H., and Oetsch, J. (2023). Neuro-symbolic visual graph question answering with LLMs for language parsing. In *Proc. of TAASP 2023*.

Brown, T. B. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Cabalar, P., Fandinno, J., and Muñiz, B. (2020). A system for explainable answer set programming. In *Proc. of ICLP*, pages 124–136.

Calimeri, F., Ianni, G., and Ricca, F. (2014). The third open answer set programming competition. *Theory Pract. Log. Program.*, 14(1):117–135.

Eiter, T., Ianni, G., and Krennwallner, T. (2009). *Answer set programming: A primer*. Springer.

Erdem, E., Gelfond, M., and Leone, N. (2016). Applications of answer set programming. *AI Mag.*, 37(3):53–68.

Gebser, M., Kaminski, R., Kaufmann, B., and Schaub, T. (2019). Multi-shot ASP solving with clingo. *Theory Pract. Log. Program.*, 19(1):27–82.

Gelfond, M. and Lifschitz, V. (1988). The stable model semantics for logic programming. In *Proc. of ICLP/SLP*, pages 1070–1080. Cambridge, MA.

Ishay, A., Yang, Z., and Lee, J. (2023). Leveraging large language models to generate answer set programs. *arXiv preprint arXiv:2307.07699*.

Kheiri, K. and Karimi, H. (2023). SentimentGPT: Exploiting GPT for advanced sentiment analysis and its departure from current machine learning. *arXiv preprint arXiv:2307.10234*.

Liga, D. and Robaldo, L. (2023). Fine-tuning GPT-3 for legal rule classification. *Comput. Law Secur. Rev.*, 51:105864.

Marriott, K. and Stuckey, P. J. (1998). *Programming with constraints: an introduction*. MIT press.

McCarthy, J. (1959). Programs with common sense.

Nguyen, H.-T., Fungwacharakorn, W., and Satoh, K. (2023a). Enhancing logical reasoning in large language models to facilitate legal applications. *arXiv preprint arXiv:2311.13095*.

Nguyen, H.-T., Goebel, R., Toni, F., Stathis, K., and Satoh, K. (2023b). How well do SOTA legal reasoning models support abductive reasoning? *arXiv preprint arXiv:2304.06912*.

Phi, M., Nguyen, H., Bach, N. X., Tran, V. D., Nguyen, M. L., and Phuong, T. M. (2020). Answering legal questions by learning neural attentive text representation. In *Proc. of COLING*, pages 988–998. International Committee on Computational Linguistics.

Rajasekharan, A., Zeng, Y., Padalkar, P., and Gupta, G. (2023). Reliable natural language understanding with large language models and answer set programming. *arXiv preprint arXiv:2302.03780*.

Sinha, K., Sodhani, S., Dong, J., Pineau, J., and Hamilton, W. L. (2019). CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *Proc. of EMNLP-IJCNLP*, pages 4505–4514. Association for Computational Linguistics.

Trinh, G. V., Benhamou, B., Pastva, S., and Soliman, S. (2024). Scalable enumeration of trap spaces in Boolean networks via answer set programming. In *Proc. of AAAI*, pages 10714–10722. AAAI Press.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proc. of NeurIPS*, pages 5998–6008.

Yang, Z., Ishay, A., and Lee, J. (2023). Neurasp: Embracing neural networks into answer set programming. *arXiv preprint arXiv:2307.07700*.

Zin, M. M., Nguyen, H., Satoh, K., Sugawara, S., and Nishino, F. (2023). Information extraction from lengthy legal contracts: Leveraging query-based summarization and GPT-3.5. In *Proc. of JURIX*, pages 177–186. IOS Press.