A Study on Contradiction Detection Using a Neuro-Symbolic Approach

Alessia Donata Camarda*, Giovambattista Ianni

University of Calabria, Rende, 86037, Italy

Abstract

Research related to automated contradiction processing is a hot topic in several scientific communities. Many solutions offer limited explainability due to the exclusive usage of neural models or machine learning approaches. Also, although reproducibility can be achieved by controlling randomness in these models, their inherent complexity and lack of transparency often hinder adoption in domains that require high levels of determinism and some elaboration tolerance. This problem can be addressed with the introduction of neuro-symbol approaches, where part of the problem can be solved by exploiting logical formalisms. In this paper, we propose a neuro-symbolic pipeline whose purpose is to identify simple contradictions within sentences expressed in natural language. The contribution of the non-explainable side of the pipeline is confined to the conversion from natural language, and to the extraction of commonsense knowledge, while reasoning and knowledge derivation is delegated to a symbolic reasoner based on Answer Set Programming. We describe the proposed architecture and present a simplified implementation. We then report about early experiments.

Keywords

Knowledge representation and reasoning, Neuro-Symbolic AI, Digital Forensics, Logic Programming, Natural Language Processing

1. Introduction

Contradiction management [1, 2] is a longstanding hot topic, especially in the Natural Language Processing field [3, 4], as it is transversal to many other tasks such as speech recognition, text classification, natural-language understanding, and textual entailment classification. Contradiction management, herein intended as all the research issues arising when dealing with the detection, automated processing, symbolic representation and usage of contradictions, has been object of study in many other fields under a different terminology. In particular, dealing with contradictions is a fundamental yet challenging issue in the legal and forensics field [5], where adding decision support tools for analyzing possibly contradictory evidence and/or possibly conflicting testimonies is as strategic as it is challenging.

In the specific, the concept of contradiction and, more generally, the notion of anomaly plays a prominent role in the forensics field. When solving a case, evidence and testimonies are typically of not straightforward interpretation, and require proper processing: the search for new evidence that can confirm or contradict previous information and/or testimonies also plays an important role.

Contradictions have been addressed from various perspectives also in the field of Knowledge Representation and Reasoning. Some examples of this line of research can be identified in the inconsistency management through argumentation frameworks [6] or in the realm of logic programming and description logic [7, 8, 9, 10].

Contradiction management is often approached with neural models [11, 12, 13], where recognizable patterns within sentences are learned: these models are often influenced by the presence of specific trigger word in sentences (e.g."however", "but", etc.) and they are limited at directly detecting the semantic presence or absence of contradictions.

The usage of logical formalisms in this respect can, in principle, help in solving some known issues in contradiction management, such as adding forms of explainability, ensuring repeatable consistency

CILC 2025: 40th Italian Conference on Computational Logic, June 25–27, 2025, Alghero, Italy

*Corresponding author.

alessiadonata.camarda@unical.it (A. D. Camarda); giovambattista.ianni@unical.it (G. Ianni)

checks, supporting conflict resolution strategies and/or generating all the non-conflicting scenarios, and facilitating the integration of heterogeneous knowledge sources under an unified symbolic format.

However, the fruitful adoption of knowledge representation tools in the realm of contradiction management is prevented by several research gaps, among which are:

- *Conversion of natural language into logical statements.* As widely investigated in the natural processing field, humans speak or write ambiguously: they omit subjects and context, they imply multiple subjects and predicates in the same sentence, and so on. A transformation from natural language to logic statements should preserve meaning, resolve ambiguities, retain context whenever applicable, and support reasoning.
- *Formalizing contradictory knowledge.* Although inconsistency is a first-class citizen in logical representations, and can be dealt with in several ways, contradictions occurring in natural language exhibit several shades that do not have an obvious mapping to sharp inconsistencies modeled in formal logic. For instance, in natural language two propositions might be contradictory because of the conflicting semantics they convey and not because one is the direct negation of the other. However, modeling semantics requires to additionally model a non-negligible portion of background knowledge.

One of the alternative options that could be explored to manage contradictions is the use of neuro-symbolic architectures, i.e. the combinations of neural models and symbolic approaches. This kind of solution is becoming increasingly widespread as it allows the introduction of explainable and deterministic modules within combined evaluation pipelines. These latter would be otherwise totally probabilistic and difficult to explain. Such approaches are ideal in areas where explainability plays an important role, such as the legal and forensic one. In this paper, we propose to confine the role of Large Language Models (LLMs) just as factual knowledge extraction tools: the extracted information is then processed in a subsequent symbolic reasoning stage that deals with structured knowledge expressed in form of rules. As a symbolic declarative programming language we will focus on Answer Set Programming [14, 15], the prominent logic programming formalism widely adopted in the state of the art, especially in applications involving non-monotonic reasoning and knowledge representation.

Our contribution can be summarized in the following points:

- We propose a neuro-symbolic pipeline whose final output identifies the presence of contradictions within sentence pairs.
- We inspect the possibility of leveraging LLMs as extractors of ground terminological knowledge, possibly re-usable for modeling contradictory statements.
- We explore a method for using LLMs as translators from natural language to logical formalisms, based on providing a fixed predicate format to the language model at hand. This approach has been made possible by recent improvement in LLMs that allow to specify the predicate format in prompts, without fine tuning and training.
- As modeling all contradictions inherent in natural language is an ambitious goal, we identify a proper class of contradictory statements which, nonetheless covers a reasonable range of practical cases, under some assumption of "context atomicity".
- We report about first experiments that we performed in this respect, and discuss possible extensions and contradiction modelings.

The rest of the paper is structured as follows. Section 2 provides some background details and reviews related work. Sections 3, 4 and 5 discuss in detail our proposed neuro-symbolic pipeline and the results obtained experimenting this pipeline. Section 6 provides some insights for possible extended modeling that analyze more specifically some different types of contradictions. Finally, Section 7 outlines our conclusions and some future work.

2. Background and related work

Contradictions classification An analysis of categories of contradictions can be found [16]. De Marneffe et al. start their analysis with a simple question: "What is a contradiction?". The answer they gave themselves is the following: "contradiction occurs when two sentences are extremely unlikely to be true simultaneously". Although this answer is far from a strict logical definition of contradiction (i.e. sentences A and B are contradictory if there is no possible world in which A and B are both true), it comes very close to the way humans reason and speak. The taxonomy proposed by De Marneffe et al can be summarized as follows:

- **Antonymy-Based Contradictions (AC)**: these arise when key words in two statements are antonyms or have opposing meanings.
- **Negation-Based Contradictions (NEC)**: these occur when one statement explicitly negates another.
- **Numeric Contradictions (NUC)**: these arise when numerical values (such as quantities, dates, or statistics) do not match.
- **Factive Contradictions (FC)**: these contradictions arise from factive verbs (e.g., "know," "realize," "confirm"), which presuppose the logical truth of the statement that is being affirmed.
- **Structural Contradictions (SC)**: these contradictions arise due to differences in sentence structure, often leading to different interpretations.
- **Lexical Contradictions (LC)**: these occur when two sentences use different words that imply opposing facts.
- **World Knowledge Contradictions (WKC)**: these contradictions rely on conflicting general world knowledge.

A similar classification is presented by Wu et al. [17]. This taxonomy is more suited for AI-based pattern detection rather than formal logic. Specially, some of the proposed classes do not have clear logical structures and are harder to formalize. For example the "Scope-Based Contradictions", which occurs when the scope of an event or fact is narrowed down or broadened in a way that contradicts the first statement. In [18] a classification of contradictions theorized by Aristotle is revisited. However, it does not fully reflect the contradictions of natural language as we have described them before.

Large Language Models A Large Language Model is a machine learning model which can generate new content in response to a particular prompt, usually in a human-like language. LLMs have become popular due to the large number of tasks they can perform, e.g. text generation, translation, content summary, rewriting, and classification. LLMs are trained on vast amounts of text data from various sources, e.g. books, articles, websites, which are usually cleaned from noise, standardized, and tokenized into smaller units. According to their internal structure, LLMs can learn the importance of different words and capture complex relationships and dependencies within the text. They learn how to predict the next word or sequence of words in the given text based on the context provided by the preceding words. After the training phase, LLMs can be further trained (fine-tuned) on specific tasks or domains to improve performance and to adapt their parameters to particular requirements of the task at hand.

LLMs as semantic parsers and knowledge generator. Large Language Models have also been used as generators of structured semantic knowledge with the aim of automating the generation of facts or logical rules so as to facilitate the knowledge designer's work. Some approaches generate structured knowledge from texts [19, 20, 21, 22] while others receive images or videos as input [23]. The main task of these studies is to exploit LLMs as semantic parsers: therefore some of these approaches discuss the possibility of fine-tuning the model in question to improve the generation capabilities and reduce the possibility of errors. In particular, the work of [24] provides a pipeline in which input tests are also provided to LLMs in order to better handle errors and possibly improve the generation. LLMs have been used for the population of ontologies whose schema already exists [25]; and to derive entities, associations and properties, in order to populate a new schema starting from texts and tables [26, 27].

Legal and Forensic Ontologies As the forensic field is a driving domain when considering contradiction management, we explored the panorama of forensic ontologies in order to assess existing knowledge models and possibly integrate our work in this context. Several ontologies have been implemented to standardize and, to some extent, automate the representation of knowledge in forensics and law. Since these two fields are closely connected, these ontologies often share overlapping concepts. However, their scope and purpose are different. The Unified Cybersecurity Ontology (UCO) [28] was designed to support information integration and situational awareness in cybersecurity systems. UCO aggregates data from various cybersecurity tools for comprehensive analysis and supports automated threat intelligence sharing. UCO has been extended by the Cyber-investigation Analysis Standard Expression (CASE) ontology [29], which has become one of the most widely adopted standards in digital forensics. CASE focuses on the representation of investigative actions and artifacts, and it helps in standardizing the reporting of cyber investigations. Another important ontology is the Cyber Forensics Ontology for Cyber Criminal Investigation [30], which aims to classify cybercrimes and examine the collection of digital evidence. In the legal domain, several well-established ontologies exist. These include:

- PROV-O [31], which models provenance information and describes the origins and history of data and entities;
- LKIF (Legal Knowledge Interchange Format) Core Ontology [32], which represents basic legal concepts such as norms, legal acts, and roles;
- OPJK (Ontology of Professional Judicial Knowledge) [33], which aggregates knowledge relevant to legal professionals, particularly for legal education or frequent questions.

However, none of the aforementioned ontologies explicitly model complex legal concepts such as the content of testimony or depositions, despite their importance in both investigative and judicial processes.

Answer Set Programming Answer Set Programming (ASP) is a declarative language used in logic programming; even if its birth dates back to around 1993, nowadays, it is widely used in the field of artificial intelligence, planning, robotics and, currently, digital forensics [34, 35, 36]. We assume the reader is familiar with the basic of modelling with Answer Set Programming and we remaind to [37] for further information on the topic.

3. Contradiction modeling and detection framework

We will illustrate next our framework and the type of contradictions we deal with.

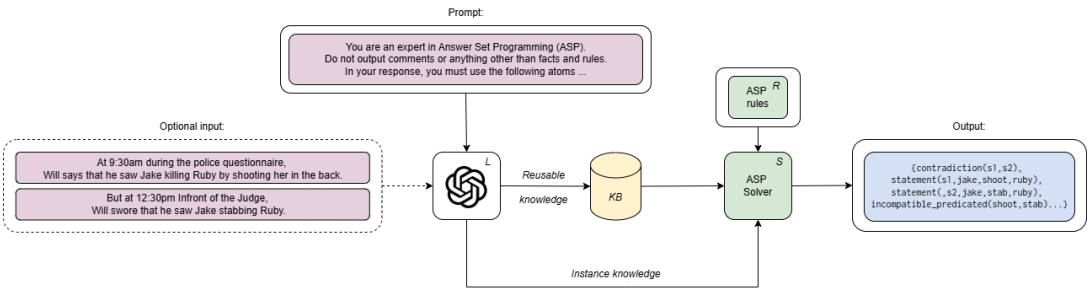

Figure 1: Pipeline used to conduct the experiments. Inside the input and output boxes we have examples of, respectively, sentences that we feed to the LLM and what we expect as output from the pipeline.

Our neuro-symbolic pipeline is depicted in Figure 1. The framework includes a Large Language Model L and an ASP solver S.

At runtime, L plays both the role of commonsense knowledge extractor and of parser converting knowledge in logical statements. L is given as input a pair of claims (c_1, c_2), whose core semantic content is translated from natural language into ASP facts (*instance knowledge*). Also L is prompted to produce logical facts that model possible incompatibilities present in the sentences, and useful for identifying contradictions (*reusable knowledge*).

Reusable knowledge produced by L can be used to populate a knowledge base KB. The content of KB, optionally containing also reusable knowledge collected in the past, is provided as input to the ASP solver S together with instance knowledge. S gets in input also a logic program R, which includes fixed knowledge and rules useful in identifying the contradictions possibly present in the input. The program rules are written by humans and not dynamically generated.
The output of S is an answer set which contains instances of the $contradiction$ predicate, indicating whether the input to the pipeline is contradictory.

3.1. Types of contradictions

We assume to deal with couples of claims (c_1, c_2) where c_1 and c_2 are in the form (id, s, p, o), where s represent the *subject* of the claim, p is the *predicate* and o is the *object*. id is a unique identifier associated to each claim. We also assume that claims refer to events happened at the same time, and that one specific main atomic event involved a single actor performing a single atomic action.

We are particularly interested in three main type of incompatibilities:

- Incompatible objects:
  - Given an object o, o is incompatible with respect to a predicate p. For example: given the predicate $stab$ and the object gun, these are incompatible, as one cannot stab someone with a gun. This type of inconsistency is modeled using the logical fact $incompatible\_object(p, o)$, which is expected to be produced by the LLM at hand according to its commonsense knowledge.
  - Given an object o_1 and an object o_2, these are incompatible simply because they are different and only one object for an atomic action can be identified as the object of the claim. For example, it is contradictory to state that an atomic action was performed by Alice with a gun and then to state that the same atomic action was performed using a $knife$. This type of incompatibility is handled using apposite logic rules.
- Incompatible predicates: predicates p_1 and p_2 could be conflicting when they are used referring to the same atomic action whenever they are not synonymous or equal. For example the predicate $stab$ and the predicate $shoot$ are incompatible, as we expect one exclusively shoots or stabs someone in the same atomic action. This is modeled on the one hand by asking the LLM at hand to produce assertions in the form $incompatible\_predicates(p_1, p_2)$, whenever applicable.
- Incompatible subjects: given a subject s_1 and a subject s_2 these are incompatible simply because they are different. For example: stating that an action was performed by $Alice$ and then stating that it was performed by Bob leads to a contradiction due to the fact that Alice and Bob are two different people and cannot have performed the same action at that moment.
- Treatment of referrals to same objects and/or predicates: the same object in the atomic action can be referred with more than one word, possibly synonymous. Also, couples of predicates can have acceptable semantic similarities. Think, e.g. at gun and $firearm$, and to the predicates hit and $beat$. This knowledge is provided by the LLM at hand by prompting it to use just one name for referrals to words denoting the same object and/or action.

4. Experimental setting

We performed some experiments over a curated version of the dataset LLU proposed by [5]. The dataset contains possibly conflicting claims of testimonies regarding simplified crime events. The training set of the original dataset contained 1019 instances, while the test set contained 813 tuples. Then, the dataset has been refined to remove duplicates and to be compliant with our assumption of action atomicity. In the specific, sentences describing sequences of actions were modified to describe a single action, and some misclassified tuples have been modified to correct them. At the end of the cleaning phase, the dataset contains 31 instances, among which 16 are contradictions pairs of sentences and the remaining ones are neutral sentences. We tested our pipeline with a choice of two LLM models and an ASP solver. The LLMs of choice were GPT o1 and GPT o3-mini which were queried via their API interface, while the ASP solver of choice was DLV [38]. The LLM was prompted with a text divided into two sections as shown in Figure 2 and 3. The prompt is constituted by a fixed system prompt, and a dynamic user prompt where the two inputs sentences $C1$ and $C2$ are fed as parameters.

```
 You are an expert in Answer Set Programming (ASP). Do not output comments or anything other than
facts and rules. In your response, you must use the following atoms:
"statement(ID, S, P, O)", where "ID" is the ID of the sentence, "S" is the main subject, "P" is the
main predicate, and "O" is the object of the sentence;
"weapon(ID, W)", where "ID" is the ID of the sentence, and "W" is the weapon involved in the
predicate P;
"incompatible_object(P, O)", where "O" is an object and "P" is a predicate of the same sentence that
cannot be done using the object O;
"incompatible_predicates(P1, P2)", where "P1" and "P2" are two different predicates which are not
synonyms of each other.
Example of sentences:
Sentence1: "Yesterday, Will said that he saw Mary killing Peter."
Sentence2: "But in front of the judge, Will swore that he saw Mary beating Peter."
Example of output: statement(s1, mary, kill, peter). statement(s2, mary, beat, peter).
incompatible_predicates(kill, beat).
```

Figure 2: The system prompt used for testing our pipeline.

```
 Return ASP facts that model the event involved in the following sentences and their possible
incompatibilities. If necessary, when you find two words that are similar or synonyms, keep only one
of the two words.
"Sentence1: {C1}"
"Sentence2: {C2}"
```

Figure 3: The user prompt used for testing our pipeline.

The prompts were designed by taking into account the following criteria:

- Specifying the format of the predicates is necessary to keep the knowledge generated by the model consistent, otherwise the LLM could provide different predicates for each answer. In this latter case, it would not be possible to reuse the knowledge obtained.
- Specifying that we do not want comments or anything else to be returned in the response is necessary to immediately obtain an executable result that can be parsed by an ASP solver. Otherwise, we would first have to clean the model output and then feed it as input to the solver.
- Including an example is of great importance given the complex request. This gives the model a clearer view of the request made previously and what output we expect.

The results obtained were then given as input to the ASP solver to compute the answer sets, together with the fixed ASP rules described next.

4.1. Answer Set Programming rules

Symbolic reasoning and contradiction entailment are delegated to the Answer Set Programming module. As we delve deeper into the context of our example (i.e. the forensic one), this module acquires great importance. In fact, it becomes the explainable, deterministic and reproducible part of the pipeline. The output obtained from the LLM becomes the input for the ASP solver along with some rules human-written that model the simplest cases of contradictions. Since we are considering the base cases, the rules do not need auxiliary predicates to those we have already discussed in the previous sections. Below we report about the logic program we adopted for our experiments. The rules aim to entail the atom "$contradiction(c)$" as new knowledge. The term c is a function symbol that allows us to take into account all the statements that are considered to be contradictory. In fact, we can find cases of incompatibility between pairs of sentences but also in single sentences, which is why we need to be able to consider both cases but to represent them in a single atom. In case an incompatibility is identified in a single sentence, the value of c will be $f(id)$; if the incompatibility occurs between two statements then the value of c will be in the form $f(id_1, id_2)$. The last four rules model our atomicity assumption, i.e. they deal with the possibility that the two input testimonies are different from each other although referring to the same event. This is a reasonable assumption in the forensic context, where one should take into account that a first testimony could differ greatly from a second, e.g. when the same person is accused of having done two different, mutually exclusive actions.

```
% Rules about incompatibles objects
contradiction(f(ID1)) :- statement(ID1, S, P, O), additional_info(ID1, weapon, W),
                         incompatible_objects(P, O).
contradiction(f(ID1)) :- statement(ID1, S, P, O), additional_info(ID1, weapon, W),
                         incompatible_objects(P, W).
contradiction(f(ID1, ID2)) :- statement(ID1, S, P, O1), statement(ID2, S, P, O2),
                              ID1 != ID2, O1 != O2.
contradiction(f(ID1, ID2)) :- statement(ID1, S, P, O), statement(ID2, S, P, O),
                              additional_info(ID1, weapon, W1), ID1 != ID2,
                              additional_info(ID2, weapon, W2), W1 != W2.

% Rules about incompatibles predicates
contradiction(f(ID1, ID2)) :- statement(ID1, S, P1, O), statement(ID2, S, P2, O),
                              incompatible_predicates(P1, P2), P1 != P2, ID1!=ID2.

% Rules about incompatible subjects
contradiction(f(ID1, ID2)) :- statement(ID1, S1, P, O), statement(ID2, S2, P, O),
                              ID1 != ID2, S1 != S2.

%Rules about changes from the first deposition
contradiction(f(ID1, ID2)) :- statement(ID1, S1, P1, O), statement(ID2, S2, P2, O),
                              ID1 != ID2, S1 != S2,
                              incompatible_predicates(P1, P2), P1 != P2.
contradiction(f(ID1, ID2)) :- statement(ID1, S, P1, O1), statement(ID2, S, P2, O2),
                              ID1 != ID2, O1 != O2,
                              incompatible_predicates(P1, P2), P1 != P2.
contradiction(f(ID1, ID2)) :- statement(ID1, S, P1, O), statement(ID2, S, P2, O),
                              additional_info(ID1, weapon, W1), W1 != W2,
                              additional_info(ID2, weapon, W2), ID1 != ID2,
                              incompatible_predicates(P1, P2), P1 != P2.
contradiction(f(ID1, ID2)) :- statement(ID1, S1, P, O1), statement(ID2, S2, P, O2),
                              ID1 != ID2, O1 != O2, S1 != S2.
```

Model	SE	TP	FP	TN	FN	ACC	PRE	REC	F1
GPT o3-mini	0.03	0.81	0.19	0.64	0.36	0.71	0.81	0.72	0.76
GPT o1	**0**	**0.87**	**0.13**	**0.80**	**0.20**	**0.84**	**0.87**	**0.82**	**0.84**

Table 1
The table reports statistics about the experiments conducted.

5. Results

Pairs of sentences were given as input to the two LLMs of choice. These could be contradictory or neutral (i.e. deemed as compatible). The results obtained were compared with the known ground truth, while we manually monitored the output of the LLMs and of the ASP solver, in order to assess the quality of the pipeline. In general, we noticed little differences between the output of GPT o1 and GPT o3-mini.

In particular, we noticed that:

- Both models struggle to represent negative expressions as triples (S, P, O). For example, the sentence “he omitted to have killed” is translated as “he killed”, so the final predicate is "kill" in both cases, thus missing to demonstrate the contradiction between the two sentences.
- GPT o3-mini tries to use all predicates present in the prompt, even if they are not needed. For example, if in two depositions the subjects are incompatible (in the first the subject is Alice while in the second the subject is Bob), the model still tries to represent this incompatibility using the predicate $incompatible\_predicates$, even though it does not model this case.
- Both models attempt to model background information that does not need to be modeled. For example, if a deposition contains the sentence "Mike confesses to killing Jessica but forensic scientists say it was Ross," the models will also attempt to model the information that Ross was the culprit.
- Both models struggle to represent information in triples (S, P, O) in some sentences. For example, if one statement states that the subject ran west and the next statement states that he ran east, then the models will use the geographic information as an attribute of the predicate, modeling them as "run_west" and "run_east" instead of attributing this information to the object.

Quantitative results are shown in Table 1. The table is structured as follow:

- The first column identifies the model used. The second column identifies the percentage of instances returned by the LLMs that present Syntactic Errors (SE).
- The third, fourth, fifth and sixth columns represent respectively: the rate of True Positives (instances classified as contradictory and which actually are), the rate of False Positives (instances classified as contradictory but which are not), the rate of True Negatives (instances classified as neutral and which actually are), and the rate of False Negatives (instances classified as neutral but which are not). Please note that by classified as contradictory we mean that, given the output of the LLM as input to the ASP solver, it detects the presence of a contradiction (therefore the atom $contradiction(c)$ for some c was entailed in the answer set).
- The remaining columns of the table represent standard evaluation metrics used to assess the models’ performance. ACC (Accuracy) indicates the rate of correct predictions over the total number of instances. PRE (Precision) measures the rate of instances correctly identified as positive out of all instances predicted as positive ones. REC (Recall) represents the rate of positive instances correctly identified by the model. F1 (F1-Score) is the harmonic mean of precision and recall.

From the obtained results, we can notice that GPT o1 was the better performer of the two models. It did not return any instances containing syntactic errors and also had higher evaluation metrics.

This suggests that GPT o1 is more reliable and effective in terms of output format and in extracting commonsense knowledge. The results are not entirely surprising: GPT o1 is a significantly larger and more powerful model compared to GPT o3-mini.

The code and the obtained results are available at the following repository: https://github.com/DeMaCS-UNICAL/CILC-contrdetect.

6. Possible extensions and alternative modelings

Our contradiction model leaves room for refinement by lifting some of our simplifying assumptions. For instance, looking more closely at the modeling of *incompatible predicates* one can see that they involve two main settings: couples of predicates can have either a completely different and unrelated meaning, or be deemed to have an opposite meaning. This last case models typical contradictions based on antonymies, as mentioned in Sec. 2. If we decide to explicitly model antonyms-based contradictions in ASP, it is necessary to introduce some terminological meta-predicate which represents couple of antonyms: e.g., we could use the atom $antonyms(x, y)$, where x and y are antonyms. Since this is a-priori information that we should have even before to run the ASP program, instances of such atom could be taken from some knowledge base. Such knowledge could be extracted from LLMs, or from fixed linguistic ontologies, like WordNet [39] or ConceptNet [40], a semantic network which includes common-sense knowledge. Leveraging such knowledge bases could further refine the neurosemantic approach, thus reducing the impact that the LLM has on the entire pipeline and shifting it towards the reasoning phase.

```
ab_contradiction(X, Y) :- sentence(X, S, P, O1), sentence(Y, S, P, O2),
                          O1 != O2, antonyms(O1, 02).
ab_contradiction(X, Y) :- sentence(X, S, P1, O), sentence(Y, S, P2, O),
                          P1 != P2, antonyms(P1, P2).
ab_contradiction(X, Y) :- sentence(X, S1, P, O), sentence(Y, S2, P, O),
                          S1 != S2, antonyms(S1, S2).
```

Contradictions based on antonyms can be considered a sub-case of a wider class of lexical contradictions, which occur when two words have incompatible meanings. One might consider modeling such contradictions, so as to cover a larger number of situations. Consider the following example:

- Sentence 1: "The cat is a kitten."
- Sentence 2: "The cat is an adult."

In this case, *kitten* and *adult* are not direct antonyms, but they refer to different mutually exclusive stages of life. To know when two concepts have an incompatible lexical meaning, we can model an ASP predicate $incompatible\_lexical\_meaning$ which will store this information. Like for the predicate $antonyms$, even $incompatible\_lexical\_meaning$ make the presence of a knowledge base necessary.

```
lc_contradiction(X, Y) :- sentence(X, S, P1, O), sentence(Y, S, P2, O),
                          P1 != P2, opposite_lexical_meaning(P1, P2).
lc_contradiction(X, Y) :- sentence(X, S, P, O1), sentence(Y, S, P, O2),
                          O1 != O2, opposite_lexical_meaning(O1, O2).
lc_contradiction(X, Y) :- sentence(X, S1, P, O), sentence(Y, S2, P, O),
                          S1 != S2, opposite_lexical_meaning(S1, S2).
```

Nonetheless, a lexical contradiction does not only occur when two subjects, two objects, or two relations respectively have opposite meanings and the rest of the sentence is unchanged between the two. Consider the following example:

- Sentence 1: "The government banned this product."
- Sentence 2: "The product is available for purchase in stores."

In this case, the object of the first sentence is the subject of the second and it is the relations of both sentences that have opposite meanings. So, we can consider cases like the following:

```
lc_contradiction(X, Y) :- sentence(X, S1, P1, O),
                          sentence(Y, O, P2, O2), P1 != P2,
                          opposite_lexical_meaning(P1, P2).
```

However, these cases are more complex and therefore fall outside the scope of this work.

7. Conclusions

In this paper we provided a study on how to model basic cases of contradiction by exploiting Large Language Models as knowledge extractors/translators while delegating symbolic reasoning aspects to an ASP solver. The obtained results can be considered promising enough to continue studying and deepening this approach, especially for more complex cases that involve sequences of events or a greater number of subjects involved.

Our pipeline could be modified in order to build the knowledge base without having to use the instance data available. In fact, the LLM could be prompted to obtain its commonsense knowledge of specific subdomains in the form of the $incompatible\_object$ and $incompatible\_predicates$ predicates. For example, one could prompt the model with the following request: *"Return all pairs of actions whose meaning is opposite using the predicate* `incompatible_predicates` *"*.

Furthermore, it could be interesting to try other translation methods, e.g. producing terminological assertions expressed in OWL or RDFS, or using entity and relationship extraction techniques, thus allowing to further reduce the role of LLMs. Moreover, in the future, it could be useful to explore the possibility of a formalization of the different types of contradictions present in the wild, in order to use such modelings for new ASP programs.

Acknowledgments

This work was partially supported by project SERICS (PE00000014) and FAIR (PE0000013) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

Declaration on Generative AI

Besides the usage of generative AI as described in the scientific contribution of the paper, that is, for the generation of logical formalisms to be used as input for the symbolic module, the authors used GPT-4 to do grammar and spelling check and rewriting of specific sentences. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] C. Condoravdi, D. Crouch, V. de Paiva, R. Stolle, D. G. Bobrow, Entailment, intensionality and text understanding, in: Proceedings of the HLT-NAACL 2003 Workshop on Text Meaning, 2003, pp. 38–45.

[2] S. M. Harabagiu, A. Hickl, V. F. Lacatusu, Negation, Contrast and Contradiction in Text Processing, in: AAAI, AAAI Press, 2006, pp. 755–762.

[3] I. Dagan, D. Roth, M. Sammons, F. M. Zanzotto, Recognizing Textual Entailment: Models and Applications, Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers, 2013.
[4] S. R. Bowman, G. Angeli, C. Potts, C. D. Manning, A large annotated corpus for learning natural language inference, in: EMNLP, The Association for Computational Linguistics, 2015, pp. 632–642.
[5] S. Surana, S. Dembla, P. Bihani, Identifying Contradictions in the Legal Proceedings Using Natural Language Models, SN Comput. Sci. 3 (2022) 187.
[6] P. M. Dung, On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games, Artif. Intell. 77 (1995) 321–358.
[7] S. Schlobach, R. Cornet, Non-Standard Reasoning Services for the Debugging of Description Logic Terminologies, in: IJCAI, Morgan Kaufmann, 2003, pp. 355–362.
[8] G. Qi, W. Liu, D. A. Bell, A revision-based approach to handling inconsistency in description logics, Artif. Intell. Rev. 26 (2006) 115–128.
[9] C. Hewitt, Inconsistency Robustness in Logic Programs, 2015. arXiv:0904.3036.
[10] H. A. Blair, V. S. Subrahmanian, Paraconsistent Logic Programming, Theor. Comput. Sci. 68 (1989) 135–154.
[11] M. Asif, T. A. Khan, W. Song, Evaluating large language models for optimized intent translation and contradiction detection using KNN in IBN, IEEE Access 13 (2025) 20316–20327.
[12] V. Dragos, Detection of contradictions by relation matching and uncertainty assessment, in: KES, volume 112 of *Procedia Computer Science*, Elsevier, 2017, pp. 71–80.
[13] M. Pielka, F. Rode, L. Pucknat, T. Deußer, R. Sifa, A linguistic investigation of machine learning based contradiction detection models: An empirical analysis and future perspectives, in: ICMLA, IEEE, 2022, pp. 1649–1653.
[14] T. Eiter, G. Ianni, T. Krennwallner, Answer Set Programming: A Primer, in: Reasoning Web, volume 5689 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 40–110.
[15] W. Faber, An Introduction to Answer Set Programming and Some of Its Extensions, in: RW, volume 12258 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 149–185.
[16] M. de Marneffe, A. N. Rafferty, C. D. Manning, Finding Contradictions in text, in: ACL, The Association for Computer Linguistics, 2008, pp. 1039–1047.
[17] X. Wu, X. Niu, R. Rahman, Topological Analysis of Contradictions in text, in: SIGIR, ACM, 2022, pp. 2478–2483.
[18] A. E. Gärtner, D. Göhlich, Automated requirement contradiction detection through formal logic and llms, Autom. Softw. Eng. 31 (2024) 49.
[19] A. Rajasekharan, Y. Zeng, P. Padalkar, G. Gupta, Reliable Natural Language Understanding with Large Language Models and Answer Set Programming, in: ICLP, volume 385 of *EPTCS*, 2023, pp. 274–287.
[20] M. A. B. Santana, I. Kareem, F. Ricca, Towards Automatic Composition of ASP Programs from Natural Language Specifications, in: IJCAI, ijcai.org, 2024, pp. 6198–6206.
[21] S. Caruso, C. Dodaro, M. Maratea, M. Mochi, F. Riccio, CNL2ASP: Converting Controlled Natural Language Sentences into ASP, Theory Pract. Log. Program. 24 (2024) 196–226.
[22] E. Coppolillo, F. Calimeri, G. Manco, S. Perri, F. Ricca, LLASP: Fine-tuning Large Language Models for Answer Set Programming, in: KR, 2024.
[23] J. Suchan, M. Bhatt, P. A. Walega, C. Schultz, Visual Explanation by High-Level Abduction: On Answer-Set Programming Driven Reasoning About Moving Objects, in: AAAI, AAAI Press, 2018, pp. 1965–1972.
[24] A. Kalyanpur, K. Saravanakumar, V. Barres, J. Chu-Carroll, D. Melville, D. A. Ferrucci, LLM-ARC: Enhancing LLMs with an Automated Reasoning Critic, CoRR abs/2406.17663 (2024).
[25] G. Ciatto, A. Agiollo, M. Magnini, A. Omicini, Large language models as oracles for instantiating ontologies with domain-specific knowledge, Knowl. Based Syst. 310 (2025) 112940.
[26] A. Oarga, M. Hart, A. M. Bran, M. Lederbauer, P. Schwaller, Scientific Knowledge Graph and Ontology Generation using Open Large Language Models, in: Neurips 2024 Workshop Foundation Models for Science: Progress, Opportunities, and Challenges, 2024.

[27] S. A. G´omez, P. R. Fillottrani, Leveraging Large Language Models for Ontology-Based Data Access: A Preliminary Analysis, 2024.
[28] Z. Syed, A. Padia, T. Finin, M. L. Mathews, A. Joshi, UCO: A Unified Cybersecurity Ontology, in: AAAI Workshop: Artificial Intelligence for Cyber Security, volume WS-16-03 of *AAAI Technical Report*, AAAI Press, 2016.
[29] E. Casey, S. Barnum, R. Griffith, J. Snyder, H. M. A. van Beek, A. Nelson, Advancing coordinated cyber-investigations and tool interoperability using a community developed specification language, Digit. Investig. 22 (2017) 14–45.
[30] H. Park, S. Cho, H. Kwon, Cyber Forensics Ontology for Cyber Criminal Investigation, in: e-Forensics, volume 8 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Springer, 2009, pp. 160–165.
[31] T. Prudhomme, G. De Colle, A. Liebers, A. Sculley, P. Xie, S. Cohen, J. Beverley, A semantic approach to mapping the Provenance Ontology to Basic Formal Ontology, Scientific Data 12 (2025).
[32] R. Hoekstra, J. Breuker, M. D. Bello, A. Boer, The LKIF Core Ontology of Basic Legal Concepts, in: LOAIT, volume 321 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2007, pp. 43–63.
[33] V. R. Benjamins, J. Contreras, P. Casanovas, M. Ayuso, M. Bécue, L. Lemus, C. Urios, Ontologies of Professional Legal Knowledge as the Basis for Intelligent IT Support for Judges, Artif. Intell. Law 12 (2004) 359–378.
[34] S. Costantini, F. A. Lisi, R. Olivieri, DigForASP: A European Cooperation Network for Logic-based AI in Digital Forensics, in: CILC, volume 2396 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2019, pp. 138–146.
[35] E. Erdem, M. Gelfond, N. Leone, Applications of Answer Set Programming, AI Mag. 37 (2016) 53–68.
[36] G. Ianni, F. Pacenza, J. Zangari, Incremental maintenance of overgrounded logic programs with tailored simplifications, Theory Pract. Log. Program. 20 (2020) 719–734.
[37] V. Lifschitz, Answer Set Programming, Springer, 2019.
[38] M. Alviano, F. Calimeri, C. Dodaro, D. Fuscà, N. Leone, S. Perri, F. Ricca, P. Veltri, J. Zangari, The ASP system DLV2, in: LPNMR, volume 10377 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 215–221.
[39] G. A. Miller, WORDNET: a lexical database for english, in: HLT, Morgan Kaufmann, 1992.
[40] R. Speer, C. Havasi, Conceptnet 5, Adv. Math. Commun. 1 (2012).