Deductive Verification of LLM generated SPARQL queries

Alexandre Rademaker[1,2], **Guilherme Lima**[1], **Sandro R. Fiorini**[1], **Viviane T. da Silva**[1]
IBM Research[1] and School of Applied Mathematics FGV[2]
Rio de Janeiro, Brazil
{alexrad,vivianet}@br.ibm.com, {guilherme.lima,srfiorini}@ibm.com

Abstract

Considering the increasing applications of Large Language Models (LLMs) to many natural language tasks, this paper presents preliminary findings on developing a verification component for detecting hallucinations of an LLM that produces SPARQL queries from natural language questions. We suggest a logic-based deductive verification of the generated SPARQL query by checking if the original NL question's deep semantic representation entails the SPARQL's semantic representation.

Keywords: SPARQL, LLM, hallucination, HOL, higher-order logic, Z3, theorem prover

1. Introduction

This paper reports the preliminary results of developing a verification component of a chat-like interface for chemists interested in retrieving information about chemical compounds from a knowledge graph (KG) like Wikidata or PubChem.[1]

In Section 2, we briefly presented our pipeline and how questions formulated in English as the Example (1-a) are translated to the SPARQL queries as the one presented in Listing 1 using an LLM like GPT-4 (OpenAI, 2023) or LLAMA-2-70b [2]. However, our focus in this paper is not on the SPARQL generation nor the correctness of the answer provided by the Knowledge Graph for the SPARQL query. Instead, we focus on validating the SPARQL query obtained from the LLM. In other words, if the SPARQL 'makes sense' given the original question formulated in English. Preventing hallucinations has been a hot topic in the literature recently (Wang et al., 2023; Cao, 2023; Ling et al., 2023; Dhuliawala et al., 2023). In our application to assist chemists asking for properties about chemical compounds in a chat, a user relying only on the LLM's final answer can be misguided if he can't read the intermediary SPARQL query produced to retrieve the facts from the Knowledge Graph.

(1) a. What is the mass of benzene?
b. Give me the benzene's toxicity.
c. What chemical compounds have less than 0.07 g/kg of solubility?
d. What is the electric dipole moment of the allyl alcohol?
e. What is the mass of the compound with InChIKey UHOVQNZJYSORNB-UHFFFAOYSA-N?

In Example (1), we present some variants of questions about chemical compounds and their properties that a chemist can submit to our chat interface. We are restricting our focus to factoid questions, answerable by simple SPARQL queries involving only simple triple patterns; there are many challenges to dealing with such questions. The first and most obvious is that the syntactic structure can vary greatly. The second challenge is that properties are hardly mentioned by their labels in KGs. For instance, in our context, toxicity should be interpreted as the substance's median lethal dose (LD50),[3] but we could not be sure in a more general context. Third, the property values are usually measured in complex units. Solubility is measured in 'grams per kilogram,' and LD50 is expressed as the mass of substance administered per unit mass of the test subject, typically as milligrams of a substance per kilogram of body mass. Moreover, Lethal dosage often varies depending on the method of administration; many substances are less toxic when administered orally than when intravenously administered. To sum up, the toxicity of a compound is usually expressed as a complex unit like 'LD50 Rat oral 3530 mg/kg'. Fourth, chemicals can be identified in various ways. For example, 'allyl alcohol' has 91 synonyms ranging from IUPAC names to identifiers in different standards proposed by the scientific communities.[4]

Figure 1 presents two logical formulas expressed in higher-order logic, particularly in ULKB Logic (Lima et al., 2023). ULKB is an open-source framework written in Python for logical reasoning over knowledge graphs. The first formula, Formula 1, is the logical semantics of Example (1-a)

[1] We will restrict our examples to Wikidata KG, but the techniques can be used with any KG.

[2] https://ai.meta.com/llama/

[3] The reader doesn't have to understand the chemical terms mentioned in this paragraph; we are only exemplifying the particularities on processing English questions on a technical domain.

[4] https://pubchem.ncbi.nlm.nih.gov/compound/Allyl-alcohol

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?v WHERE {
    wd:Q2270 wdt:P2067 ?v .
}
```

Listing 1: SPARQL query

obtained with MRS Logic (Rademaker et al., 2023), a library to translate English sentences into logical formulas built on top of ULKB and 'deep' linguistic specialized tools. The second formula, Formula 2, is the translation of the SPARQL to the ULKB logic, a functionality also presented in the ULKB Library. Provided a proper map between the predicates «wdt:P2067» [5] and _mass_n_off and also between «wdt:Q2270» and _benzene_n_1, we can conclude that Formula 1 entails Formula 2, certifying that the query is indeed related to the English questions. Section 3 presents the tools we rely on, and Section 4 elaborates on our process for validating the SPARQL query.

To summarize, our contribution is a deductive approach to prevent LLM hallucinations in translating English questions to SPARQL queries. The validation process reported here is a component of a chat-based assistant for chemists interested in obtaining information about chemical compounds from a large and complex KG without having to construct a SPARQL query manually. In other words, we are focusing not on ordinary questions presented in datasets like (Trivedi et al., 2017) but on questions made by chemists about chemical compounds, a deep technical domain with plenty of technical terms. On the other hand, our method is not restricted to any particular Knowledge Graph Question Answering (KGQA) approach; it can be adapted for different domains and systems such as (Zhou et al., 2021). Before describing the SPARQL validation method based on the semantic parsing of the NL utterances, we present in Section 2 the overview of our text to SPARQL conversion pipeline based on LLM. Our pipeline is one of the components of ChemChat, a conversational expert assistant in material science (Erdmann et al., 2024).

2. The architecture of our system

Using prompt engineering, we used 'few-short learning' (Brown et al., 2020). We implemented an LLM pipeline to translate English questions to SPARQL queries. Figure 2 presented the pipeline and the workflow to process the Example (1-a). To create the prompts, with the help of some chemists, we collected examples of English questions and manually annotated the technical chemical terms on them and their related SPARQL queries.

Let us first describe the step-by-step process of constructing a SPARQL from an English sentence. The pipeline (blue boxes) starts by sending the input question through an LLM with examples to suggest our goal of extracting a table with the relevant terms (usually adjective and noun phrases) and their classification as either 'property' or 'entity.' We construct 'Prompt 1' from the set of examples we have collected. Next, we parse the table received from the LLM to disambiguate the terms (that is, grounding them to Wikidata identifiers) using a full-text search on a database populated with relevant Wikidata items and properties with their labels. Utilizing the fact we are building a specialized interface for chemists, this database (an Elastic Search [6] index) is constructed from Wikidata's chemical taxonomy using items like 'type of chemical entity' (Q113145171) and the 'WikiProject Chemistry' (Q8487234) as seeds and exploring their descendants and related concepts and properties. We use the query results to construct a disambiguation table mapping each term obtained from the completion of Prompt 1 to its best-matching Wikidata identifier. Once we have the Wikidata identifiers, we can produce the second prompt (Prompt 2), now using examples of pairs of English questions and SPARQL queries together with their identifiers. We submit Prompt 2 to a second LLM to generate the final SPARQL query. Sometimes, the LLM fails to produce a valid SPARQL query; to deal with that, we repeat the process a few times and use the most frequent answer as the expected solution.

As stated in the introduction, this article does not aim to describe the LLM pipeline completely nor discuss its performance, precision, or recall. These topics will be the subject of another article. Our focus here is on one specific component: the method to validate the SPARQL obtained for each question in English, presented in the salmon boxes of Figure 2. First, the same English question submitted by the chemist is parsed with a computational grammar for English, and a semantic representation of the sentence is produced. This semantic representation is translated to a sentence (Sentence A) in higher-order logic using the MRS Logic library.

[5] We are adopting the notation «...» as a simplified way to reference the fully qualified URI of an item from the Wikidata data schema.

[6] https://www.elastic.co

$$\exists\, x_{13},\ \_benzene\_n\_1\ x_{13}\ \wedge\ (\exists\, x_8,\ \_mass\_n\_of\ x_8\ x_{13}\\ \wedge\ (\exists\, x_3,\ thing\ x_3\ \wedge\ (\exists\, e_2,\ \_be\_v\_id\ e_2\ x_3\ x_8))) \tag{1}$$

$$\exists\, value,\ \text{«wdt:P2067»}\ \text{«wdt:Q2270»}\ value \tag{2}$$

Figure 1: The formal semantics of the sentence in Example (1-a) and the translation to a logical formula of the SPARQL query from Listing 1. «wdt:P2067» is a binary predicate (associated with an RDF edge), and «wdt:Q2270» is a constant (associated with an RDF node). In ULKB, the «...» indicates that the function or predicate is associated with a URI. Applying a function or predicate to its arguments is usually written as $f(x, y)$ in many formal languages. Still, in ULKB, we write $f\ x\ y$, omitting the parenthesis and commas.

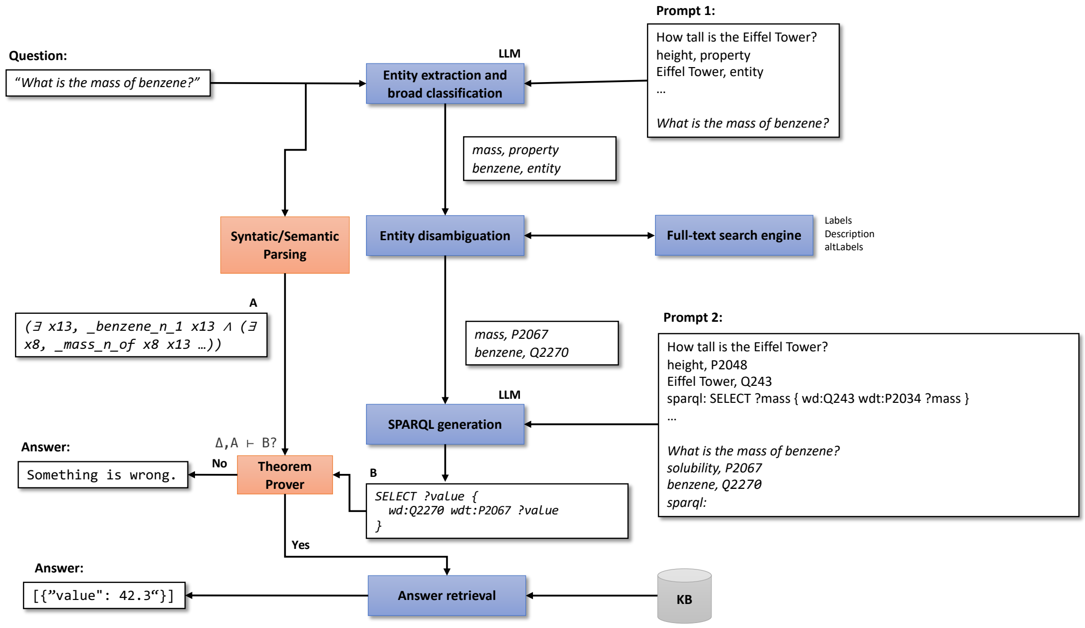

Figure 2: In our pipeline, the natural language question goes through one initial LLM prompted to extract relevant question terms. These terms are disambiguated using a full-text search over a subset of relevant Wikidata items. The search result helps a second LLM to generate SPARQL queries with the correct Wikidata identifiers.

The SPARQL query obtained from the LLM is also translated to Sentence B in higher-order logic using the ULKB library SPARQL to HOL translation. Using the ULKB wrapper to theorem provers, we check if sentence A (together with some additional axioms in the set of formulas Δ) entails sentence B. The following section describes the natural language processing tools we used for parsing and constructing logical formulas.

3. Background

MRS Logic (Rademaker et al., 2023) is a Python library to convert NL utterances into higher-order logic formulas. It is built on top of many other components that we describe below.

The main component of MRS Logic (Rademaker et al., 2023) is the English Resource Grammar (ERG) (Flickinger, 2000; Flickinger et al., 2000; Copestake and Flickinger, 2000). The English Resource Grammar is a broad-coverage, linguistically precise, general-purpose computational grammar. It is implemented in the theoretical framework of Head-driven Phrase Structure Grammar (Pollard and Sag, 1994) where both morphosyntactic and semantic properties of English are expressed in a declarative format. Combined with specialized processing tools, it can map running English text to highly normalized representations of meaning called Minimal Recursion Semantics (MRS) (Copestake et al., 2005). ERG is developed as part of the international Deep Linguistic Processing with HPSG Initiative (DELPH-IN). It can be processed by several parsing and realization systems, including ACE (Crysmann and Packard, 2012).[7]. MRS Logic uses PyDelphin (Goodman, 2019) library to communicate with ACE.

MRS structures directly interface with syntax and can be underspecified in various aspects, such as word senses and quantifier scopes. This underspecification enables a single MRS to encompass multiple interpretations. Figure 3 shows one among the nine possible MRSs for Example (1-a). It consists of a multiset of relations called elementary predications (EPs). An EP usually corresponds to a single lexeme but can represent grammatical features (e.g., `thing` and `udef_q`, called abstract predicates). Each EP has a label or handle, a predicate symbol, which, in the case of lexical predicates, encodes information about lemma, part-of-speech, and coarse-grained sense distinctions, and a list of numbered arguments: `ARG0`, `ARG1`, etc. The value of an argument can be either a scopal variable (a hole representing the places where alternative labels could fill) or a non-scopal variable (events, states, or entities). The `ARG0` argument has the EP's distinguished variable. This variable denotes an event, state, or referential or abstract entity (e_i or x_i, respectively). Each non-quantifier EP has its unique distinguished variable. Finally, an MRS has a set of handle constraints describing how the EPs' scopal arguments can be nested with EP labels. A constraint $h_i =_q h_j$ denotes equality modulo quantifier insertion. In addition to the indirect linking through handle constraints, EPs are directly linked by sharing the same variable as argument values, capturing the predicate-argument structure of the sentence. Finally, MRS also records properties on variables indicating morpho-syntactic marks of person, number, tense, aspect, etc. The topmost relation in Figure 3 is _be_v_id, which has the non-empty arguments x_5 and x_9. The x_5 is the distinguished variable of the relation `thing`. A handle constraint equates the sentential variable h_2 with h_1, the top handle. The rest of the EPs can be explained similarly. Note that h_7 does not appear in the handle constraints, suggesting that we have more than one possible way to equate this hole with the available labels.

For solving the underspecification of the scopes of quantifiers in an MRS, MRS Logic employs the Utool scope resolution Java Library (Koller and Thater, 2005, 2006, 2010). From a single MRS, Utool can produce many possible scope trees, fully scoped resolved trees, reflecting the different possible order of quantifiers in the final logical formula. For instance, the MRS of Figure 3 has an alternative reading for the order of quantifiers in Formula 1, e.g., $\exists\, x_8, \ldots \exists\, x_{13}, \ldots$, but in this case, the two are semantically equivalent.

The scope trees are not yet a concrete logical expression in any logical language. The literature has many proposals for representing NL utterance semantics. One of the most fundamental issues about which logic to use is whether one assumes any structure on the individuals. Other issues are the complexity, decidability, and tools for reasoning in a particular logic. Type theories are widely used in formal theories of the semantics of natural languages (Chatzikyriakidis and Luo, 2020; Ranta, 1994; Winter, 2016). A subset of that, simple type theory, also called higher-order logic (HOL), is a natural extension of first-order logic, which is elegant, highly expressive, and practical (Farmer, 2008).

The ULKB Logic (Lima et al., 2023) implements HOL in Python for logical reasoning over knowledge graphs. The formulas presented in Figure 1 are HOL formulas encoded in ULKB Logic. ULKB provides an interactive theorem prover-like environment that can interact with external provers such as E prover (Schulz et al., 2019) and Z3 SMT solver (de Moura and Björner, 2008). In (Lima et al., 2023), the authors present the logical foundations and implementation of ULKB Logic and its interfaces for fetching statements from knowledge graphs and calling external provers. These interfaces are vital for achieving ULKB Logic's primary goal, which is twofold: (i) provide a common language and interactive theorem prover-like environment for representing commonsense and linguistic knowledge, and (ii) facilitate the use of state-of-the-art computational logic tools to reason over the knowledge available in knowledge graphs. For (ii), ULKB uses SPARQL (W3C SPARQL Working Group, 2013), the standard query language of the Semantic Web, and allows users to use logic formulas as queries, parse SPARQL queries into logic formulas and submit SPARQL queries to KG endpoints.

Finally, consider the possible senses for the word 'mass.' ERG only distinguishes senses that are morphosyntactically marked. Since further sense distinctions could never be disambiguated based on grammatical structure alone, the ERG predicate symbol `_mass_n_of` intended to be an underspecified representation of all the specific word senses. For instance, Wordnet 3.1 (Miller, 1995) contains eleven possible nominal senses for this word. We use UKB (Agirre and Soroa, 2009) for Word Sense Disambiguation (WSD), the ERG predicates. UKB performs graph-based disambiguation using any pre-existing knowledge base, provided the structure of the graph (nodes and edges) and the dictionary of words or multi-word expressions associated with each node.

To summarize, MRS Logic takes an NL utterance and calls ACE to obtain all possible MRSs. Given an MRS, it is transformed into a scope tree using Utool and passed to UKB to disambiguate the ERG predicates, linking them to nodes in a reference

[7] `http://sweaglesw.org/linguistics/ace/`

$$\begin{array}{l}
\langle\, h_1, e_3\{\text{SF } \textit{ques}, \text{TENSE } \textit{pres}, \text{MOOD } \textit{indicative}, \text{PROG -}, \text{PERF -}\}, \\
\quad \left| \begin{array}{l}
h_4\text{:thing}\langle 0\!:\!4\rangle(\text{ARG0 } x_5\{\text{PERS } \textit{3}, \text{NUM } \textit{sg}\}), \\
h_6\text{:which\_q}\langle 0\!:\!4\rangle(\text{ARG0 } x_5,\ \text{RSTR } h_8,\ \text{BODY } h_7), \\
h_2\text{:\_be\_v\_id}\langle 5\!:\!7\rangle(\text{ARG0 } e_3,\ \text{ARG1 } x_5,\ \text{ARG2 } x_9\{\text{PERS } \textit{3}, \text{NUM } \textit{sg}\}), \\
h_{10}\text{:\_the\_q}\langle 8\!:\!11\rangle(\text{ARG0 } x_9,\ \text{RSTR } h_{12},\ \text{BODY } h_{11}), \\
h_{13}\text{:\_mass\_n\_of}\langle 12\!:\!16\rangle(\text{ARG0 } x_9,\ \text{ARG1 } x_{14}\{\text{PERS } \textit{3}, \text{NUM } \textit{sg}\}), \\
h_{15}\text{:udef\_q}\langle 20\!:\!28\rangle(\text{ARG0 } x_{14},\ \text{RSTR } h_{17},\ \text{BODY } h_{16}), \\
h_{18}\text{:\_benzene\_n\_1}\langle 20\!:\!27\rangle(\text{ARG0 } x_{14})
\end{array} \right| \\
\quad \{\, h_1 =_q h_2,\ h_8 =_q h_4,\ h_{12} =_q h_{13},\ h_{17} =_q h_{18} \,\}\,\rangle
\end{array}$$

Figure 3: The first MRS return by ERG for the Example (1-a).

KG.[8] Finally, the MRS is translated into ULKB formulas. MRS Logic integrates all the technologies described above. At the high level, the translation starts from the topmost node of the scope tree, the handle in the higher position, usually a quantifier. The translation is fully explained in (Rademaker et al., 2023).

4. Validating an SPARQL

Our main problem can be defined as the logical entailment test in Equation 3.

$$\Delta, T(\alpha) \models G(\alpha) \tag{3}$$

where α is an English question, $T(\alpha)$ is one of the possible higher-order logic formulas obtained from the English question α by the MRS Logic (Section 3), e.g., Formula 1, and $G(\alpha)$ is a first-order logic formula obtained from the translation of the SPARQL query, in our case, produced as the translation of the same English question to SPARQL by an LLM, e.g., the Formula 2 (Section 1). Finally, Δ is a set of axioms to support the entailment. This logic theory connects the symbols from the ERG grammar presented in the MRS to those obtained from the SPARQL query, the KG identifiers.

Consider again the natural language (English) question from Example (1-a). From the MRS in Figure 3, the UKB disambiguation step, using Wordnet 3.1 as KG disambiguates, produces the mapping of e_2 to the sense "have the quality of being; copula, used with an adjective or a predicate noun" (Synset `02610777-v`), variable x_8 to "the property of a body that causes it to have a weight in a gravitational field" (Synset `05031420-n`. Synset `05031420-n` is associated with the Wikidata item `Q11423` by the 'WordNet 3.1 Synset ID' (P8814) Wikidata property. This item has 'Wikidata property' linking it to the property `P2067`). The variable x_{13} is disambiguated to "a colorless liquid hydrocarbon" (Synset id `14798860-n`, associated to the Wikidata item `Q2270` by the same P8814 Wikidata property. In other words, from the disambiguation produced by UKB, we can follow the links to the Wikidata items and properties. This process allow us to properly associating the ERG predicates _benzene_n_1 to `Q2270` and _mass_n_of to `P2067`. The Wikidata item `Q2270` does not have a value for 'Wikidata property,' which means it is not an item used as a property of something. This disambiguation process will be revised to better use the Wikidata Lexeme data, which can have more flexible mappings to the items and properties of Wikidata.

From the last paragraph, we have enough information to instantiate some axioms in the ULKB theory, the Δ above. Axiom 4 tells us that something that is the argument of the ERG unary predicate _benzene_n_1 is the «wsd:benzene» item in Wikidata. Axiom 5 tells us that the ERG binary predicate _mass_n_of can be translated to the Wikidata property «wsd:mass». A Python function in ULKB can construct both axioms; these functions are actually macros in HOL. The function gets the ERG predicates and the mappings from UKB. From the mapping and the type (and arity) of the Wikidata entities (item or property), the function can instantiate the axioms from a set of templates.

$$\forall\, x, \_benzene\_n\_1\ x\ \ \rightarrow x = \text{«wsd:benzene»} \tag{4}$$

$$\forall\, x\ y, \_mass\_n\_of\ x\ y\ \ \rightarrow \exists\, v, \text{«wsd:mass»}\ y\ v \tag{5}$$

Example (1-b) would instantiate another axiom, the word 'toxicity' evokes an ERG unary predicate _toxicity_n_1 and not a binary predicate as the word 'mass.' The connection between the words 'benzene' and 'toxicity' is mediated by the abstract predicate poss (possessive) from ERG. The template that handles this case would produce the Axiom 6.

$$\begin{aligned}
&\forall\, x\ y\ z,\ \_toxicity\_n\_1\ y \wedge \\
&(\_of\_p\ z\ y\ x\ \vee\ poss\ z\ y\ x\ \vee\ compound\ z\ y\ x) \\
&\qquad\qquad \rightarrow \exists\, v, \text{«LD50»}\ x\ v
\end{aligned} \tag{6}$$

Provided the axioms 4 and 5 above, Z3 SMT Solver (de Moura and Björner, 2008) can easily

[8] This process can later be refined to use the Wikidata Lexemes.

prove the entailment $\Delta, T(\alpha) \models G(\alpha)$ certifying that the SPARQL query is indeed entailment by the HOL formula, the semantics of the original NL question.

We admit that Example (1-a) discussed above is quite simple. We have not addressed the more complicated cases with properties and entities expressed by more than one word (Sag et al., 2002) and complex expressions of units of measurement. The literature is vast on possible methods for linking entities and their use in domain-specific cases (Zhou et al., 2023). However, it is worth highlighting that (1) It seems that few templates deal with the most common cases of variants of syntactic constructions used in English questions we are considering, that is, questions in a technical domain such as chemistry; (2) any entity detection and entity disambiguation (also called entity linking) method can be equally employed in our framework; and (3) Since 2018, Wikidata has also stored linguistic data such as words, phrases, and sentences. This information is stored in new types of entities called Lexemes (L), Forms (F), and Senses (S). These entities can be linked appropriately to Q items and properties, facilitating the disambiguation process during the semantic parsing of the sentence and constructing the axioms above by demand.

5. Conclusions and Future Work

In conclusion, we have presented a logic-based approach to validate SPARQL queries derived from translations of natural language (NL) questions. Our focus on addressing the well-documented risks of hallucinations in KGQA amidst the widespread utilization of Large Language Models (LLMs) positions our work as a neuro-symbolic endeavor toward ensuring 'Safe AI.' While much previous research has also explored the use of semantic parsing for question-answering (Gu et al., 2022; Berant et al., 2013) – mainly using machine learning methods for semantic parsing and producing representations like AMR (Banarescu et al., 2013) – and evaluated LLMs in this context (Faria et al., 2023), the novelty of our approach lies in the use of Minimal Recursion Semantics (MRS) produced by ERG, a high-precision computational grammar, and the translation of MRS to higher-order logic (HOL) to represent the semantics of English sentences and the further compositional and deterministic translation of HOL formulas to SPARQL (query) and from SPARQL (to validate).

Central to our methodology is MRS Logic, a Python Library built upon 'deep' linguistic processing technologies from the DELPH-IN Consortium. By extending DELPH-IN tools to translate MRS to HOL formulas and employing ULKB to reason with these formulas and query KGs, our approach bridges linguistic and statistical processing methods for semantic understanding. To the best of our knowledge, our work is the first comprehensive report on the translation of MRS to a higher-order logic language, the subsequent translation of SPARQL to/from HOL, and the use of these methodologies for comparing English questions with SPARQL queries.

While a preliminary evaluation of the MRS Logic capability of translating NL utterances to HOL statements has been conducted using text entailment tests (Rademaker et al., 2023) in the SICK dataset (Marelli et al., 2014), we recognize the necessity of further evaluating our SPARQL validation procedure, mainly as we aim to tackle more challenging questions in domains like chemistry, leveraging insights from existing KGQA systems (Zhou et al., 2023). The translation from HOL to SPARQL is compositional and deterministic, but still, many nuances of NL utterances may not be captured adequately by our current implementation. At this stage, there is no dataset of NL queries in the chemistry domain associated with SPARQL with and without hallucinations to test our approach. Note that we focus on technical domains rather than on general-purpose common sense datasets like LC-QUAD (Trivedi et al., 2017). We are not dealing with unrestricted entities and their properties (people, places, events, etc.) that make entity recognition and entity and word sense disambiguation almost a guess without a reasonable context. The LLM pipeline for SPARQL generation was used precisely for its coverage and robustness, and it is unclear if the symbolic processing will capture few or many of the actual possible queries that a domain expert may submit. A quantitative evaluation of our approach will undoubtedly be necessary in subsequent work. It is worth mentioning the complexity of constructing a domain-specific QA dataset with questions that need to be relevant (not toy examples) and with different levels of complexity.

As part of our future endeavors, we aspire to reimplement our approach using Lean (Moura and Ullrich, 2021), a programming language and interactive theorem prover, thus transitioning from HOL to dependent types. Dependent type theory has been widely acknowledged as a formal tool for understanding natural language (Ranta, 1994; Chatzikyriakidis and Luo, 2020), and exploring this avenue could further enhance the robustness and applicability of our methodology.

6. Bibliographical References

Eneko Agirre and Aitor Soroa. 2009. Personalizing PageRank for word sense disambiguation. In

The 12th Conference of the European Chapter of the ACL (EACL 2009), pages 33–41, Athens, Greece. ACL.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

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.

Lang Cao. 2023. Enhancing reasoning capabilities of large language models: A graph-based verification approach. *arXiv preprint arXiv:2308.09267*.

Stergios Chatzikyriakidis and Zhaohui Luo. 2020. *Formal Semantics in Modern Type Theories*. Wiley.

Ann Copestake and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage english grammar using hpsg. In *The Second Linguistic Resources and Evaluation Conference*, pages 591–600, Athens, Greece.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A Sag. 2005. Minimal recursion semantics: An introduction. *Research on language and computation*, 3:281–332.

Berthold Crysmann and Woodley Packard. 2012. Towards efficient HPSG generation for German, a non-configurational language. In *COLING*, pages 695–710.

Leonardo de Moura and Nikolaj Björner. 2008. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, Berlin. Springer.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. Chain-of-verification reduces hallucination in large language models.

Tim Erdmann, Stefan Zecevic, Sarath Swaminathan, Brandi Ransom, Krystelle Lionti, Dmitry Zubarev, Siya Kunde, Stephanie Houde, James Hedrick, Nathaniel Park, and Kristin Schmidt. 2024. Chemchat: Conversational expert assistant in material science and data visualization. In *American Chemical Society (ACS) Spring Meeting*.

Bruno Faria, Dylan Perdigão, and Hugo Gonçalo Oliveira. 2023. Question Answering over Linked Data with GPT-3. In *12th Symposium on Languages, Applications and Technologies (SLATE 2023)*, volume 113 of *Open Access Series in Informatics (OASIcs)*, pages 1:1–1:15, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

William M. Farmer. 2008. The seven virtues of simple type theory. *Journal of Applied Logic*, 6(3):267–286.

D. Flickinger, A. Copestake, and I. A. Sag. 2000. Hpsg analysis of english. In *Verbmobil: Foundations of speech-to-speech translation*, pages 321–330. Springer, Berlin, Germany.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28.

Michael Wayne Goodman. 2019. A python library for deep linguistic resources. In *2019 Pacific Neighborhood Consortium Annual Conference and Joint Meetings (PNC)*, Singapore.

Yu Gu, Vardaan Pahuja, Gong Cheng, and Yu Su. 2022. Knowledge base question answering: A semantic parsing perspective. *arXiv preprint arXiv:2209.04994*.

Alexander Koller and Stefan Thater. 2005. Efficient solving and exploration of scope ambiguities. In *The ACL Interactive Poster and Demonstration Sessions*, pages 9–12, Ann Arbor, Michigan. ACL.

Alexander Koller and Stefan Thater. 2006. An improved redundancy elimination algorithm for underspecified representations. In *The 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 409–416, Sydney, Australia. ACL.

Alexander Koller and Stefan Thater. 2010. Computing weakest readings. In *The 48th Annual Meeting of the ACL*, pages 30–39, Uppsala, Sweden. ACL.

Guilherme Lima, Alexandre Rademaker, and Rosario Uceda-Sosa. 2023. Ulkb logic: A hol-based framework for reasoning over knowledge graphs. In *Proceedings of the 26th Brazilian Symposium on Formal Methods*.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of chain-of-thought reasoning.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *The Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland. ELRA.

George A. Miller. 1995. WordNet: A lexical database for English. *Commun. ACM*, 38(11):39–41.

Leonardo de Moura and Sebastian Ullrich. 2021. The lean 4 theorem prover and programming language. In *The 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings 28*, pages 625–635. Springer.

OpenAI. 2023. Gpt-4 technical report.

C. Pollard and I. A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. The University of Chicago Press and CSLI Publications, Chicago, IL and Stanford, CA.

Alexandre Rademaker, Guilherme Lima, and Renato Cerqueira. 2023. Extracting higher-order logic formulas from english sentences. Under evaluation.

Aarne Ranta. 1994. *Type-theoretical grammar*. Oxford University Press.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for nlp. In *Proceedings of the Computational Linguistics and Intelligent Text Processing*, pages 1–15.

Stephan Schulz, Simon Cruanes, and Petar Vukmirović. 2019. Faster, higher, stronger: E 2.3. In *Automated Deduction – CADE 27*, pages 495–507. Springer.

Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. 2017. LC-QUAD: A corpus for complex question answering over knowledge graphs. In *International Semantic Web Conference*, pages 210–218. Springer.

W3C SPARQL Working Group. 2013. SPARQL 1.1 overview. W3C recommendation, W3C. `http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/`.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models.

Yoad Winter. 2016. *Elements of Formal Semantics: An Introduction to the Mathematical Theory of Meaning in Natural Language*. Edinburgh University Press.

Xiaochi Zhou, Daniel Nurkowski, Sebastian Mosbach, Jethro Akroyd, and Markus Kraft. 2021. Question answering system for chemistry. *Journal of Chemical Information and Modeling*, 61(8):3868–3880.

Xiaochi Zhou, Shaocong Zhang, Mehal Agarwal, Jethro Akroyd, Sebastian Mosbach, and Markus Kraft. 2023. Marie and bert – a knowledge graph embedding based question answering system for chemistry. *ACS omega*, 8(36):33039–33057.