Softening Ontological Reasoning with Large Language Models

Teodoro Baldazzi[1], Davide Benedetto[1], Luigi Bellomarini[2], Emanuel Sallinger[3,4] and Adriano Vlad[3]

[1]Università Roma Tre, Italy
[2]Banca d'Italia, Italy
[3]TU Wien, Austria
[4]University of Oxford, United Kingdom

Abstract

Logic-based Knowledge Graphs (KGs) and Knowledge Representation and Reasoning (KRR) have emerged as fundamental methodologies in many data-intensive areas, fostering trust and accountability for effective decision-making. However, the knowledge captured by such approaches is often restricted by the rigidity of their structured rule-based formalisms. More recently, the rising adoption of Large Language Models (LLMs) has introduced a new layer of semantic understanding and flexibility in human-data interaction. Yet, these models are inherently limited in reasoning capabilities and lack systematic and explainable outcomes due to their opaque nature. To address today's challenge of combining the strengths of both technologies, we propose a novel neurosymbolic solution that leverages the power of LLMs to "soften" rule activations, enhancing adaptability in ontological reasoning while preserving robustness and transparency of KRR systems.

Keywords

Ontological reasoning, Language models, Knowledge graphs

1. Introduction

In recent years, the widespread interest in querying and exploiting large volumes of data has catalyzed the development of increasingly mature, efficient, and scalable solutions capable of capturing and reasoning over real-world scenarios. In this context, ensuring the transparency of data-driven processes is paramount to provide high levels of trustworthiness and accountability in decision-making, especially over critical domains such as finance and biomedicine [1, 2]. Powered by logic-based *Knowledge Representation and Reasoning* (KRR) formalisms, such *intelligent systems* are fully explainable [3], as they provide factual conclusions augmented with the consequentially logical steps that led to them through the inference. Among these formalisms, logic programming-based database query languages, such as Datalog and its extensions [4], are a yardstick, thanks to their effective trade-off between expressive power and computational complexity. Leveraging such languages, factual data from corporate databases can be combined with business-level definitions as ontologies in *Knowledge Graphs* (KGs), and further augmented via *ontological reasoning* [5].

However, ontological reasoning systems are constrained by the rigid nature of KRR formalisms at their foundation, which limits their adaptability to the complexities of real-world data. Indeed, these systems typically rely on query-based interactions, operating at a low level and often proving challenging for non-specialists to use effectively. Moreover, all inputs and outputs are confined to structured formats such as facts, n-tuples, or triples, and the generation of new knowledge through rule activation is restricted to what can be syntactically captured by predefined logical predicates and via

RuleML+RR'24: Companion Proceedings of the 8th International Joint Conference on Rules and Reasoning, September 16–22, 2024, Bucharest, Romania

teodoro.baldazzi@uniroma3.it (T. Baldazzi); davide.benedetto93@gmail.com (D. Benedetto); luigi.bellomarini@bancaditalia.it (L. Bellomarini); sallinger@dbai.tuwien.ac.at (E. Sallinger); adriano.vlad@gmail.com (A. Vlad)
0000-0002-1762-1431 (T. Baldazzi); 0000-0001-6079-4250 (D. Benedetto); 0000-0001-6863-0162 (L. Bellomarini); 0000-0001-7441-129X (E. Sallinger)

precise *bindings* to actual values. This rigidity fundamentally clashes with the inherent ambiguity of unstructured or raw data that may not fit into predefined categories. Together with the incompleteness, inconsistencies, and inaccuracies that might affect such data, these issues inhibit the applicability of KRR in real-world scenarios where understanding the semantic meaning of information is crucial. Consequently, we are observing a critical need for solutions that enable more semantic-aware and flexible reasoning capabilities in such systems.

As an intuitive example, let us consider the natural language (NL) sentence "*Through a series of five transactions, E. Musk has acquired 52% of Twitter in October 2022, after previously expressing interest in the platform during several interviews.*" and the logical rule *Owns(owner,owned,shares)*, $shares > 0.5 \rightarrow$ *Controls(owner,owned)*, stating that "*a financial entity owning more than 50% of the shares of another one controls it.* While a human would readily understand that *Elon Musk* now controls *Twitter*, automatically inferring this result presents significant challenges for a KRR system. Indeed, it should first recognize that, despite the unstructured nature of the input, the rule's body could bind to it, given the close semantic relationship between acquisition and ownership. Then, it would need to correctly *map* the arguments of the *Owns* predicate to the corresponding portions of the input, i.e., identifying *E. Musk* as the entity owning, *Twitter* as the entity owned, and 52% as the shares involved in the ownership. Moreover, information such as the number of transactions and the time-frame, while not affecting rule activation, still provides relevant context in the financial domain the example belongs to and should not be filtered out, whereas details like Musk's prior expressions of interest can be omitted. Finally, the rule would be activated, producing as output *Controls(E. Musk,Twitter)*, ideally augmented with such contextually-relevant details as additional metadata to enrich explainability.

The demand for solutions that enable more adaptable and flexible reasoning to navigate the intricacies and ambiguities of real-world data has gained further traction with the recent breakthrough of AI-based chatbots and Large Language Models (LLMs) [6], which has marked a significant turning point in the field of Natural Language Processing (NLP) and a pivotal shift in the access to data and knowledge towards more human-friendly and high-level paradigms. Today, LLMs such as OpenAI's GPT [7] and Meta's Llama [8] are effectively adopted to address a plethora of tasks across multiple domains [9]. Following the development of techniques such as chain-of-thought prompting [10], recent attempts have been carried out to employ LLMs for complex data analyses as well as multi-step reasoning [11, 12]. Yet, despite the advancements in the field and the proficiency of these models in handling semantic relationships within natural language, concerns persist due to their intrinsic opacity and unpredictability [13]. Indeed, they often fall short in providing systematic, explainable outcomes necessary for big data processing and robust decision-making in high-stakes domains [14, 15].

This paper addresses the challenge of synergistically combining the robustness and transparency of KRR systems with the power of LLMs in understanding the semantic meaning of NL knowledge. We propose a *neurosymbolic* solution that leverages LLMs to augment the ontological reasoning process with real-world semantic flexibility, injecting "*softness*" into rule activations. Specifically, we operate in the context of the Vadalog [16] system, a Datalog-based reasoning engine for KGs, that finds many industrial applications [17]. The semantics of a Vadalog set Σ of rules can be defined in an operational way via the well-known chase [18] procedure. Given a database in input, this algorithmic tool expands it with new facts entailed via the application of the rules in Σ, until all of them are satisfied. Intuitively, a rule is applied when an exact binding is identified, i.e., a set of mappings of the variables in the rule's body to the constants of structured facts in the database.

With the goal of extending the traditional chase mechanism to address the complexities of unstructured data, our approach leverages a pre-trained Llama 3 model to act as a *semantic unifier*, responsible for identifying bindings in the chase between rules and such data. In practice, given the next rule to be applied via the reasoner, both the rule in its natural language form and the candidate facts to activate it on are passed to the LLM. The model leverages its semantic understanding capabilities to generate bindings as sets of mappings from the variables of the rule body to the proper excerpts of the NL facts. These mappings then undergo a validation phase, which includes a feedback loop to confirm their correctness and coherence and to address potential hallucinations. Once validated, the resulting

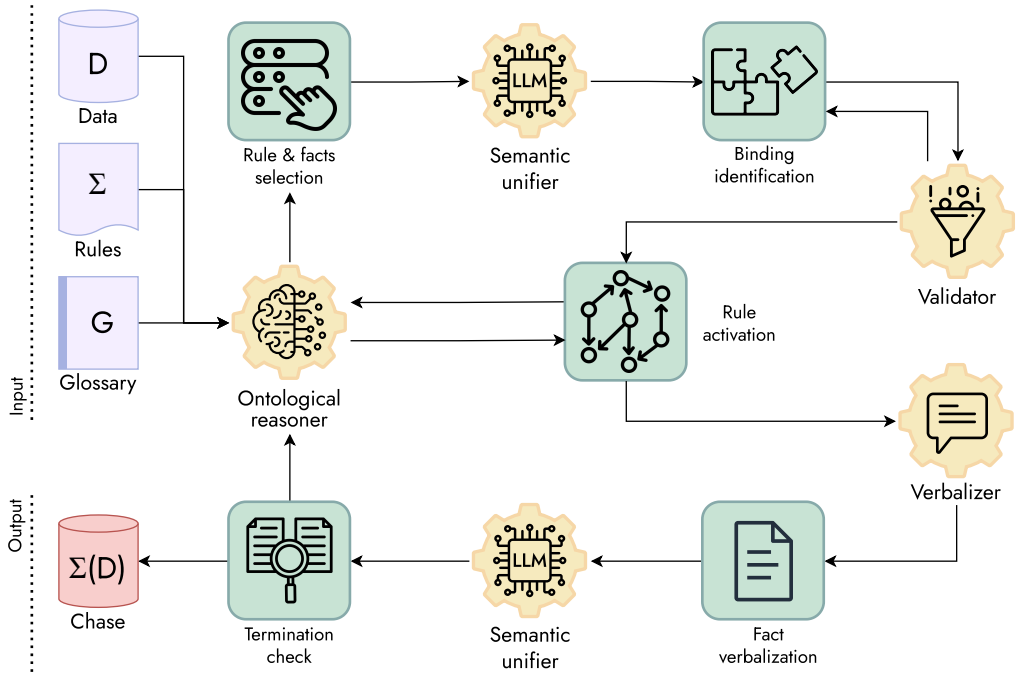

Figure 1: Neurosymbolic reasoning pipeline for LLM-powered soft CHASE. D represents input data collected from relational databases and natural language sources connected to the ontological reasoning system. Σ denotes the set of logic rules to be applied on D. $\Sigma(D)$ refers to the original data augmented with new knowledge inferred by applying the rules in Σ throughout the reasoning process.

binding is provided to the reasoner, which employs it to attempt rule activation. If all the conditions in the rule are satisfied, a new fact is inferred and additional details of the parents NL facts are preserved as CHASE metadata. Finally, the newly produced fact is *verbalized* into natural language via a dedicated module and a termination check is performed, leveraging again the LLM to ensure that the knowledge it provides had not already been generated at a previous step of the reasoning. If that is the case, the fact is added as new input in the CHASE, and the procedure continues until no more bindings can be identified. A high-level summary of the pipeline, illustrated in Figure 1, will guide our discussion.

More in detail, our **contributions** can be summarized as follows.

- We present a novel **soft CHASE technique** that extends logic rule bindings and termination control of traditional CHASE methodologies to unstructured data, leveraging the semantic awareness of LLMs and a deterministic verbalization of logic facts into NL.
- We deliver such an approach in a new **neurosymbolic KRR-centered architecture** (powered by VADALOG, but compatible with any CHASE-based reasoner) to enable more adaptable and flexible ontological reasoning while preserving robustness and explainability.
- We discuss a preliminary **experimental evaluation** confirming the validity of our approach and comparing standard CHASE with its soft counterpart, powered by pre-trained and *Retrieval-Augmented Generation* (RAG) [19]-enriched versions of the LLM.

Overview. In Section 2 we provide essential background notions. In Section 3 we present our proposed neurosymbolic architecture. A preliminary experimental evaluation is provided in Section 4. Section 5 discusses related work. We draw our conclusions in Section 6.

2. Chase-based Ontological Reasoning in the Vadalog System

To guide the rest of our discussion, we first lay out some preliminary notions on ontological reasoning over KGs, with a specific focus on the Vadalog system and the chase procedure at its foundation.

Relational foundations. Let $\mathbf{C}$ and $\mathbf{V}$ be disjoint countably infinite sets of *constants* and *variables*, respectively. A (*relational*) *schema* $\mathbf{S}$ is a finite set of relation symbols (or *predicates*) with associated arity. A *term* is either a constant or a variable. An atom over $\mathbf{S}$ is an expression of the form $R(\bar{v})$, where $R \in \mathbf{S}$ is of arity $n > 0$ and $\bar{v}$ is an n-tuple of terms. A *database* (*instance*) over $\mathbf{S}$ associates to each symbol in $\mathbf{S}$ a relation of the respective arity over the domain of constants. The members of the relations are called *tuples* or *facts*. Given two conjunctions of atoms ς_1 and ς_2, we define a *homomorphism* from ς_1 to ς_2 as a mapping $h : \mathbf{C} \cup \mathbf{V} \rightarrow \mathbf{C} \cup \mathbf{V}$ s.t. $h(t) = t$ if $t \in \mathbf{C}$ and for each atom $a(t_1, \ldots, t_n) \in \varsigma_1$, then $h(a(t_1, \ldots, t_n)) = a(h(t_1), \ldots, h(t_n)) \in \varsigma_2$.

Vadalog syntax. Vadalog is a declarative language for ontological reasoning. It is based on *Warded Datalog*$^{\pm}$, a member of the Datalog family that, at the price of very mild syntactic restrictions, extends Datalog with existential quantifiers and guarantees PTIME data complexity for query answering [20]. A Warded Datalog$^{\pm}$ program consists of a set of *tuples* (or *facts*) and *tuple-generating dependencies* (TGDs) of the form $\forall \bar{x} \forall \bar{y} (\varphi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, \psi(\bar{x}, \bar{z}))$, where $\varphi(\bar{x}, \bar{y})$ (the *body*) and $\psi(\bar{x}, \bar{z})$ (the *head*) are conjunctions of atoms over the respective predicates, $\bar{x}$, $\bar{y}$ are vectors of universally quantified variables and constants, and $\bar{z}$ is a vector of existentially quantified variables. Quantifiers can be omitted and conjunction is denoted by comma. In this context, Vadalog extends the Warded fragment with features of practical utility to address real-world scenarios [16]. Support for aggregate functions, namely *sum*, *prod*, *min*, *max* and *count*, is achieved by means of *monotonic aggregations* [21]. Other relevant extensions include *negations* and negative constraints of the form $\varphi(\bar{x}, \bar{y}) \rightarrow \bot$, where $\varphi(\bar{x}, \bar{y})$ is a conjunction of atoms and $\bot$ denotes the truth constant false to model disjointness or non-membership, as well as *expressions* in rule bodies, modelled with comparison ($>, <, \geq, \leq, \neq$) and algebraic ($+, -, *, /$, etc.) operators.

Chase Procedure. KRR approaches model KGs as the combination $\Sigma(D)$ of an *extensional component*, essentially the ground business data in a database D, and an *intensional component*, which formally describes the business knowledge as a set Σ of rules in a declarative language such as Vadalog. Performing ontological reasoning over the KG augments it with new inferred knowledge derived from the application of the rules over the input data. Specifically, the semantics of a Vadalog program can be defined in an operational way with the chase *procedure* [18]. It enforces the satisfaction of a set Σ of rules over a database D, incrementally augmenting D with facts entailed via the application of the rules over D, until fixpoint. While Vadalog guarantees that such fixpoint exists when only the core features are used [16], the joint presence of algebraic operations and recursion must be carefully handled, as even simple Datalog programs can be in general non-terminating [22]. A TGD $\sigma : \varphi(\bar{x}, \bar{y}) \rightarrow \psi(\bar{x}, \bar{z})$ is applicable to D if: (i) there exists a homomorphism θ (technically known as *binding*) such that $\theta(\varphi(\bar{x}, \bar{y})) \subseteq D$, that is, if there exists a set of *mappings* from the terms of $\varphi(\bar{x}, \bar{y})$ to the constants of facts in D such that each term maps to exactly one constant, and (ii) $\theta(\psi(\bar{x}, \bar{z}))$ is a fact not already present in D. If such a binding θ exists, then $\theta(\psi(\bar{x}, \bar{z}))$, derived by applying these mappings to the conclusion of the TGD, is added to D via a chase *step*. The chase *graph* $\mathcal{G}(D, \Sigma)$ is the directed acyclic graph with the facts from the chase $\Sigma(D)$ as nodes and an edge from a node n to a node m if m derives from n (and possibly other facts) via a chase step [4]. Dedicated works [23, 24] have thoroughly explored chase termination [22] in Vadalog in the presence of recursion and algebraic operations.

Vadalog reasoner. The Vadalog system is a state-of-the-art ontological reasoning engine that leverages the theoretical underpinnings of the chase procedure and the vast experience of the database community on provenance to power efficient, scalable, and explainable reasoning tasks over critical business domains and large KGs [16]. To achieve this, it adopts a *streaming* data processing architecture based on the *pipes and filters* style [16, 25]. Here, the set of rules Σ and the queries are translated into active *data scans* (linear scans for linear TGDs, join scans for join TGDs, and an output scan for the query), connected by intermediate buffers in a processing pipeline. The reasoning process is performed as a data stream along the pipeline, where each filter (i.e., scan) reads tuples from the respective parent, from the output

scan down to the external data stores that inject ground facts into the pipeline. Interactions between scans occur by means of primitives such as *next()*, which fetches facts from the parent stream, if present. Since, for each filter, multiple parent filters may be available, VADALOG selects which one to invoke by employing specific *routing strategies* (*round-robin*, *shortest path*, etc.) that manage a priority queue of the sources. This methodology allows VADALOG to keep track of the provenance of each result, derived from one or more CHASE steps. Unlike traditional *semi-naive* approaches [22], VADALOG generalizes the *volcano iterator model* [26], operating in a *pull-based query-driven* fashion in which, ideally, facts are materialized only at the end of the CHASE and if they contributed to the reasoning task.

3. Neurosymbolic Reasoning by Softening the Chase

The input blocks of the soft CHASE pipeline in Figure 1 are a set D of data, a set Σ of reasoning rules expressed in VADALOG, and a glossary G. Without loss of generality, we define D as the collection of structured data from relational databases D_s and unstructured data from natural language sources D_u, all connected to VADALOG for the reasoning task. The glossary G lists the predicates in Σ and their corresponding natural language descriptions.

Let us first introduce our running example. Here, D contains a collection of acquisitions and ownerships of companies' shares by financial entities in the market, both persons and other companies.

Example 1. *The following set Σ contains the VADALOG rules governing who has decision power in a financial entity, based on who owns, directly or indirectly via intermediaries, a significant amount of shares of the financial entity [27]*

$$Owns(x, y, s) \rightarrow OwnedShares(x, y, y, s) \qquad (\sigma_1)$$

$$SignificantShares(x, z), Owns(z, y, s) \rightarrow OwnedShares(x, z, y, s) \qquad (\sigma_2)$$

$$OwnedShares(x, \_, y, s), ts = msum(s), ts > 0.3 \rightarrow SignificantShares(x, y) \qquad (\sigma_3)$$

A financial entity x directly owning s shares of another financial entity y, owns such shares via y itself (rule σ_1). If x owns significant shares of a financial entity z and z owns s shares of y, then x owns s shares of y via z (rule σ_2). Finally, if x owns, directly or indirectly, a total amount of y's shares greater than 0.3, then x owns a significant portion of y's shares (rule σ_3).

Consider the following subset of data D_s = {*Owns*(*Elon Musk*, *Tesla*, 0.19), *Owns*(*Google LLC*, *DeepMind*, 0.7), *Owns*(*BlackRock*, *Google*, 0.4)}, and the query Q: "*what are all the entailed significant shares?*" as ontological reasoning task. Note that the example is not intended to reflect real-world dynamics.

In pure KRR settings, the set $\Sigma(D_s)$ is computed via the standard CHASE: starting from $\Sigma(D_s) = D_s$, it augments $\Sigma(D_s)$ with facts derived from the application of the rules in Σ up to fixpoint. Figure 2 illustrates the CHASE graph derived from the activation of Σ over D_s. Specifically, rule σ_1 generates *OwnedShares*(*Elon Musk*, *Tesla*, *Tesla*, 0.19), *OwnedShares*(*Google LLC*, *DeepMind*, *DeepMind*, 0.7), and *OwnedShares*(*BlackRock*, *Google*, *Google*, 0.4) representing the direct ownership entailed from the input facts. Then, *SignificantShares*(*Google LLC*, *DeepMind*) and *SignificantShares*(*BlackRock*, *Google*) are inferred via rule σ_3, whereas *Elon Musk* does not own significant shares of *Tesla* directly. Note that we cannot automatically derive, via rule σ_2 and rule σ_3, that *BlackRock* owns significant shares of *DeepMind* indirectly through *Google*, as rule σ_2 does not activate on the join argument $\langle$*Google LLC*, *Google*$\rangle$.

Reasoning with the soft CHASE. Let us now extend Example 1 by taking into account an additional source of information apart from D_s. For instance, consider the following input NL data D_u = {"*E. Musk bought 21% additional shares of Tesla in 2023*", "*Andy Jassy is CEO of Amazon since 2021*"}. Indeed, in this instance relevant information would be lost via the standard CHASE due to the absence of syntactic bindings from the rule bodies to NL knowledge in D_u. Thus, we extend binding identification by introducing the soft CHASE, in which an LLM acts as a semantic unifier between rule bodies and $\Sigma(D)$, $D = D_s \cup D_u$, injecting NL understanding capabilities into the reasoning process.

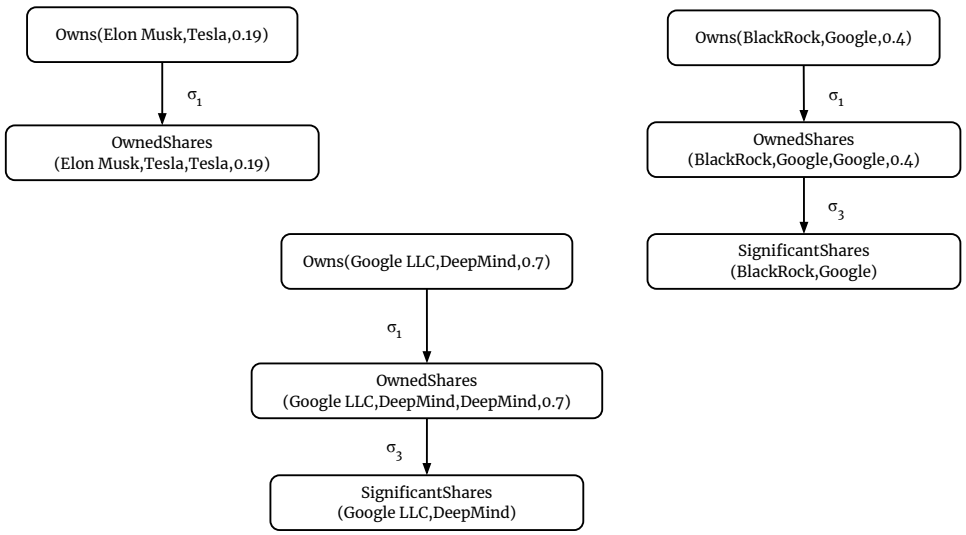

Figure 2: Instance of standard CHASE graph for Example 1.

Algorithm 1 Soft Chase Procedure.

```
1:  function SOFT_CHASE(D, Σ, G, model)
2:    Σ(D) ← D                                              ▷ initialize chase facts to D
3:    while VADALOG.hasNext() do                            ▷ continue until all rules and facts are processed
4:      σ, i ← VADALOG.next()                               ▷ fetch next rule and facts to process according to routing strategy
5:      i_mappings, attempts ← ∅, 0
6:      if LINEAR(σ) then
7:        i_mappings ← model.bindLinear(σ, i)               ▷ get mappings via LLM for linear rules
8:      else
9:        if JOIN(σ) then
10:         i_mappings ← model.bindAndMatchJoin(σ, i)       ▷ get mappings via LLM and check join conditions
11:     while attempts < LIMIT do
12:       feedback ← validate(i_mappings, model)            ▷ validate mappings
13:       if feedback == "OK" then
14:         break                                           ▷ exit loop if feedback is positive
15:       else
16:         i_mappings, attempts ← ∅, attempts + 1
17:         if attempts < LIMIT then
18:           i_mappings ← model.refineMappings(σ, i, feedback)   ▷ refine mappings based on feedback
19:     if i_mappings ≠ ∅ then
20:       i′.logic ← VADALOG.apply(σ, i_mappings)           ▷ activate rule via VADALOG
21:       i′.metadata ← storeMetadata(i_mappings)           ▷ preserve additional NL details as metadata
22:       i′.nl ← verbalize(i′,G)                           ▷ verbalize the new fact into NL
23:       if model.checkTermination(Σ(D), i′.nl) then       ▷ check termination via LLM
24:         Σ(D) = Σ(D) ∪ i′                                ▷ add newly generated fact to the chase
25:   return Σ(D)
```

Specifically, the soft CHASE can be distinguished into five distinct phases, discussed below for Example 1 with the aid of Algorithm 1 and Figure 1.

1. Initialization and rule selection. As in a standard CHASE procedure, we begin by initializing the set $\Sigma(D)$ of CHASE facts to the ground ones in D (line 2, in the algorithm). Next, we consider the data from $\Sigma(D)$ to activate the rules in Σ and generate new knowledge. Current rule and data to check for bindings are fetched via *next()* primitive in VADALOG, if present, according to a routing strategy. Let us assume that we are employing the default round-robin strategy. Let us also assume that each rule features both its logical form and a natural language description, easily produced as a preprocessing step by deterministically verbalizing the atoms in body and head into a *"Since {body}, then {head}"* sentence [28] according to *select-project-join* semantics and looking up the glossary G. Similarly, if the input facts belong to the D_s database, they are verbalized as well.

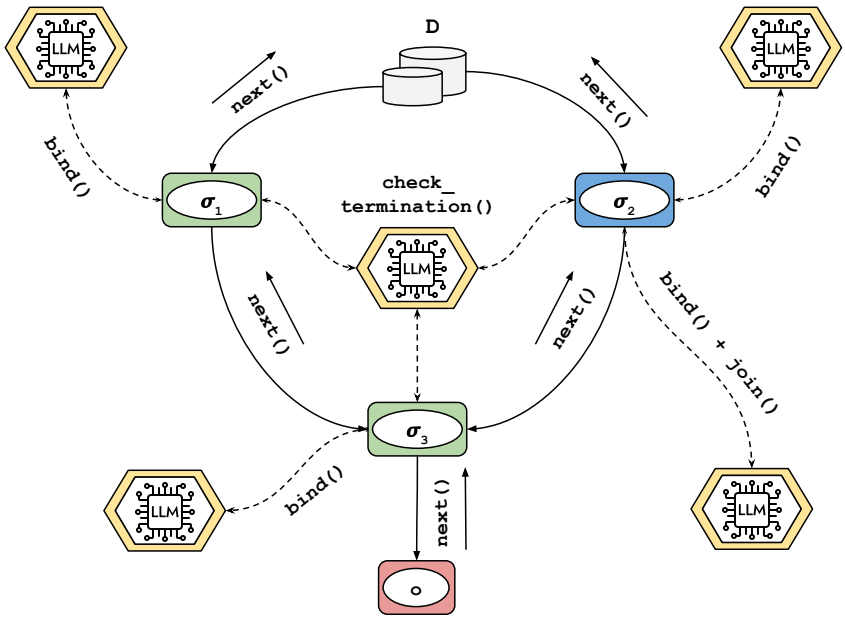

Figure 3: VADALOG processing pipeline of soft CHASE for Example 1. Green nodes are linear rules, the blue one is a join rule, and the red one is the output of the reasoning task. Solid edges are logical dependencies between the rules, and dashed ones denote an interaction with the semantic unifier of the type specified in the label (bind(), join(), check_termination()).

2. Binding identification. The goal of this step is identifying the possible binding of the current rule body with the input facts. To achieve this, the LLM is employed, acting as semantic unifier to generate a set of variable-to-constant mappings. Specifically, we operate with a pre-trained model, augmented only with some manually defined examples of mappings in a *few-shot learning* fashion to increase accuracy and limit hallucinations, both in the actual task and in the output format.

Here we observe distinct behaviours according to the type of the rule. Indeed, if the rule is *linear*, i.e., it features a single atom in the body (such as σ_1 in Example 1), the model only verifies whether there exists a set of mappings from the verbalized atom to excerpts of the NL fact (line 7). For instance, the NL fact "*E. Musk bought 21% additional shares of Tesla in 2023* " maps to the verbalized form of atom *Owns*(*x*,*y*,*s*), that is, "*A financial entity x owns s% shares of another financial entity y*". If a possible binding is identified, the LLM returns as output the structured set of mappings from the rule body to the fact, e.g., $\{ x \rightarrow \textit{E. Musk}, y \rightarrow \textit{Tesla}, s \rightarrow \textit{0.21} \}$, together with the details around the time-frame as additional metadata. Otherwise, it returns the empty set.

If instead the rule involves a *join*, first the model performs the same binding identification as in the linear case, for each individual atom in the body. Then, it further processes the resulting sets of mappings to check whether the values corresponding to the join variables match semantically (line 10), in which case the mappings are returned as output. For instance, the input fact "*Google LLC owns 70% shares of DeepMind*" (the NL version of *Owns*(*Google LLC*, *DeepMind*, 0.7)) and "*BlackRock owns significant shares of Google*" (the NL version of *SignificantShares*(*BlackRock*, *Google*)) match on the join argument $\langle \textit{Google LLC}, \textit{Google} \rangle$, unlike the standard CHASE approach discussed above.

3. Binding validation. After generating the candidate mappings i_{mappings}, a validation step occurs. Specifically, it first performs a deterministic check to ensure that all the variables in the body have been mapped to exactly one constant (e.g., an excerpt of the NL fact). This step is required to comply with the definition of binding as a homomorphism introduced in Section 2. Then, a separate LLM is employed as well, acting as a validator to confirm the response of the binding identification phase in a feedback loop fashion (lines 11-18). Indeed, if the candidate mappings do not pass the check, the cause of the issue is provided to the semantic unifier, which is tasked with repeating the step. A limit is enforced on the maximum number of attempts before considering the rule as unable to be bound to the current data.

4. Rule activation. If the set of mappings is not empty after validation, the logic rule can be deterministically activated via the Vadalog reasoner according to the binding. Before this, standard applicability checks of the rule occur, verifying the pre-existence of the unified head in $\Sigma(D)$. If that is not the case, and if additional conditions that might be present in the rule, such as selections and negations, are satisfied, the rule is activated and the new logic fact i' is inferred (line 20). Then, the fact is verbalized via the dedicated module and according to the glossary (line 22). For instance, from the binding $\{ x \rightarrow \textit{E. Musk}, y \rightarrow \textit{Tesla}, s \rightarrow \textit{0.21} \}$, rule σ_1 generates the fact *OwnedShares*(*E. Musk*, *Tesla*, *Tesla*, 0.21), verbalized into "*E. Musk owns 21% shares of Tesla directly*", with the specific time-frame of the parent fact as additional chase metadata, i.e., "*acquisition occurred in 2023* ".

5. Termination check. Finally, the resulting fact i' undergoes a semantic *termination check* to ensure that it is not already present in the chase instance $\Sigma(D)$. This step, essential to prevent loops in recursive settings, in the soft chase version goes beyond standard applicability checks as it limits redundancy of inferred knowledge throughout the reasoning by pruning facts whose semantic meaning has already been derived in a previous step. Thus, the semantic unifier is employed once again and the verbalized version of i' is semantically compared with the ones of the facts in $\Sigma(D)$ (line 23). Such a phase needs to be properly handled to prevent the removal of relevant facts. For instance, in our running example the fact "*E. Musk owns 21% shares of Tesla directly*" must be added to $\Sigma(D)$, thus it must not be pruned due to "*Elon Musk owns 19% shares of Tesla*". To address this, the LLM is enriched with specific examples, and the chase metadata of the compared facts is taken into account as well. If i' passes the check, it is added to $\Sigma(D)$ and the soft chase begins a new iteration, until fixpoint.

Extending soft chase with RAG. To further specialize the LLM into the domain of interest for the reasoning task, thus enabling a more accurate semantic unification throughout the procedure, we can also make available additional knowledge and terminology via RAG mechanisms. RAG enhances the model's contextual understanding by retrieving relevant documents or data points that contain specific information related to the concepts (i.e., the atoms and the facts) involved in the binding at hand. As further discussed in the next section, this proved to have a relevant impact in practical settings. For instance, in pure soft chase the fact *significantShares(Andy Jassy, Amazon)* is inferred from the NL input "*Andy Jassy is CEO of Amazon since 2021*" via rule σ_1, due to the mapping $\{ x \rightarrow \textit{Andy Jassy}, y \rightarrow \textit{Amazon}, s \rightarrow \textit{0.51} \}$, and then rule σ_3. In this instance, the LLM is incorrectly assuming that being CEO of a company entails owning the majority of its shares. We can prevent this incorrect inference by explicitly specifying, in the domain knowledge provided via RAG, that, in the absence of additional information, a CEO does not necessarily own any shares of the company at all.

Figure 4 illustrates the soft chase graph for Example 1. As further discussed in the next section, it can be observed how the soft chase variants augment the resulting chase instance with multiple relevant facts derived from LLM's semantic understanding of the domain.

4. Preliminary Experimental Evaluation

We integrated our proposed pipeline with the Vadalog system, although it can be integrated with any chase-based ontological reasoner. A full-scale evaluation of the architecture is beyond the scope of this work. Conversely, in this section we provide a preliminary comparison of standard and soft chase (in its pure and RAG-powered versions) over an instance of Example 1.

Setup. The experiments were conducted over a KG comprising ownership relationships between companies and persons as financial entities, represented using various nomenclatures such as full names, stock symbols, phrases, or common abbreviations. The KG featured inherent ambiguities and synonymous terms, reflecting real-world complexities and inconsistencies typical of semi-structured and unstructured corporate data. Moreover, natural language sentences describing ownership and acquisition facts were provided separately as input, simulating the scenario introduced in the previous section. We employed a pre-trained Llama 3 70B model as the semantic unifier.

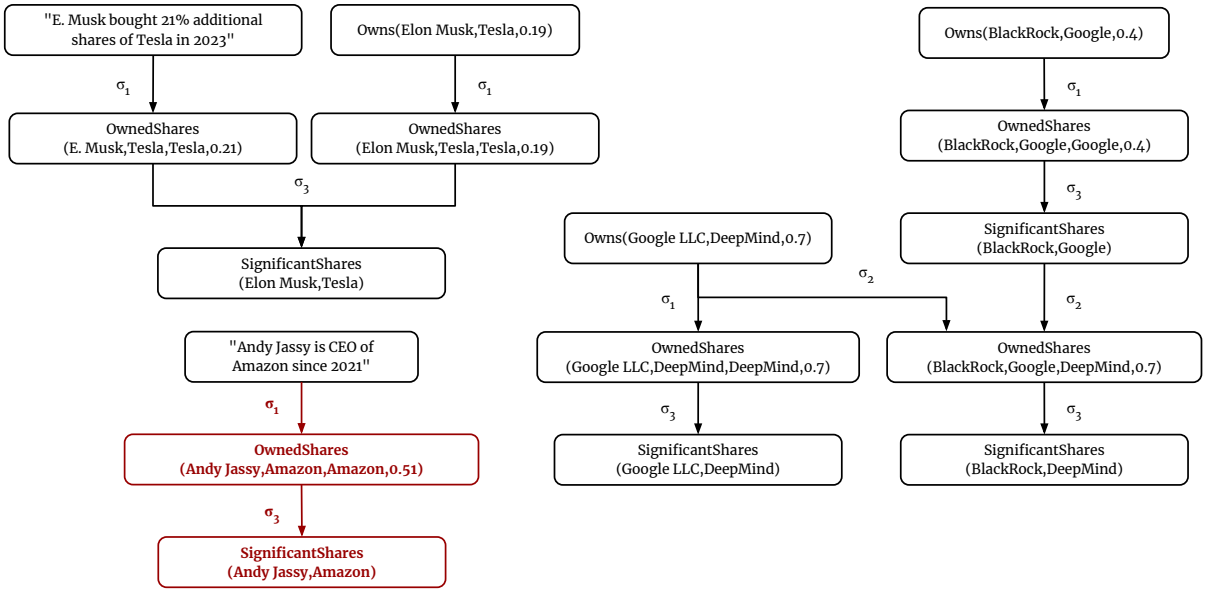

Figure 4: Instance of soft CHASE graph for Example 1. Red nodes and edges denote an incorrect derivation due to LLM hallucination, prevented in the version featuring RAG support.

Goal and Metrics. The primary goal of this evaluation is to assess the extent to which the injection of "softness" enhances the standard CHASE by recognizing similar entities and relationships according to real-world semantics. This enables augmenting the inference capabilities of the traditional approach while also preventing the generation of redundant data that represents the same knowledge in different syntactic forms. We conducted the experiments both before and after integrating the LLM with detailed knowledge of the domain of interest via RAG, with the purpose of further improving the model's accuracy in recognizing domain-specific entities and relationships. We compared the three distinct approaches (standard CHASE, soft CHASE, soft CHASE with RAG) according to the following metrics:

- *precision*, i.e., the fraction of inferred significant shares that are correct;
- *recall*, i.e., the fraction of correct significant shares that are inferred;
- *F1 score*, i.e., the harmonic mean of precision and recall;
- *false positive (FP) shares*, i.e., the fraction of incorrect significant shares that are inferred.

For this evaluation, the correct instances of significant shares were determined through a manually curated golden set, where domain experts verified the correctness of the inferred relationships.

Discussion. Results are illustrated in Figure 5. The standard CHASE featured full precision by definition, as it derived significant shares solely through strict logical binding with structured facts. However, this precision came at the cost of recall, as the standard CHASE was unable to bind rules to unstructured input, leading to missed inferences throughout the reasoning process. For instance, it failed to derive the direct relationship *OwnedShares (E. Musk,Tesla,Tesla,0.21)* from the input knowledge "*E. Musk bought 21% additional shares of Tesla in 2023*", and consequently did not infer that *Elon Musk* holds significant shares of *Tesla*. On the other hand, the soft CHASE demonstrated lower precision but higher recall, as it leveraged the LLM to recognize and semantically unify unstructured concepts with structured relationships. Furthermore, the introduction of RAG significantly improved both precision and recall, reducing the generation of incorrect facts such as *significantShares(Andy Jassy, Amazon)* from the input "*Andy Jassy is CEO of Amazon since 2021*". Indeed, the domain-specific knowledge provided by RAG effectively mitigated LLM hallucinations, reducing false positive bindings and consequently the incorrect inference of significant shares in the soft CHASE approach.

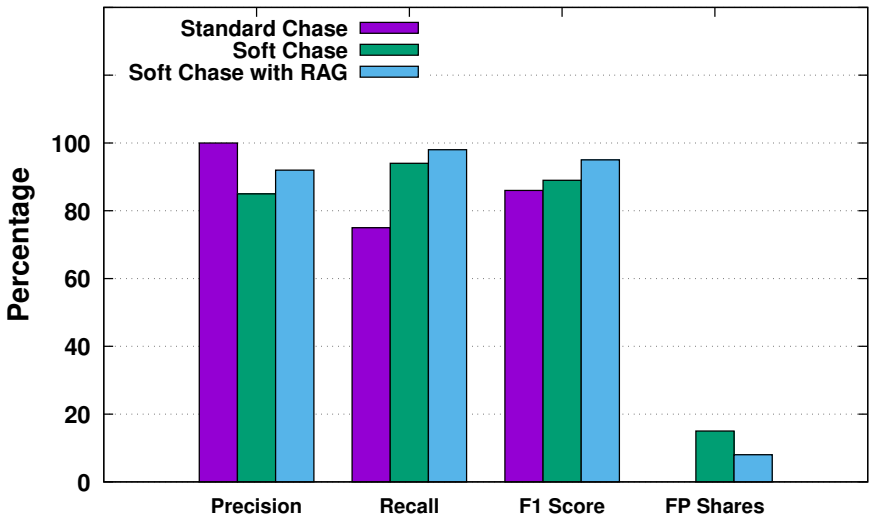

Figure 5: Comparison of precision, recall, F1 score, and FP shares for standard CHASE, soft CHASE, and soft CHASE with RAG, evaluated on instance of Example 1.

5. Related Work

Neurosymbolic methodologies are currently at the forefront of both academic and industrial research due to their potential in developing more intelligent, versatile, and explainable AI applications [29]. In this context, the integration of logic-based KGs and, more broadly, KRR approaches with LLMs has shown significant promise [30, 31]. Among the distinct forms of hybrid interactions between the two paradigms [32], studies have primarily focused on enriching LLMs with domain-specific knowledge encapsulated in KGs [28], as well as employing these models for tasks such as KG construction from unstructured text [33] and exploration [34].

A recent line of research involves integrating LLMs with foundational reasoning skills, modeling implicit structure information within the text and performing explicit logical reasoning over them to deduce the conclusion [35]. However, while these approaches improve reasoning capabilities, they often lack the robust, transparent reasoning structures that KRR systems inherently provide. To address this, frameworks like LOGIC-LM have been introduced, which first translates natural language problems into symbolic formulations using LLMs, and then employs a deterministic symbolic solver for inference [36].

To the best of our knowledge, this is the first approach that goes beyond the pure combination of LLMs with symbolic solvers to translate and solve specific logical problems. Our proposal is designed to seamlessly integrate LLMs within a KRR-centric framework to enhance ontological reasoning with semantic understanding throughtout the whole process, injecting human-like flexibility for complex real-world tasks while also preserving the inherent transparency of the paradigm.

6. Conclusion

In this paper, we addressed the limitations of traditional ontological reasoning systems, particularly their inherent rigidity in managing the intricacies and ambiguities of natural language data. We proposed a novel neurosymbolic approach that integrates Large Language Models as semantic interpreters between logic rules and such unstructured knowledge, enhancing the flexibility and robustness of rule activations. Our preliminary experiments demonstrate the effectiveness of our solution in preserving correctness and explainability while significantly improving adaptability. As future work, we aim to further refine the underlying formalism of our proposal and tackle challenges related to accuracy and scalability, particularly critical when processing large amounts of text as input knowledge for complex reasoning tasks. We believe this approach lays the foundation for deeper and more synergistic interactions between KRR systems and LLMs, fostering human-like reasoning in real-world contexts.

Acknowledgments

The work on this paper was partially supported by the Vienna Science and Technology Fund (WWTF) [10.47379/ICT2201, 10.47379/ VRG18013, 10.47379/NXT22018]; and the Christian Doppler Research Association (CDG) JRC LIVE.

References

[1] L. Bellomarini, L. Bencivelli, C. Biancotti, L. Blasi, F. P. Conteduca, A. Gentili, et al., Reasoning on company takeovers: From tactic to strategy, Data Knowl. Eng. 141 (2022) 102073.
[2] O. P. Dwyer, T. Baldazzi, J. Davies, E. Sallinger, A. Vlad, Reasoning over health records with Vadalog: a rule-based approach to patient pathways (2023).
[3] L. Caroprese, E. Vocaturo, E. Zumpano, Argumentation approaches for explanaible ai in medical informatics, Intelligent Systems with Applications 16 (2022) 200109. URL: https://doi.org/10.1016/j.iswa.2022.200109.
[4] A. Calì, G. Gottlob, T. Lukasiewicz, A general datalog-based framework for tractable query answering over ontologies, J. Web Semant. 14 (2012) 57–83. doi:10.1016/j.websem.2012.03.001.
[5] L. Bellomarini, G. Gottlob, A. Pieris, E. Sallinger, Swift logic for big data and knowledge graphs: Overview of requirements, language, and system, in: SOFSEM 2018: Theory and Practice of Computer Science: 44th International Conference on Current Trends in Theory and Practice of Computer Science, Krems, Austria, January 29-February 2, 2018, Proceedings 44, Springer, 2018, pp. 3–16.
[6] D. K. Kanbach, L. Heiduk, G. Blueher, M. Schreiter, A. Lahmann, The genai is out of the bottle: generative artificial intelligence from a business model innovation perspective, Review of Managerial Science (2023) 1–32.
[7] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al., Improving language understanding by generative pre-training, . (2018).
[8] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al., Llama 2: Open foundation and fine-tuned chat models, arXiv preprint arXiv:2307.09288 (2023).
[9] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, et al., A survey on evaluation of large language models, ACM Transactions on Intelligent Systems and Technology 15 (2024) 1–45.
[10] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, Y. Iwasawa, Large language models are zero-shot reasoners, NIPS 35 (2022) 22199–22213.
[11] D. Shu, T. Chen, M. Jin, Y. Zhang, M. Du, Y. Zhang, Knowledge graph large language model (kg-llm) for link prediction, arXiv preprint arXiv:2403.07311 (2024).
[12] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al., Chain-of-thought prompting elicits reasoning in large language models, NIPS 35 (2022) 24824–24837.
[13] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng, H. Cai, S. Wang, D. Yin, M. Du, Explainability for large language models: A survey, arXiv preprint arXiv:2309.01029 (2023).
[14] J. Fandinno, C. Schulz, Answering the "why" in answer set programming–a survey of explanation approaches, Theory and Practice of Logic Programming 19 (2019) 114–203.
[15] J. I. Hong, Teaching the fate community about privacy, Commun. ACM 66 (2023) 10–11.
[16] L. Bellomarini, D. Benedetto, G. Gottlob, E. Sallinger, Vadalog: A modern architecture for automated reasoning with large knowledge graphs, IS 105 (2022).
[17] L. Bellomarini, D. Fakhoury, G. Gottlob, E. Sallinger, Knowledge graphs and enterprise AI: the promise of an enabling technology, in: ICDE, 2019, pp. 26–37.
[18] C. Beeri, M. Y. Vardi, A proof procedure for data dependencies, Journal of the ACM (JACM) 31 (1984) 718–741.

[19] P. Lewis, E. Perez, A. Piktus, F. Petroni, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, NeurIPS 33 (2020) 9459–9474.
[20] G. Gottlob, A. Pieris, Beyond sparql under owl 2 ql entailment regime: Rules to the rescue, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.
[21] A. Shkapsky, M. Yang, C. Zaniolo, Optimizing recursive queries with monotonic aggregates in deals, in: 2015 IEEE 31st International Conference on Data Engineering, IEEE, 2015, pp. 867–878.
[22] S. Abiteboul, R. Hull, V. Vianu, Foundations of databases, volume 8, Addison-Wesley Reading, 1995.
[23] L. Bellomarini, E. Sallinger, G. Gottlob, The vadalog system: Datalog-based reasoning for knowledge graphs, Proc. VLDB Endow. 11 (2018) 975–987. URL: https://doi.org/10.14778/3213880.3213888. doi:10.14778/3213880.3213888.
[24] T. Baldazzi, L. Bellomarini, E. Sallinger, P. Atzeni, Reasoning in warded datalog+/-with harmful joins., in: SEBD, 2022, pp. 292–299.
[25] T. Baldazzi, L. Bellomarini, M. Favorito, E. Sallinger, Ontological reasoning over shy and warded datalog+/–for streaming-based architectures, in: International Symposium on Practical Aspects of Declarative Languages, Springer, 2024, pp. 169–185.
[26] G. Graefe, W. J. McKenna, The volcano optimizer generator: Extensibility and efficient search, in: ICDE, IEEE Computer Society, 1993, pp. 209–218.
[27] A. Gulino, S. Ceri, G. Gottlob, E. Sallinger, L. Bellomarini, Distributed company control in company shareholding graphs, in: IEEE 37th International Conference on Data Engineering (ICDE), Los Alamitos, CA, USA, 2021, pp. 2637–2648.
[28] T. Baldazzi, L. Bellomarini, S. Ceri, A. Colombo, A. Gentili, E. Sallinger, Fine-tuning large enterprise language models via ontological reasoning, in: International Joint Conference on Rules and Reasoning, Springer, 2023, pp. 86–94.
[29] A. d. Garcez, L. C. Lamb, Neurosymbolic ai: The 3 rd wave, Artificial Intelligence Review 56 (2023) 12387–12406.
[30] X. L. Dong, Generations of knowledge graphs: The crazy ideas and the business impact, arXiv preprint arXiv:2308.14217 (2023).
[31] K. Hamilton, A. Nayak, B. Bozic, L. Longo, Is neuro-symbolic AI meeting its promise in natural language processing? A structured review, CoRR abs/2202.12205 (2022). URL: https://arxiv.org/abs/2202.12205. arXiv:2202.12205.
[32] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, X. Wu, Unifying large language models and knowledge graphs: A roadmap, arXiv preprint arXiv:2306.08302 (2023).
[33] M. Trajanoska, R. Stojanov, D. Trajanov, Enhancing knowledge graph construction using large language models, arXiv preprint arXiv:2305.04676 (2023).
[34] T. Baldazzi, L. Bellomarini, S. Ceri, A. Colombo, A. Gentili, E. Sallinger, " please, vadalog, tell me why": Interactive explanation of datalog-based reasoning., in: EDBT, 2024, pp. 834–837.
[35] S. Wang, Z. Wei, J. Xu, T. Li, Z. Fan, Unifying structure reasoning and language model pre-training for complex reasoning, arXiv preprint arXiv:2301.08913 (2023).
[36] L. Pan, A. Albalak, X. Wang, W. Y. Wang, Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning, arXiv preprint arXiv:2305.12295 (2023).