Explaining Enterprise Knowledge Graphs with Large Language Models and Ontological Reasoning

Teodoro Baldazzi
Università Roma Tre, Italy

Luigi Bellomarini
Banca d'Italia, Roma, Italy

Stefano Ceri
Politecnico di Milano, Italy

Andrea Colombo
Politecnico di Milano, Italy

Andrea Gentili
Banca d'Italia, Roma, Italy

Emanuel Sallinger
TU Wien, Austria
University of Oxford, UK

Paolo Atzeni
Università Roma Tre, Italy

Abstract

In recent times, the demand for transparency and accountability in AI-driven decisions has intensified, particularly in high-stakes domains like finance and bio-medicine. This focus on the provenance of AI-generated conclusions underscores the need for decision-making processes that are not only transparent but also readily interpretable by humans, to built trust of both users and stakeholders. In this context, the integration of state-of-the-art Large Language Models (LLMs) with logic-oriented Enterprise Knowledge Graphs (EKGs) and the broader scope of Knowledge Representation and Reasoning (KRR) methodologies is currently at the cutting edge of industrial and academic research across numerous data-intensive areas. Indeed, such a synergy is paramount as LLMs bring a layer of adaptability and human-centric understanding that complements the structured insights of EKGs. Conversely, the central role of ontological reasoning is to capture the domain knowledge, accurately handling complex tasks over a given realm of interest, and to infuse the process with transparency and a clear provenance-based explanation of the conclusions drawn, addressing the fundamental challenge of LLMs' inherent opacity and fostering trust and accountability in AI applications. In this paper, we propose a novel neuro-symbolic framework that leverages the underpinnings of provenance in ontological reasoning to enhance state-of-the-art LLMs with domain awareness and explainability, enabling them to act as natural language interfaces to EKGs.

2012 ACM Subject Classification Computing methodologies → Knowledge representation and reasoning; Computing methodologies → Natural language processing; Theory of computation → Data provenance

Keywords and phrases provenance, ontological reasoning, language models, knowledge graphs

Digital Object Identifier 10.4230/OASIcs.Tannen.1

Funding 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.
Stefano Ceri: S.C. is supported by the PNRR-PE-AI FAIR project, funded by the NextGenerationEU program.
Andrea Colombo: A.C. kindly acknowledges INPS for the funding of his Ph.D. program.

1 Introduction

In today's data-driven industrial landscape, adhering to the principles of Fairness, Accountability, Transparency, and Ethics (FATE) has become paramount for AI applications [44]. Indeed, the absence of transparency in AI's decision-making processes prevents stakeholders and users from assessing their fairness, detecting potential biases, and verifying their overall reliability. This is particularly relevant in high-stakes domains such as finance and biomedicine, where there is an increasing demand for a clear, comprehensible, natural language explanation of AI's conclusions (i.e., their *provenance*) to back trustworthy critical decisions. Such a feature would act as a bridge between the opaque inner workings of AI and human comprehension, fostering informed decision-making and mitigating risks associated with black-box models, in favour of users' trust and adherence to ethical standards [35].

This requirement has gained further traction with the recent breakthrough of AI-based chatbots and Large Language Models (LLMs) [46], 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 natural, user-friendly, and high-level paradigms. Notably, LLMs such as OpenAI's GPT [55] and Meta's Llama [61] have transcended traditional academic and industrial applications, capturing the general public's attention towards generative AI capabilities. However, concerns persist regarding their lack of factual knowledge and accuracy over enterprise domains, even when fine-tuning is involved [5], and, more importantly, their opaqueness due to a very limited explainability of their conclusions [66].

Conversely, traditional *Knowledge Representation and Reasoning* (KRR) [48] approaches are inherently *domain-aware* and explainable [24]. Indeed, logic-based reasoning in query answering, often referred to as *ontological reasoning* [22], allows for FATEness, as it is designed to provide factual conclusions augmented with the logically consequential steps, in the form of top-down logical inference, that led to such results [25, 24]. For this reason, in the database and the AI communities, we are observing the surge of increasingly mature, efficient, and scalable *intelligent systems with reasoning capabilities*, backed by expressive logic-based KRR formalisms. Among them, database query languages based on logic programming, such as Datalog and its extensions [1, 22, 20, 21], are a yardstick for AI systems rooted in ontological reasoning, thanks to their effective trade-off between expressive power and computational complexity [11, 28, 48]. Leveraging such systems, domain-specific knowledge can be captured by combining factual data from corporate databases with business-level definitions as ontologies in *Enterprise Knowledge Graphs* (EKGs), and further augmented by reasoning over them. Despite this, the interaction remains query-based, often operating at a low level and lacking flexibility, thus proving itself challenging for non-specialists.

In light of these considerations, it becomes clear why *neuro-symbolic* methodologies are at the forefront of academic and industrial interest. Indeed, their goal to synergistically combine the intrinsic domain-expertise and transparency of deductive systems with the power of LLMs in understanding and generating fluent and interpretable text holds immense potential to build more intelligent, versatile, and explainable AI-based applications on KGs, paving the way for a new era of transparent data-driven decision-making within organizations [54, 31, 42].

With the goal of contributing to such a pivotal challenge, this paper strives to strengthen LLMs in their use as explainable NL interfaces to EKGs by leveraging the power of ontological reasoning. In simple terms, our goal consists in *enabling the answer to "why this conclusion" questions in natural language and posed by the user over an EKG*. We operate in the context of the Vadalog [14, 8] system, a Datalog-based reasoning engine for knowledge graphs, that finds many industrial applications [12, 15, 10, 4, 3, 32, 62]. We employ Vadalog

to explore the factual information derived by applying the domain rules in high-stakes domains of interest, via the well-known CHASE procedure [53] of databases. This enables us to augment LLAMA 2-70B models with the derived domain knowledge and the provenance of the inferred conclusions, while preserving their human-like orientation and flexibility at handling question-answering tasks in natural language.

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

- We present a **chase verbalization technique** to enhance the domain expertise and explanation capabilities of LLMs by leveraging ontological reasoning over knowledge graphs and use cases in relevant domains.
- We deliver such an approach in KGLM, a **novel neuro-symbolic pipeline** to build LLM-powered explainable interfaces to EKGs by interacting with CHASE-based reasoning engines such as the VADALOG system.
- We provide a **practical application** of KGLM by integrating it within KG-ROAR [13, 6], a framework we developed to showcase VADALOG reasoning on financial use cases.

Related Work. Numerous studies have been conducted to investigate different aspects of provenance, namely its tracking, storing, and presentation [18, 26, 36, 58, 43, 47]. A longstanding challenge is dealing with the complexity of provenance expressions, with the goal of presenting them in a user-comprehensible form. To this aim, semirings models [40, 39, 56, 59] are the benchmark for their ability to present the provenance in an efficient and mathematically elegant form. These structures capture the two key aspects of data usage in provenance: joint contribution (represented by addition) and alternative sources (represented by multiplication). This allows for a concise representation of how various inputs contribute to the final output, enabling reasoning about aspects like confidence, access control, and cost associated with the data lineage. However, this kind of representation is still not easily accessible by non-expert users. In other studies, some graph-based representations of the provenance have been proposed [23, 27, 60], allowing, for instance, user control over the provenance graph, i.e., by visually tracking contributors and sources. While they have the advantage of enhancing user interaction and inspection, they are still not the most natural way of interacting and understanding the provenance. A more recent line of research attempts to present provenance and answer to queries in natural language [33, 29, 49, 30, 19]. In most cases, these efforts only support queries of low complexity and the NL sentence depends on the quality of an input query asked in natural language. In other cases, explanations are fragmented, presenting rules without a cohesive narrative [19]. Additionally, linking provenance representation to an input query does not allow unlocking innovative uses of such precious information, such as exploiting it for generating a training corpus for LLMs. Current models are very effective for general tasks, but often struggle when it comes to specific domains. For example, it has been shown that FinBert [51] and BloombergGPT [64], two fine-tuned models on financial textual data, outperform generic LLMs on question-answering tasks. In such a context, provenance information from business reasoning tasks could be a valuable resource for generating a corpus of domain-specific knowledge to be injected into an LLM for fine-tuning or via *Retrieval-Augmented Generation* [50] (RAG) mechanisms.

Overview. The remainder of this paper is organized as follows. In Section 2 we provide essential background notions on ontological reasoning and introduce the VADALOG system. In Section 3 we illustrate the verbalization technique for LLM explanation and the KGLM pipeline. Section 4 delves into the application of KGLM within KG-ROAR. Our conclusions are drawn in Section 5.

2 Ontological Reasoning in the Vadalog System

To guide our discussion, we first lay out some preliminary notions on ontological reasoning over enterprise knowledge graphs, 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*.

Dependencies. A VADALOG program consists of a set of tuples and *tuple-generating dependencies* (TGDs), i.e., function-free Horn clauses 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. A predicate is *intensional* (IDB) if it occurs in at least one head of the schema $\mathbf{S}$, otherwise it is *extensional* (EDB) [1, 25, 38]. A fact corresponding to an intensional predicate is intensional, otherwise it is extensional.

Vadalog Extensions. Real-world applications may require support for multiple features that extend the declarative language. Among them, aggregate functions, namely *sum*, *prod*, *min*, *max* and *count*, as well as SQL-like grouping constructs, are particularly relevant. In the VADALOG context, support for aggregate functions is achieved by means of *monotonic aggregations* [63]. Other essential extensions, integrated in VADALOG to address real-world scenarios, 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.

Reasoning Task. KRR approaches model KGs as the combination of an *extensional component*, essentially the ground business data in a database, and an *intensional component*, which formally describes the business knowledge as a set 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 business data. More formally, given a database D and the query $Q = (\Sigma, Ans)$, where Σ is the set of rules and Ans an n-ary predicate, a reasoning task consists of finding an instance J such that a tuple $\bar{t} \in J$ if and only if $\bar{t} \in Q(D)$ and for every other instance J' such that $\bar{t} \in J'$ if and only if $\bar{t} \in Q(D)$, there is a homomorphism h from J to J'.

Chase Procedure. The semantics of a VADALOG program can be defined in an operational way with the *CHASE procedure* [45, 53]. 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 [14], the joint presence of algebraic operations and recursion must be carefully handled, as even simple Datalog programs can be in general non-terminating [1].

A TGD $\sigma : \varphi(\bar{x}, \bar{y}) \rightarrow \psi(\bar{x}, \bar{z})$ is applicable to D if there exists a homomorphism θ such that $\theta(\varphi(\bar{x}, \bar{y})) \subseteq D$. Then, a *chase step* adds the fact $\theta(\psi(\bar{x}, \bar{z}))$ to D, if not already in D. 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 [20]. A comprehensive examination of reasoning termination in VADALOG has been thoroughly explored in dedicated works [16, 9].

▶ **Example 1** (Trading Activity). Let us consider a simple trading activity managed with a *smart contract*. Here, D contains a log over time of buy/sell orders from the traders who invest in it as well as market information, e.g., asset prices (*Price*), or market shutdowns (*MarketClosed*). The following set Σ contains the VADALOG rules governing the basic functioning of the market, i.e., under which conditions the orders are accepted and how profits and losses are computed.

$$Open(x, y, t_1), \neg\, MarketClosed(t_1) \rightarrow Accepted(x, y, t_1) \qquad (\sigma_1)$$

$$Accepted(x, y, t_1), Price(p_1, t_1), k = y * p_1 \rightarrow Position(x, y, k, t_1) \qquad (\sigma_2)$$

$$Close(x, t_2), Price(p_2, t_2), Position(x, y, k, t_1), \\ t_2 > t_1, pl = y * p_2 - k \rightarrow Return(x,pl) \qquad (\sigma_3)$$

If a trader x wants to open a position (buy) on a certain asset of size y at time t_1 and the market is open at t_1, the order is accepted (rule σ_1). If the order by x is accepted and the asset price at t_1 is p_1, then x holds a position on the market at time t_1 of size y and of notional (total value) k equal to $y * p_1$ (rule σ_2). If, later at t_2, trader x decides to close its position (sell) and the price at t_2 is p_2, then x gets returns (profits or losses) from its trading activity as $y * p_2 - k$ (rule σ_3).

Let us also consider an excerpt of database $D = \{Open(EGTech,0.3,1),\ Price(124,1),\ Price(147,9),\ Close(EGTech,9),\ MarketClosed(5)\}$. Figure 1 illustrates the CHASE graph derived from the activation of Σ over D. Specifically, rule σ_1 generates the fact *Accepted*(*EGTech*,0.3,1), as the market is not closed at time 1. Then, the fact *Position*(*EGTech*,0.3, 37.2,1) is derived via rule σ_2. Finally, as trader *EGTech* closes the position, i.e., sells the asset, at time 9 and the price goes up to 147$, then *EGTech* gets a profit of 6.9$ as return via rule σ_3.

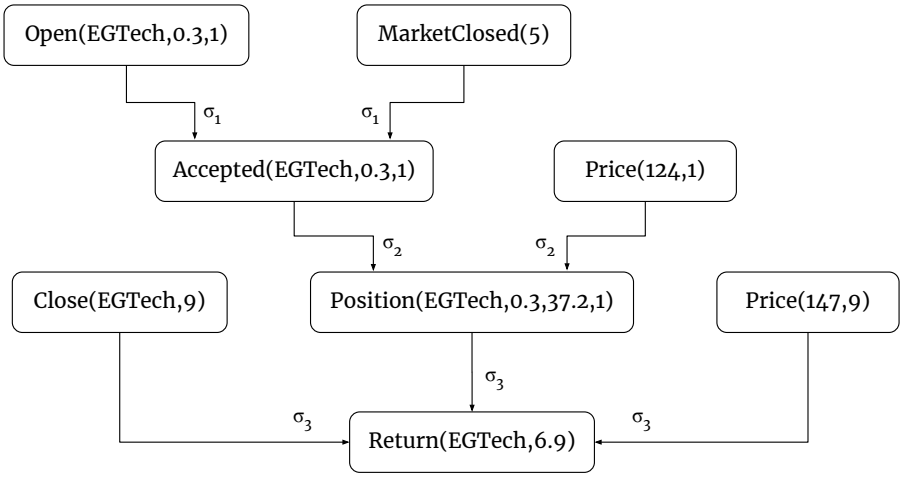

Figure 1 Instance of CHASE graph for Example 1.

The Vadalog System. 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 enterprise KGs [8].

To achieve this, it adopts a *streaming* data processing architecture based on the *pipes and filters* style [14, 7]. Here, the set of logic 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 *open()*, *next()*, *get()*, *close()*, which open the parent stream, ask for the presence of a fact to fetch, obtain it, and close the communication, respectively. Since, for each filter, multiple parent filters may be available, VADALOG selects which one to invoke (via *next()* call) 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 [1], VADALOG generalizes the *volcano iterator model* [37], operating in a *pull-based query-driven* fashion in which, ideally, facts are materialized only at the end of the CHASE procedure and if they contributed to the reasoning task.

Figure 2 illustrates VADALOG processing pipeline for the scenario in Example 1 given, as ontological reasoning task, the query q of finding the returns for the trader *EGTech*. Here the output filter o sends a *next()* message to the query processor, which propagates it to the TGDs scans. Note that each filter in the figure is labelled with the corresponding rule number from the above scenario.

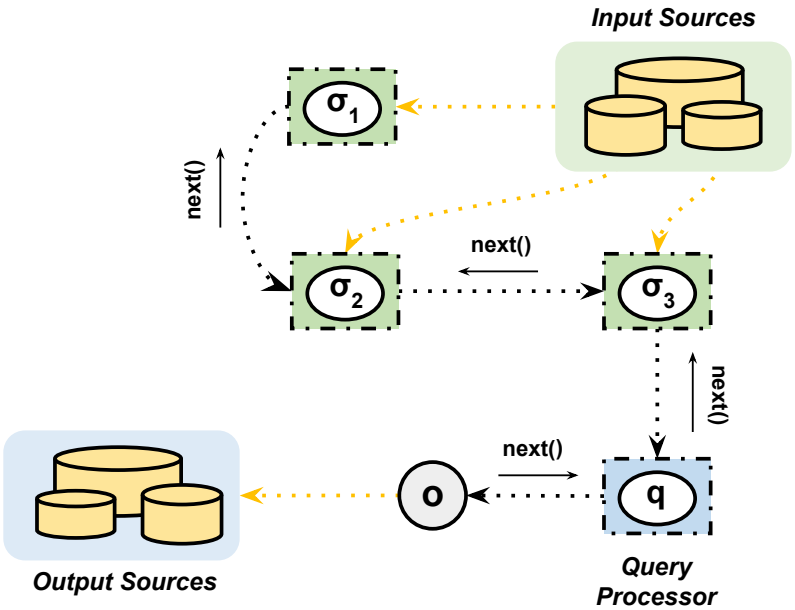

Figure 2 Processing pipeline of VADALOG for Example 1.

3 Overview of the KGLM Pipeline

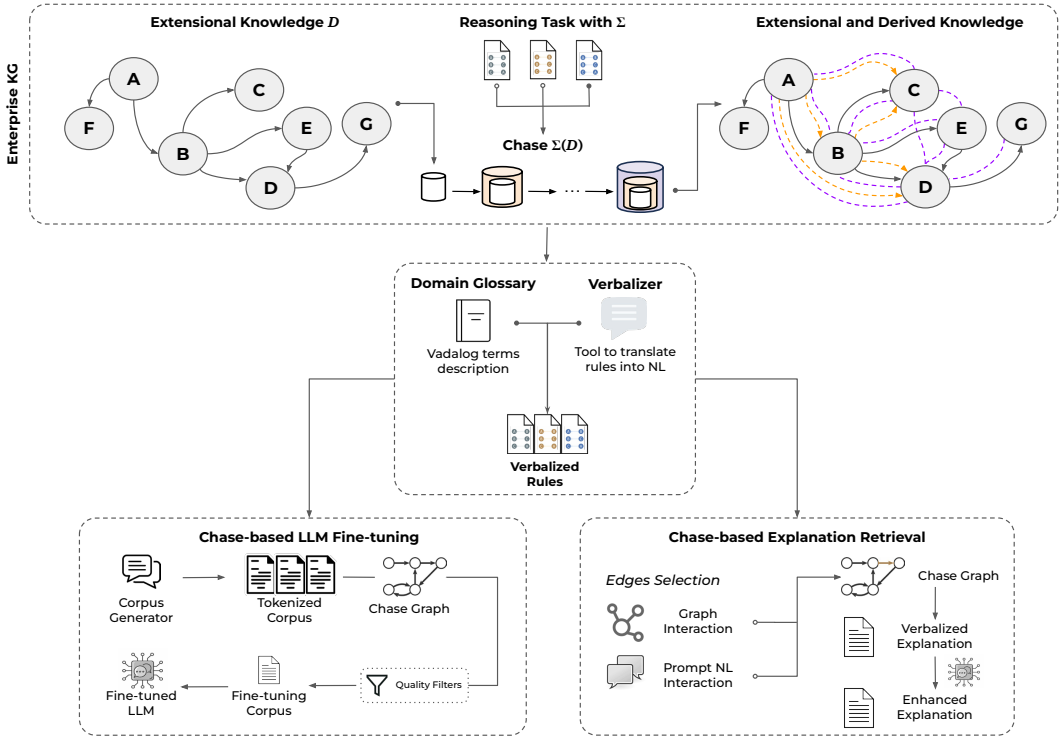

Figure 3 Visual overview of the KGLM pipeline.

Logic-oriented Enterprise Knowledge Graphs and Knowledge Representation and Reasoning approaches are at the forefront of explainable AI, which is as of today an area of paramount importance for both the academic community and the industry sector. Current research in KRR focuses on developing ontological reasoning systems, such as VADALOG, that effectively combine factual knowledge (extensional knowledge) with formally defined domain expertise (intensional knowledge) expressed through logical rules. In this context, integrating state-of-the-art Large Language Models with logic-oriented EKGs holds immense promise for developing explainable and human-oriented AI tools. A significant challenge, however, lies in the mismatch between the technologies. In fact, KRR systems require queries to be formulated in their specific formalisms, with knowledge generation restricted to what reasoning rules can capture. In contrast, LLMs lack the comprehensive domain models that are a cornerstone of KRR approaches. One promising approach involves leveraging the strengths of both paradigms in a synergistic fashion. Indeed, LLMs excel at natural language generation, allowing them to translate the complex relationships and entities within EKGs into human-understandable explanations. EKGs, in turn, provide LLMs with a foundation of factual knowledge, provenance, and ontological reasoning, thus mitigating the risk of inconsistencies and biases often present in LLMs' outputs.

Vadalog-powered KGLM. Specifically, we identify two main synergies between LLMs and logic-based EKGs. On the one hand, we can leverage the reasoning output, enriched by the provenance of each conclusion, to inject domain-specific knowledge for fine-tuning an LLM.

On the other hand, we can exploit the text manipulation capabilities of such models to enable a natural language navigation of the EKG, presenting its inferred edges and provenance in a human-like form. In the context of the VADALOG system, we recall that the provenance is provided by leveraging the CHASE procedure. Indeed, the CHASE graph resulting from a reasoning process corresponds to a directed acyclic graph representation of how each output fact was derived from the input facts by applying one or more VADALOG rules.

We develop such synergies between VADALOG-powered EKGs and LLMs in KGLM, the first, to the best of our knowledge, neuro-symbolic framework to enhance state-of-the-art LLMs with domain awareness and explainability, making them suitable to act as natural language interfaces to EKGs. Indeed, as further showcased later in this section and in Section 4, the features provided by KGLM are highly complementary, and have as their foundation a so-called *verbalizer* module that deterministically translates the KG into natural language sentences. Such *verbalizations* can then either be used as input to build question-answering fine-tuning corpora for LLMs, or they can be composed to retrieve the NL explanation of how a certain result was derived in the CHASE, which can then be injected into the LLM to achieve more fluent explanations that, in industrial applications, represent readable business reports. Indeed, these integrations can potentially enable a proper neuro-symbolic reasoning behavior by combining unstructured information with internal business rules to derive novel knowledge.

Reasoning Use Cases. To support our discussion, let us consider two scenarios involving a financial Enterprise Knowledge Graph. Specifically, we refer to an *ownership* KG, where the extensional component consists of *Owns* relationships, as depicted in Figure 4.

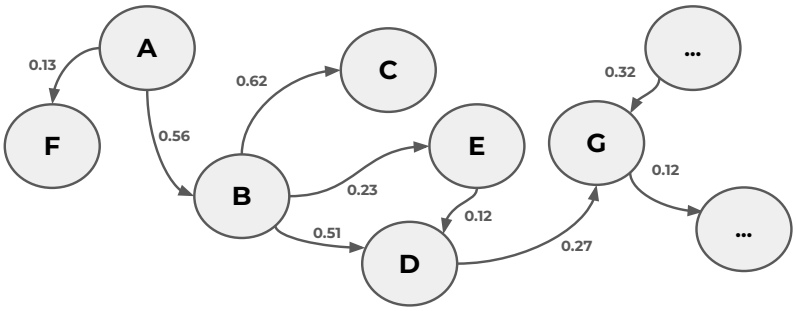

Figure 4 Portion of ownership knowledge graph. Nodes are companies. Solid edges are *owns* relationships with their shares.

On top of the KG, we can formalize an intensional component into VADALOG rules, which augment the graph with new knowledge in the form of novel edges.

▶ **Example 2** (Company Control)**.** This scenario allows analysts to understand who has decision power in companies, based on who controls the majority of votes, in a "one-share one-vote" assumption. To this end, the task augments the ownership graph with "*control*" edges, as follows [41]: *A company x directly owning s shares of a company y, controls such shares via y itself* (rule σ_4). *If x controls a company z and z owns s shares of y, then x controls s shares of y via z* (rule σ_5). *Finally, if x controls the majority of the shares of y, directly or indirectly, then x controls y* (rule σ_6).

$$Owns(x, y, s) \rightarrow ControlledShares(x, y, y, s) \qquad (\sigma_4)$$

$$Control(x, z), Owns(z, y, s) \rightarrow ControlledShares(x, z, y, s) \qquad (\sigma_5)$$

$$ControlledShares(x, \_, y, s), ts = sum(s), ts > 0.5 \rightarrow Control(x, y) \qquad (\sigma_6)$$

▶ **Example 3** (Close Link). Integrated ownership refers to the total stake a single entity holds in another entity, considering both direct and indirect ownership throughout the graph, with finite or infinite paths [52]. The value of integrated ownership I is calculated as $lim_{\epsilon \to 0} \sum_{P_i \in B_\epsilon} w_\epsilon(P_i)$, where B_ϵ is the set of all paths $P = [x, p_1, \ldots, p_k, y]$ in the ownership graph such that $x \neq p_i$ for $i = 1, \ldots, k$, and where $w_\epsilon(P) = \Pi_{(p_i,p_j) \in P} w(p_i, p_j) > \epsilon$, with $w(p_i, p_j)$ representing the direct ownership of p_i on p_j, $\epsilon \in \mathbb{R}^+$, and $0 < \epsilon \leq 1$. Note that integrated ownership differs from simple ownership used in Example 2. Applying this formulation to the regulation of the European Central Bank [34], we can say that x is in *close link* with y if: *(i) the integrated ownership of x on y is at least* 20% (rule σ_7); *(ii) y is in close link with x* (rule σ_8); *(iii) there is a third company z, whose integrated ownership on x and y is at least* 20% (rule σ_9).

$$IntOwns(x, y, s), s > 0.2 \rightarrow CloseLink(x, y) \quad (\sigma_7)$$

$$CloseLink(x, y) \rightarrow CloseLink(y, x) \quad (\sigma_8)$$

$$IntOwns(x, y, s_1), IntOwns(x, z, s_2), s_1 > 0.2, s_2 > 0.2, x \neq y, x \neq z, y \neq z \rightarrow CloseLink(y, z) \quad (\sigma_9)$$

The result of the application of the above VADALOG rules to the excerpt of the ownership Knowledge Graph is shown in Figure 5. In the rest of the section, we will provide an overview of how KGLM operates to address the two LLM-EKG synergies introduced, namely, *chase-based LLM fine-tuning* and *chase-based explanation retrieval*. Figure 3 illustrates the high-level pipeline of the framework.

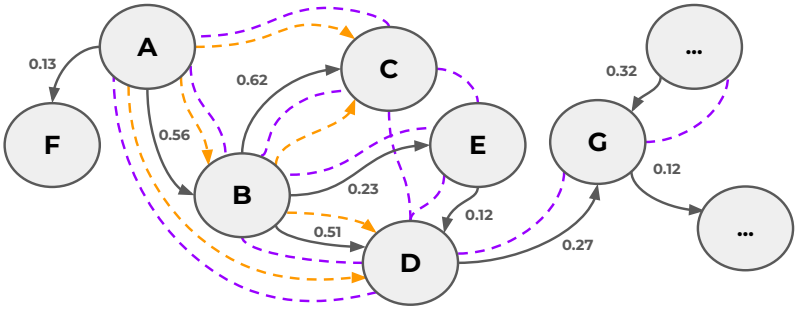

Figure 5 Portion of ownership knowledge graph from Figure 4 augmented with new edges by reasoning on the scenarios from Example 2 and 3. Nodes are companies. Solid edges are *owns* relationships with their shares (*int owns* omitted to avoid clutter). Dashed edges respectively denote *control* relationships (orange, directed) and *close links* (purple, undirected).

3.1 Chase-based LLM Fine-tuning

The goal of this KGLM task is to synthesize question-answering fine-tuning corpora that cover the entire "reasoning space" of the domain of interest, conveying domain-specificity and provenance-awareness to the LLM. Algorithm 1 and Algorithm 2 provide the pseudo-code describing the task, corresponding to the left path of the pipeline in Figure 3.

The first step consists in the execution of the ontological reasoning task of interest with the VADALOG system. Given an extensional knowledge D and the set Σ of VADALOG rules encoding the domain knowledge as the intensional component, the KG is built by augmenting D with the new inferred edges. Consequently, the corresponding CHASE graph $\Sigma(D)$ is generated (line 2 Algorithm 1). In the presence of results deriving from the application of

Algorithm 1 Reasoning and plan verbalization in KGLM.

```
1:  function ReasoningAndPlanVerbalization(D, Σ, glossary)
2:    chase ← Vadalog.reason(D, Σ)                          ▷ chase generation
3:    aggrChase ← ∅
4:    for each step in chase do                             ▷ chase aggregation
5:      stepAggrContrib ← ∅
6:      if hasAggregate(step.getRule()) then
7:        stepAggrContrib ← collectAggrContributors(step, chase)
8:      aggrStep ← aggregateStep(step, stepAggrContrib)
9:      aggrChase ← verbChase ∪ {aggrStep}
10:   verbRules ← verbalizeRules(Σ, glossary)               ▷ rules verbalization
11:   verbPlan ← verbalizePlan(Σ.getLogicPlan(), verbRules)
12:   return (aggrChase, verbPlan)
```

Algorithm 2 Chase-based LLM fine-tuning task in KGLM.

```
1:  function ChaseBasedFineTuning(aggrChase, verbPlan, model, threshold)
2:    tokenizedCorpus ← filter(generate(verbPlan))          ▷ tokenized corpus generation
3:    chaseCorpus ← ∅
4:    for each aggrStep in aggrChase do                     ▷ chase corpus generation
5:      chasePromptResp ← map(tokenizedCorpus, aggrStep)
6:      chaseCorpus ← chaseCorpus ∪ {chasePromptResp}
7:    for each pair ⟨prompt, resp⟩ in chaseCorpus do        ▷ quality-driven optimization
8:      qualityScore ← checkQuality(⟨prompt, resp⟩)
9:      if qualityScore ≤ threshold then
10:       chaseCorpus ← chaseCorpus \ {⟨prompt, resp⟩}
11:     else
12:       chaseCorpus ← chaseCorpus ∪ paraphrase(⟨prompt, resp⟩)
13:   fineTuningCorpus ← postprocess(chaseCorpus)
14:   fineTunedModel ← fineTune(model, fineTuningCorpus)    ▷ model fine-tuning
15:   return fineTunedModel
```

aggregation functions, the CHASE is further processed to collect all the contributors that led to the resulting aggregated value by *unfolding* the corresponding path of CHASE steps altogether [2, 9] (line 9 Algorithm 1).

Here, as previously introduced, our framework uses a *verbalizer* component to transform Σ into a set of natural language sentences via a deterministic transformation. To achieve this, the verbalizer is equipped with a *domain glossary*, containing descriptions of the terms and predicates involved in the rules (line 10 Algorithm 1). In the case of the *company control* scenario introduced in Example 2, the glossary in Figure 6 could be employed.

Predicate	Description
$Owns(x, y, s)$	<x> owns <s> shares of <y>
$ControlledShares(x, z, y, s)$	<x> controls <s> of <y> via <z>
$Control(x, y)$	<x> controls <y>

Figure 6 Domain glossary for the *company control* scenario.

Plan Parts	Verbalized Plan	Tokenized Corpus	Fine-Tuning Corpus
$Owns(x,y,s) \rightarrow ControlledShares(x,y,y,s)$ $\downarrow$ $ControlledShares(x,y,y,s)$, $\text{ts} = \text{sum(s)}$, $\text{ts} > 0.5$ $\rightarrow Control(x,y)$	*Since <x> owns <s> shares of <y>, then <x> controls <s> of <y> via <y>. Since <x> controls <s> of <y> via <y> and <ts> is the sum of <s> and <ts> is higher than 0.5, then <x> controls <y>*	Q1: How does <x> exercise control over <y>? A1: <x> controls <y> as it owns <s> shares, which is the majority	Q1: How does A exercise control over B? A1: A controls B as it owns 0.56 shares, which is the majority
		Q2: How does <x> have a majority over <y>? A2: <x> controls <y> because it owns a total combined number <ts> of its shares, through its subsidiaries	Q2: How does A have a majority over C? A2: A controls C because it owns a total combined number 0.62 of its shares, through its subsidiaries
...		...	...
$IntOwns(x,y,s)$, $s > 0.2$ $\rightarrow CloseLink(x,y)$	*Since <x> has an integrated ownership of <s> over <y>, and <s> is over 0.2, then <x> and <y> are in close link*	Q1: Are <x> and <y> in a close link relationship? A1: <x> and <y> are in a close link relationship, as the integrated ownership <s> of <x> over <y> is higher than 0.2	Q1: Are B and D in a close link relationship? A1: B and D are in a close link relationship, as the integrated ownership 0.51 of B over D is higher than 0.2
...		...	...
$Open(x,y,t_1)$, $\neg MarketClosed(t_1)$ $\rightarrow Accepted(x,y,t_1)$ $\downarrow$ $Accepted(x,y,t_1)$, $Price(p_1,t_1)$, $k = y * p_1$ $\rightarrow Position(x,y,k,t_1)$	*Since the trader <x> at time <t_1> sends an order to open a position of size <y> and it is not true that <t_1> is a time when the market is closed, then the order of size <y> by <x> is accepted at time <t_1>. Since the order of size <y> by <x> is accepted and the price is <p_1> at time <t_1>, then <x> holds a position of size <y> and notional <k> at time <t_1>*	Q1: When did <x> send an order to open a position with notional <k>? A1: The order to open that position was sent at time <t_1>	Q1: When did EGTech send an order to open a position with notional 37.2? A1: The order to open that position was sent at time 1
		Q2: Why was the order sent by trader <x> at time <t_1> accepted? A2: Because at time <t_1> the market was open	Q2: Why was the order sent by trader EGTech at time 1 accepted? A2: Because at time 1 the market was open
...		...	...

Figure 7 Generation of domain-specific corpora from scenarios in Examples 1-3 for LLM fine-tuning. Parts of interest of the plan are extracted, verbalized and passed to an LLM for generating the Q&A pair. Then, based on the actual CHASE graph, the final corpus is generated.

The NL translation of the rules leverages the *select-project-join* semantics, rewriting logic-based rules into textual "*since-then* closures". All VADALOG syntactic elements are converted into their textual counterparts. Conjunctions are rendered as *"and"* tokens, built-in operators are represented with specific keywords, e.g., $>$ becomes "*is higher than*", and the same occurs for aggregations, e.g., $x = count(y)$ becomes "x *is the total number of* y", etc. Moreover, rule variables are converted into tokens acting as placeholders. For instance, with respect to Example 2, the rule $Owns(x,y,s) \rightarrow ControlledShares(x,y,y,s)$ (rule σ_4) is verbalized as: "$Since < x > owns < s > shares\ of < y >, then < x > controls < s > of < y > via < y >$".

Then, with the verbalization of the rules available, we can generate a fine-tuning corpus to train an LLM to answer novel questions, which might relate to simple information retrieval tasks or more complex ones, i.e., involving an effort toward reasoning on novel input data. To generate the fine-tuning corpus, we exploit the effectiveness of powerful pre-trained LLMs [17], such as GPT-4 and LLAMA 2, in text understanding and manipulation capabilities. In fact, we can ask it to act as a human agent and synthesize a set of possible prompt-response pairs based on an input text. Note that here we have two goals: 1) minimizing the number of "calls" to the LLM, for cost- and time-efficiency reasons; 2) avoiding any ground value (coming from the EKG) being disclosed to the LLM, for data protection reasons. Thus, we leverage

the regularity of logical languages and resort to a *lifting technique* based on the creation of a *logic plan* out of Σ (line 11 Algorithm 1). Intuitively, a plan is the equivalent in our context of a database execution plan and can be seen as the dependency graph of the rules of Σ, where nodes represent rules and edges stand for head-body dependencies (similarly, at a high level, to VADALOG's processing pipeline introduced in Section 2). By sending natural language excerpts of the plan, obtained from the aggregation of the verbalized rules, we can automatically generate a corpus of tokenized question-answers, which capture the knowledge encoded in the rules and their interconnections (line 2 Algorithm 2). With such an approach, we achieve a corpus generation pipeline that is cost- and time-effective and that protects confidential data. The tokenized set of prompt-response pairs is then passed on to *filters*, which discard low-quality pairs. For instance, pairs in which the LLM generates new tokens are discarded. Such a step could also resort to a human-in-the-loop approach: in fact, as the pairs are a finite and relatively small set of *templates*, a validator could be hired to discard non-informative ones.

Finally, we materialize the actual fine-tuning corpus via the CHASE graph. For each new fact derived from CHASE procedure, we look up the corresponding verbalized portion of the plan and the tokenized corpus pairs. Each pair is instantiated by mapping the tokens to the corresponding constant arguments of the fact (line 4 Algorithm 2). In Figure 7, we provide some examples of tokenized corpora and their instantiations over artificial data for each of our presented use cases. The corpus undergoes a more thorough quality check where each pair is filtered according to a BERT-score-based scoring model that returns a so-called *R-score*, to evaluate generated text on coherence and factual consistency with the input text [65] (line 7 Algorithm 2). The threshold under which a pair is dropped can be selected by the user. The filtered-in pairs are enhanced via *NLP paraphrasing* to improve generalization, cleansed with additional post-processing procedures, and finally injected into the LLM for domain-specific question-answering fine-tuning (line 14 Algorithm 2).

3.2 Chase-based Explanation Retrieval

The CHASE-based fine-tuning discussed above enables us to inject into the LLM the awareness of the full domain of interest, encapsulating both factual data and the logical connections between them in the form of provenance. However, due to the *closed-book* nature of the approach [57], we empirically observed how the fine-tuned model alone is not able to effectively address complex questions regarding why and how a certain fact exists in the augmented KG. Indeed, answering such *explanatory questions* often involves composing back along the provenance paths that led, from extensional facts, to infer intensional ones in the reasoning process. To address this, we extended KGLM with a dedicated module that supports the LLM in correctly handling such questions by retrieving the CHASE-based explanations and injecting them into the model to enrich its answers. Algorithm 1 and Algorithm 3 provide the pseudo-code describing the task, corresponding to the right path of the pipeline in Figure 3.

The procedure assumes that the first steps, introduced in the context of the LLM fine-tuning task, have already occurred. The reasoning task was performed by VADALOG (line 2 Algorithm 1), the CHASE graph was generated, contributors to aggregations were collected (line 9 Algorithm 1), and the logic plans were translated into combinations of natural language sentences corresponding to the rules in Σ (line 11 Algorithm 1).

Now, let us consider a user interacting with the KG and asking questions about a possible explanation of generated facts. As summarized in Figure 8, this can be performed by either selecting the corresponding edge of the KG or in NL with a prompt-based interaction,

Algorithm 3 Chase-based LLM explanation retrieval task in KGLM.

```
1:  function ChaseBasedExplRetrieval(chase, verbPlan, model, glossary, query)
2:    fact ← extractFact(model, query, glossary)              ▷ queried fact extraction
3:    factStep ← extractStep(chase, fact)
4:    explanationSubgraph ← {factStep}
5:    predecessors ← {factStep}
6:    while not predecessors.isEmpty() do                     ▷ explanation subgraph creation
7:      currentStep ← predecessors.dequeue()
8:      currentPreds ← getPredecessors(currentStep, chase)
9:      for each predStep in currentPreds do
10:       if not predStep in explanationSubgraph then
11:         explanationSubgraph ← explanationSubgraph ∪ {predStep}
12:         predecessors.enqueue(predStep)
13:   detExplanation ← concatenate(map(verbPlan, explanationSubgraph))
14:   refExplanation ← model.refine(detExplanation)
15:   return model.answer(query, refExplanation)
```

depending on the application KGLM is running into. In both cases, the LLM can extract from the question the corresponding fact of interest, whose explanation is requested by leveraging the model's acquired knowledge of the KG and the glossary (line 2 Algorithm 3).

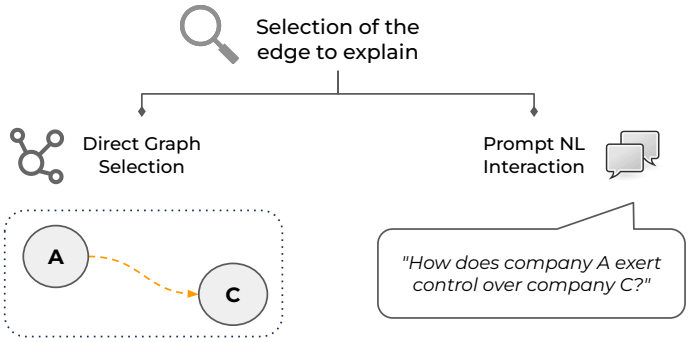

Figure 8 Possible interactions with KGLM to select facts to explain.

At this point, the actual retrieval process begins. First, the CHASE step that led to the inference of the fact is identified in the CHASE graph (line 3 Algorithm 3). As we are interested in the full explanation of the fact, we then perform a *back-composition* along the provenance paths (according to the previously mentioned *unfolding* approach [2, 9]), from the fact of interest up to the extensional ones that began such sequences of rule activations during the reasoning process. The result is an *explanation subgraph* of the CHASE graph, featuring both the facts and the activated rules in the paths (line 6 Algorithm 3). For instance, let us consider the portion of ownership KG in Figure 5, augmented with *control* edges derived from the reasoning task. Now, we may wonder *"How does company A exert control over company C?"* and submit such a request to KGLM as an NL prompt-based interaction. First, the LLM identifies the fact "$Control(A, C)$" of interest. Once the fact has been correctly identified, the corresponding trace is retrieved from the CHASE graph, as shown in Figure 9.

Next, we leverage the previously generated verbalized plan, mapping its tokens to the corresponding constant arguments of the facts in the subgraph. By concatenating the resulting verbalized CHASE steps, we obtain the deterministic NL explanation of the fact

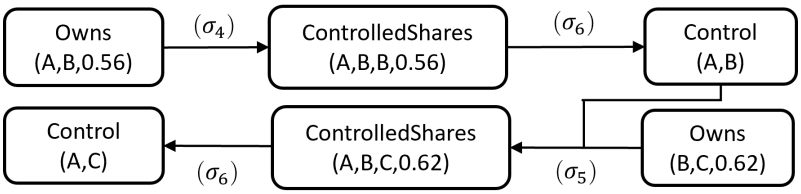

Figure 9 An instance of explanation subgraph for the *company control* scenario.

of interest as a sequence of *"Since {body}, then {head}"* sentences (line 13 Algorithm 3). Note that the final order of the verbalizations reflects the breadth-first traversal of the graph and thus respects logical dependency. However, the explanation produced by the verbalizer can be long, complex, and hard to read, especially in the presence of complex provenance paths. For instance, continuing with our example, we get the following explanation for fact "*Control*(A, C)": *"Since A owns* 0.56 *shares of B, then A controls* 0.56 *of B via B. Since A controls* 0.56 *of B via B and* 0.56 *is higher than* 0.5*, then A controls B. Since A controls B and B owns* 0.62 *shares of C, then A controls* 0.62 *of C via B. Since A controls* 0.62 *of C via B and* 0.62 *is higher than* 0.5 *then A controls C"*. To make it more understandable, we leverage the text manipulation capabilities of our model, improving fluency and clarity of the result. By prompting the LLM with the following request: *"Please produce a more readable version of the explanation: ..."*, we achieve a refined report that is highly accurate in content and comprehensible (line 14 Algorithm 3). For instance, the above explanation becomes: *"A's direct ownership of* 0.56 *of B translates to its control over B. With B's ownership of* 0.62 *of C, A, through B, also controls* 0.62 *of C. As the percentage exceeds* 0.5*, A effectively controls C"*. Finally, the explanation is passed to our domain-aware model, acting as KG interface, to answer the original user question (line 15 Algorithm 3).

4 KGLM Integration for Financial Applications

We recently introduced KG-Roar [13], a framework we created to showcase ontological reasoning with Vadalog on real-world cases that can be suitably modeled with a KG. It consists of a web-based environment designed for the interactive development and navigation of logical KGs derived from augmenting an input graph database with intensional definitions of new nodes and edges in the form of Vadalog programs. Such programs are user-defined or pre-built code snippets encapsulated in *reasoning widgets* (as illustrated in Figure 10), and act as a metaphor for capturing and integrating business-specific knowledge into the KG, thus enriching its representational power. Indeed, KG-Roar offers an interactive productivity environment where the user can select widgets of interest and augment the KG interactively at runtime with new knowledge thanks to the scalability and responsiveness of the underlying Vadalog reasoning engine. A *virtual* representation of the KG is provided in an interactive visualization window within the environment. The goal of KG-Roar is to enable users to perform complex analyses, seamlessly building, browsing, and querying even complex and large knowledge graphs, such as the European ownership KG of financial companies.

KGLM in KG-Roar. To support such a purpose, we enrich the KG-Roar environment with the KGLM framework presented in the previous section. Going beyond the visual representation of the derived edges in the KG, often unsatisfactory as missing the actual motivation for their existence (i.e., their provenance), we provide users with a *knowledge pal*

that enables natural language-based interactions with the underlying KG. Specifically, such an agent acts as a chatbot, standing upon Llama 2-70B models that are specialized in the domain of interest by leveraging KGLM's chase-based fine-tuning corpora. Depending on the nature of the user question, it performs distinct actions to provide the answer.

- In case of *descriptive queries*, i.e., questions that are focused on investigating the specifics of a certain fact in the KG, the knowledge pal leverages the expertise of the domain, acquired via fine-tuning, and the verbalized rules provided in a RAG-like mechanism to generate the answer. For instance, in the context of Example 1, to the question *"What is the notional of EGTech's position at time 1?"*, it responds with *"EGTech's notional at time 1 is 37.2$, equal to the product between 124$, the price of the asset at that time, and 0.3, the size of the opened position"*.
- In case of *explanatory queries*, i.e., more complex questions that are focused on investigating why a certain fact exists in the KG (that is, its provenance), the knowledge pal retrieves the full chase-based explanation of the fact from KGLM, further enhancing its readability and providing it to the user as an accurate business report. For instance, in the context of Example 1, to the question *"How does EGTech make profits from its trading activity?"*, it responds with *"By opening a position of size 0.3 at time 1, having the position accepted with an initial price of 124$ and a notional of 37.2$ at time 1, and then closing it at time 9 with a final price of 147$, thus achieving a profit of 6.9$"*.

Let us consider the application of the *company control* scenario from Example 2 and the *close link* one from Example 3 to showcase KGLM's integration into KG-Roar. As previously mentioned, KGLM supports the selection of the fact to explain from the corresponding edge in the knowledge graph. Thus, it is possible to click on an edge in the visualized KG to request the corresponding explanation, as illustrated in Figure 11. In the presence of highly connected KGs, such as the one in Figure 12, the visual exploration becomes less intuitive and the request can directly be prompted in NL to the knowledge pal. In both cases, the generated response will be presented in a dedicated box that can be interactively explored by highlighting tagged entities both in the text and in the visualized KG, and KG-Roar will zoom in on the portion of the graph involved.

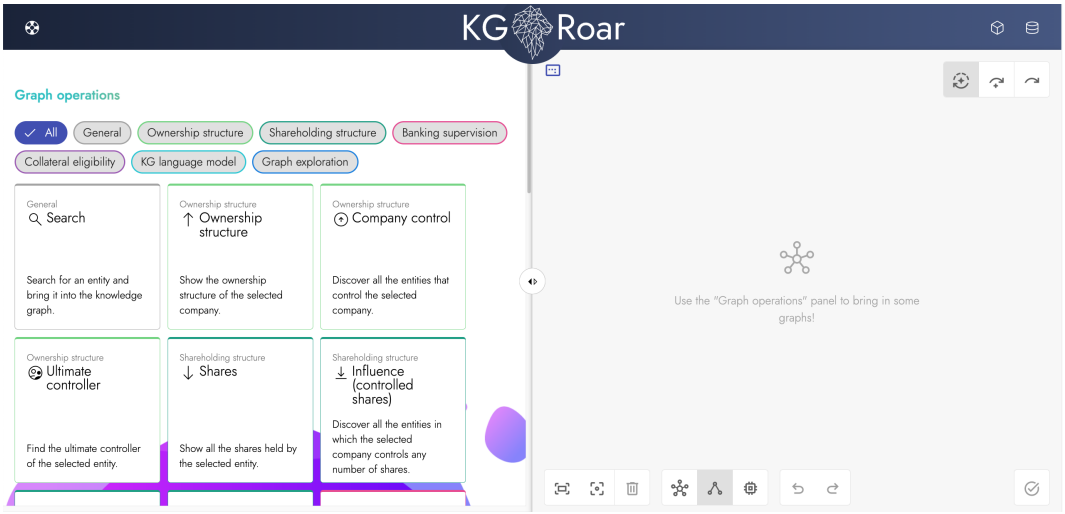

Figure 10 Financial use cases in the form of executable widgets.

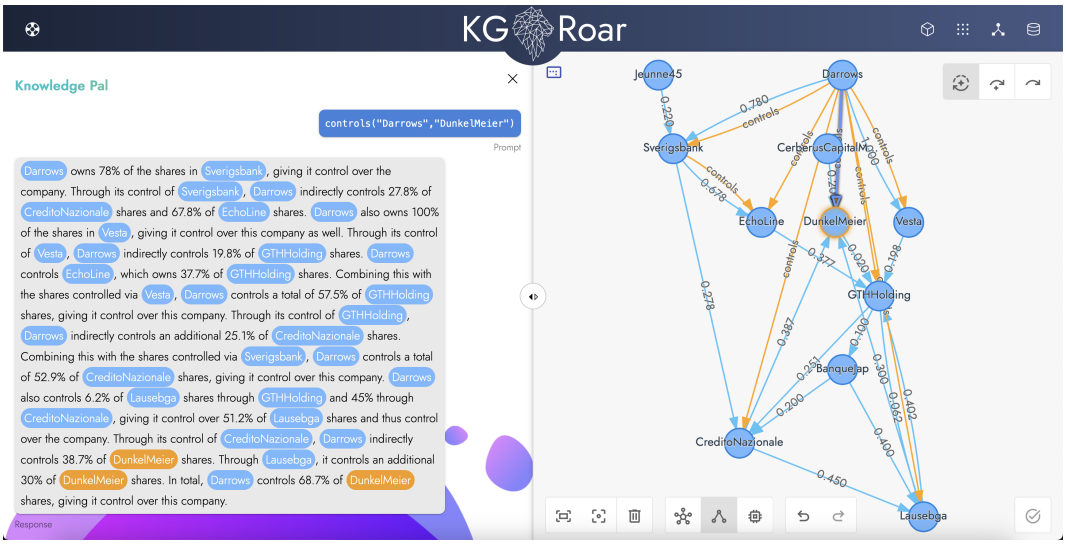

Figure 11 Instance of explanation panel for *company control* use case via direct graph selection.

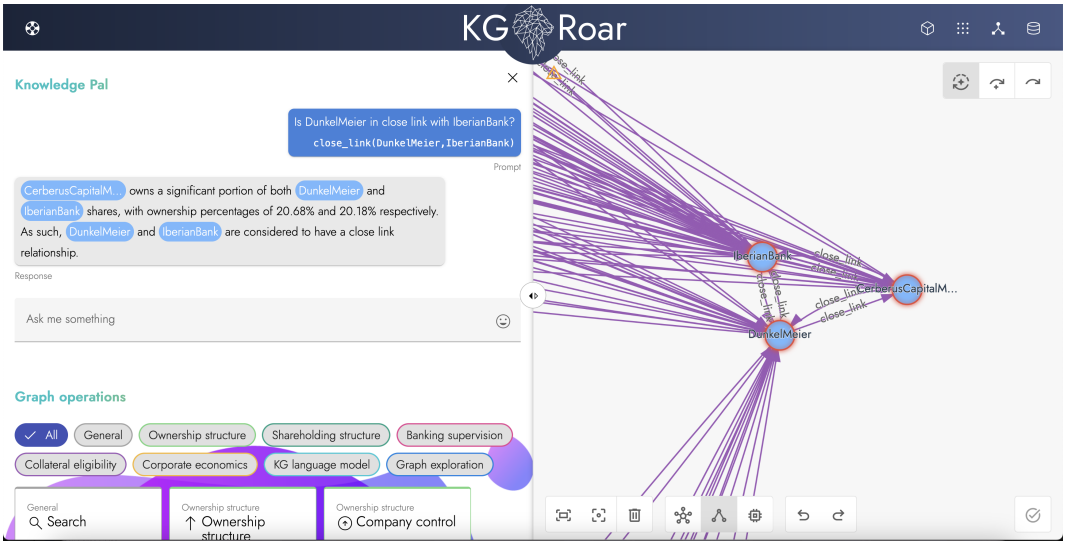

Figure 12 Instance of explanation panel for *close link* use case via prompt NL interaction.

5 Conclusion

In today's industrial landscape, we observe an increasingly pressing demand for AI applications to sustain transparency, fairness, and accountability of decision-making processes, especially in high-stakes domains like finance and bio-medicine. To achieve this, novel neuro-symbolic solutions are rising to bridge the gap between the opaque nature of ML-based technologies such as LLMs and the required human-interpretable provenance of their responses.

In this paper, we strived to contribute to such a paradigm shift by presenting KGLM, the first, to the best of our knowledge, neuro-symbolic pipeline that enhances LLMs with domain-specific knowledge and provenance-based explainability by leveraging the inherent domain-awareness of Knowledge Representation and Reasoning methodologies at the foundation of the Vadalog system. Capitalizing on our experience in the financial field, we built a

framework that can act as a natural language interface to Enterprise Knowledge Graphs, empowering business analysts to investigate the rationale behind data investigation performed via reasoning tasks in a human-oriented fashion. We also integrated our solution into a well-established KG environment for financial tasks, KG-Roar, which allows such users to conduct graph-based data analysis, now enhanced by the possibility of interacting with them in natural language. By further leveraging provenance information, we also aim to expand our KGLM framework in future works, injecting reasoning capabilities into Large Language Models in a chain-of-thought fashion to directly reason over EKGs, possibly enriched by additional unstructured data. We believe that such contributions could become significant assets for organizations, enabling them to harness the advanced capabilities of LLMs without sacrificing the clarity and accountability of informed decision-making processes.

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of databases*, volume 8. Addison-Wesley Reading, 1995.

2 Foto N. Afrati, Manolis Gergatsoulis, and Francesca Toni. Linearisability on datalog programs. *Theor. Comput. Sci.*, 308(1-3):199–226, 2003. doi:10.1016/S0304-3975(02)00730-2.

3 Paolo Atzeni, Luigi Bellomarini, Michela Iezzi, Emanuel Sallinger, and Adriano Vlad. Augmenting logic-based knowledge graphs: The case of company graphs. In *KR4L@ECAI*, volume 3020, pages 22–27. CEUR-WS.org, 2020.

4 Paolo Atzeni, Luigi Bellomarini, Michela Iezzi, Emanuel Sallinger, and Adriano Vlad. Weaving enterprise knowledge graphs: The case of company ownership graphs. In *EDBT*, pages 555–566. OpenProceedings.org, 2020.

5 Teodoro Baldazzi, Luigi Bellomarini, Stefano Ceri, Andrea Colombo, Andrea Gentili, and Emanuel Sallinger. Fine-tuning large enterprise language models via ontological reasoning. In *International Joint Conference on Rules and Reasoning*, pages 86–94. Springer, 2023.

6 Teodoro Baldazzi, Luigi Bellomarini, Stefano Ceri, Andrea Colombo, Andrea Gentili, and Emanuel Sallinger. "please, vadalog, tell me why": Interactive explanation of datalog-based reasoning. In Letizia Tanca, Qiong Luo, Giuseppe Polese, Loredana Caruccio, Xavier Oriol, and Donatella Firmani, editors, *Proceedings 27th International Conference on Extending Database Technology, EDBT 2024, Paestum, Italy, March 25 - March 28*, pages 834–837. OpenProceedings.org, 2024. doi:10.48786/EDBT.2024.82.

7 Teodoro Baldazzi, Luigi Bellomarini, Marco Favorito, and Emanuel Sallinger. Ontological reasoning over shy and warded datalog+/–for streaming-based architectures. In *International Symposium on Practical Aspects of Declarative Languages*, pages 169–185. Springer, 2024.

8 Teodoro Baldazzi, Luigi Bellomarini, Markus Gerschberger, Aditya Jami, Davide Magnanimi, Markus Nissl, Aleksandar Pavlović, and Emanuel Sallinger. Vadalog: Overview, extensions and business applications. *Reasoning Web. Causality, Explanations and Declarative Knowledge: 18th International Summer School 2022, Berlin, Germany, September 27–30, 2022, Tutorial Lectures*, pages 161–198, 2023.

9 Teodoro Baldazzi, Luigi Bellomarini, Emanuel Sallinger, and Paolo Atzeni. Eliminating harmful joins in warded datalog+/-. In *RuleML+RR*, volume 12851 of *Lecture Notes in Computer Science*, pages 267–275. Springer, 2021.

10 Teodoro Baldazzi, Davide Benedetto, Matteo Brandetti, Adriano Vlad, Luigi Bellomarini, and Emanuel Sallinger. Datalog-based reasoning with heuristics over knowledge graphs. In *Datalog*, volume 3203 of *CEUR Workshop Proceedings*, pages 114–126. CEUR-WS.org, 2022.

11 Pablo Barceló and Reinhard Pichler. *Datalog in Academia and Industry: Second International Workshop, Datalog 2.0, Vienna, Austria, September 11-13, 2012, Proceedings*, volume 7494. Springer, 2012.

12 Luigi Bellomarini, Lorenzo Bencivelli, Claudia Biancotti, Livia Blasi, Francesco Paolo Conteduca, Andrea Gentili, Rosario Laurendi, Davide Magnanimi, Michele Savini Zangrandi, Flavia

Tonelli, Stefano Ceri, Davide Benedetto, Markus Nissl, and Emanuel Sallinger. Reasoning on company takeovers: From tactic to strategy. *Data Knowl. Eng.*, 141:102073, 2022.

13 Luigi Bellomarini, Marco Benedetti, Andrea Gentili, Davide Magnanimi, and Emanuel Sallinger. Kg-roar: Interactive datalog-based reasoning on virtual knowledge graphs. *Proc. VLDB Endow.*, 16(12):4014–4017, August 2023.

14 Luigi Bellomarini, Davide Benedetto, Georg Gottlob, and Emanuel Sallinger. Vadalog: A modern architecture for automated reasoning with large knowledge graphs. *Inf. Syst.*, 105:101528, 2022. doi:10.1016/j.is.2020.101528.

15 Luigi Bellomarini, Daniele Fakhoury, Georg Gottlob, and Emanuel Sallinger. Knowledge graphs and enterprise AI: the promise of an enabling technology. In *ICDE*, pages 26–37, 2019.

16 Luigi Bellomarini, Emanuel Sallinger, and Georg Gottlob. The vadalog system: Datalog-based reasoning for knowledge graphs. *Proc. VLDB Endow.*, 11(9):975–987, May 2018. doi:10.14778/3213880.3213888.

17 Tom Brown and et al. Language models are few-shot learners. In *NeurIPS*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

18 Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. Why and where: A characterization of data provenance. In Jan Van den Bussche and Victor Vianu, editors, *Database Theory — ICDT 2001*, pages 316–330. Springer Berlin Heidelberg, 2001.

19 Pedro Cabalar, Jorge Fandinno, and Brais Muñiz. A system for explainable answer set programming. *Electronic Proceedings in Theoretical Computer Science*, 325:124–136, 2020. doi:10.4204/eptcs.325.19.

20 Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. *J. Web Semant.*, 14:57–83, 2012. doi:10.1016/j.websem.2012.03.001.

21 Andrea Calì, Georg Gottlob, Thomas Lukasiewicz, Bruno Marnette, and Andreas Pieris. Datalog+/-: A family of logical knowledge representation and query languages for new applications. In *2010 25th annual IEEE symposium on logic in computer science*, pages 228–242. IEEE, 2010.

22 Andrea Calì, Georg Gottlob, and Andreas Pieris. Towards more expressive ontology languages: The query answering problem. *Artificial Intelligence*, 193:87–128, 2012.

23 Steven P Callahan, Juliana Freire, Emanuele Santos, Carlos E Scheidegger, Cláudio T Silva, and Huy T Vo. Vistrails: visualization meets data management. In *ACM SIGMOD 2006*, pages 745–747, 2006.

24 Luciano Caroprese, Eugenio Vocaturo, and Ester Zumpano. Argumentation approaches for explanaible ai in medical informatics. *Intelligent Systems with Applications*, 16:200109, 2022. doi:10.1016/j.iswa.2022.200109.

25 Stefano Ceri, Georg Gottlob, Letizia Tanca, et al. What you always wanted to know about datalog(and never dared to ask). *IEEE transactions on knowledge and data engineering*, 1(1):146–166, 1989.

26 James Cheney, Laura Chiticariu, Wang-Chiew Tan, et al. Provenance in databases: Why, how, and where. *Foundations and Trends® in Databases*, 1(4):379–474, 2009.

27 Sarah Cohen-Boulakia, Olivier Biton, Shirley Cohen, and Susan Davidson. Addressing the provenance challenge using zoom. *Concurrency and Computation: Practice and Experience*, 20(5):497–506, 2008.

28 Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Comput. Surv.*, 33(3):374–425, September 2001. doi:10.1145/502807.502810.

29 Daniel Deutch, Nave Frost, and Amir Gilad. Provenance for natural language queries. *Proc. VLDB Endow.*, 10(5):577–588, January 2017. doi:10.14778/3055540.3055550.

30 Daniel Deutch, Nave Frost, and Amir Gilad. Natural language explanations for query results. *SIGMOD Rec.*, 47(1):42–49, September 2018. doi:10.1145/3277006.3277017.

31 Xin Luna Dong. Generations of knowledge graphs: The crazy ideas and the business impact. *arXiv preprint arXiv:2308.14217*, 2023.

32 Owen P. Dwyer, Teodoro Baldazzi, Jim Davies, Emanuel Sallinger, and Adriano Vlad. Reasoning over health records with vadalog: a rule-based approach to patient pathways. In Jan Vanthienen, Tomás Kliegr, Paul Fodor, Davide Lanti, Dörthe Arndt, Egor V. Kostylev, Theodoros Mitsikas, and Ahmet Soylu, editors, *Proceedings of the 17th International Rule Challenge and 7th Doctoral Consortium @ RuleML+RR 2023 co-located with 19th Reasoning Web Summer School (RW 2023) and 15th DecisionCAMP 2023 as part of Declarative AI 2023, Oslo, Norway, 18 - 20 September, 2023*, volume 3485 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2023. URL: `https://ceur-ws.org/Vol-3485/paper9111.pdf`.

33 Esra Erdem and Umut Oztok. Generating explanations for biomedical queries. *Theory and Practice of Logic Programming*, 15(1):35–78, 2015.

34 European Central Bank. Guideline (eu) 2011/14 of the ecb guideline, 2011.

35 Jorge Fandinno and Claudia Schulz. Answering the "why" in answer set programming–a survey of explanation approaches. *Theory and Practice of Logic Programming*, 19(2):114–203, 2019.

36 Boris Glavic, Renée J. Miller, and Gustavo Alonso. *Using SQL for Efficient Generation and Querying of Provenance Information*, pages 291–320. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. `doi:10.1007/978-3-642-41660-6_16`.

37 Goetz Graefe and William J. McKenna. The volcano optimizer generator: Extensibility and efficient search. In *ICDE*, pages 209–218. IEEE Computer Society, 1993.

38 Todd J Green, Shan Shan Huang, Boon Thau Loo, Wenchao Zhou, et al. Datalog and recursive query processing. *Foundations and Trends® in Databases*, 5(2):105–195, 2013.

39 Todd J Green, Grigoris Karvounarakis, and Val Tannen. Provenance semirings. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 31–40, 2007.

40 Todd J. Green and Val Tannen. The semiring framework for database provenance. In *PODS*, pages 93–99. ACM, 2017.

41 A. Gulino, S. Ceri, G. Gottlob, E. Sallinger, and L. Bellomarini. Distributed company control in company shareholding graphs. In *IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2637–2648, Los Alamitos, CA, USA, 2021.

42 Kyle Hamilton, Aparna Nayak, Bojan Bozic, and Luca Longo. Is neuro-symbolic AI meeting its promise in natural language processing? A structured review. *CoRR*, abs/2202.12205, 2022. `arXiv:2202.12205`.

43 Melanie Herschel and Marcel Hlawatsch. Provenance: On and behind the screens. In *SIGMOD 2016*, pages 2213–2217, New York, NY, USA, 2016. Association for Computing Machinery. `doi:10.1145/2882903.2912568`.

44 Jason I. Hong. Teaching the fate community about privacy. *Commun. ACM*, 66(8):10–11, July 2023.

45 David S. Johnson and Anthony C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. Comput. Syst. Sci.*, 28(1):167–189, 1984.

46 Dominik K Kanbach, Louisa Heiduk, Georg Blueher, Maximilian Schreiter, and Alexander Lahmann. The genai is out of the bottle: generative artificial intelligence from a business model innovation perspective. *Review of Managerial Science*, pages 1–32, 2023.

47 David Koop, Marta Mattoso, and Juliana Freire. *Provenance in Workflows*, pages 2912–2916. Springer New York, New York, NY, 2018. `doi:10.1007/978-1-4614-8265-9_80745`.

48 Markus Krötzsch and Veronika Thost. Ontologies for knowledge graphs: Breaking the rules. In *International Semantic Web Conference*, pages 376–392. Springer, 2016.

49 Seokki Lee, Bertram Ludäscher, and Boris Glavic. Provenance summaries for answers and non-answers. *Proc. VLDB Endow.*, 11(12):1954–1957, August 2018. `doi:10.14778/3229863.3236233`.

50 Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

51 Zhuang Liu, Degen Huang, Kaiyu Huang, Zhuang Li, and Jun Zhao. Finbert: A pre-trained financial language representation model for financial text mining. In *IJCAI 2020*, 2021.

52 Davide Magnanimi and Michela Iezzi. Ownership graphs and reasoning in corporate economics. In *EDBT/ICDT Workshops*, volume 3135 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2022.

53 David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv. Testing implications of data dependencies. *ACM TODS*, 4(4):455–469, 1979. doi:10.1145/320107.320115.

54 Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying large language models and knowledge graphs: A roadmap. *arXiv preprint arXiv:2306.08302*, 2023.

55 Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. ., 2018.

56 Yann Ramusat, Silviu Maniu, and Pierre Senellart. Semiring provenance over graph databases. In *Proceedings of the 10th USENIX Conference on Theory and Practice of Provenance*, TaPP'18, page 7, USA, 2018. USENIX Association.

57 Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In *EMNLP (1)*, pages 5418–5426. ACL, 2020.

58 Sudeepa Roy and Dan Suciu. A formal approach to finding explanations for database queries. In *SIGMOD 2014*, pages 1579–1590, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2588555.2588578.

59 Pierre Senellart. *Provenance in Databases: Principles And Applications*, pages 104–109. Springer-Verlag, Berlin, Heidelberg, 2022. doi:10.1007/978-3-030-31423-1_3.

60 Yogesh L Simmhan, Beth Plale, and Dennis Gannon. Karma2: Provenance management for data-driven workflows. *International Journal of Web Services Research (IJWSR)*, 5(2):1–22, 2008.

61 Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

62 Adriano Vlad, Sahar Vahdati, Mojtaba Nayyeri, Luigi Bellomarini, and Emanuel Sallinger. Towards hybrid logic-based and embedding-based reasoning on financial knowledge graphs. In *EDBT/ICDT Workshops*, volume 3135, 2022.

63 Yisu Remy Wang, Mahmoud Abo Khamis, Hung Q. Ngo, Reinhard Pichler, and Dan Suciu. Optimizing recursive queries with progam synthesis. In *SIGMOD 2022*, pages 79–93, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3514221.3517827.

64 Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David S. Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. *CoRR*, abs/2303.17564, 2023.

65 Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020. URL: https://openreview.net/forum?id=SkeHuCVFDr.

66 Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. Explainability for large language models: A survey. *arXiv preprint arXiv:2309.01029*, 2023.