Template-based Explainable Inference over High-Stakes Financial Knowledge Graphs

Andrea Colombo
Politecnico di Milano
andrea1.colombo@polimi.it

Teodoro Baldazzi
Università Roma Tre
teodoro.baldazzi@uniroma3.it

Luigi Bellomarini*
Banca d'Italia
luigi.bellomarini@bancaditalia.it

Emanuel Sallinger
TU Wien & University of Oxford
sallinger@dbai.tuwien.ac.at

Stefano Ceri
Politecnico di Milano
stefano.ceri@polimi.it

ABSTRACT

Declarative query languages based on logic programming, like Datalog and its extensions, have recently found successful applications in modeling complex knowledge-based scenarios, such as reasoning over Enterprise Knowledge Graphs (EKG), by encoding business rules to derive new valuable knowledge. Presenting this derived knowledge with comprehensible natural language explanations is paramount to increasing transparency, accountability, and fairness in AI-based systems. While Large Language Models (LLMs) offer promising directions, full industrial adoption in critical settings requires a trustworthy solution that ensures both accurate, clear explanations and compliance with strict data protection standards (i.e., by not sharing data with third parties).

This work introduces a novel approach for the generation of textual explanations from data-driven inference processes where data protection is crucial, such as in sensitive financial applications governed by deductive rules encoded by the Central Bank of Italy. We propose a static structural analysis method that identifies a finite set of reasoning patterns from business rules, which are then used to generate fluent natural language explanations. By capturing the main interconnections between rules, our approach generates explanations comparable in quality to those produced by LLMs, but without requiring data sharing through external APIs or cloud servers, thus ensuring data protection in high-stakes, sensitive applications. Furthermore, our method guarantees that explanations are both correct and complete, unlike LLM-generated ones, which may suffer from critical omissions.

1 INTRODUCTION

With the growing importance of data in today's industrial environment, applications that leverage artificial intelligence (AI) for decision-making are increasingly focused on adhering to the FATE principles: Fairness, Accountability, Transparency, and Ethics [41]. This is especially crucial in high-stakes domains such as financial supervision, where ensuring the FATE compliance of these knowledge-driven processes is paramount for building trust in critical decisions [31, 34, 45].

In the database community, the quest for explainability has been framed as *explaining instances* or *facts* contained in the result of a query $Q(D)$. This involves augmenting the query results with information about the origin of the facts that satisfy Q, both from the original tuples in the database D and from the tuples generated by applying a set of logical rules Σ that encode the business domain. In this context, deductive AI approaches built on top of declarative *Knowledge Representation and Reasoning* (KRR) formalisms are experiencing rising interest, as they enable FATEness by providing a full *explanation* of conclusions. This approach is both intuitively and practically close to the top-down logical inference methods typically adopted in KRR [21, 22].

In the realm of KRR, Enterprise Knowledge Graphs (EKGs) serve as powerful models to express business rules, i.e., the *intensional* component, which expands factual knowledge, i.e., the *extensional* component. The extensional component corresponds to a database of facts, structured as a graph where nodes represent entities and edges denote relationships between these entities. The intensional component provides a formal specification of business experience, through the definition of inference rules that operate on the extensional component. By applying these rules, i.e., performing a *reasoning task*, new knowledge emerges as novel nodes and edges within the EKG. Such rules can be naturally expressed in database query languages based on logic programming, such as Datalog [1, 7, 19, 59] and its extensions [3, 15–18, 20, 33], a yardstick for AI systems based on ontological reasoning thanks to the good trade-off they offer between expressive power and computational complexity [7, 24, 47].

Problem Statement. From a practical perspective, there is a growing need for natural language (NL) explanations of the knowledge inferred through reasoning tasks. Given an inferred fact from a reasoning task run over an extensional EKG, the interest of a general user is to understand how such knowledge was derived [28]. This can be achieved by examining its proof, i.e., sets of logical steps that allow deriving conclusions from premises (Figure 1). However, formal proofs are hardly readable for non-technical users, and there is a need to convert such proofs into natural language to improve the transparency of automated reasoning tools. While deterministic proof-to-NL conversion provides some clarity, it can result in explanations that are verbose, repetitive, and difficult to interpret. In fact, an optimal textual explanation should be *comprehensible*, making the reasoning clear without delving into the technical details of inference; *fluent*, capable of explaining even complex reasoning scenarios, e.g., of unpredictable length or involving algebraic operations; *compact and accurate*, recognizing recurring reasoning patterns to be conveyed in general terms and adhering to the inference rules.

Natural Language Explanations Approaches. Proposals for *natural language provenance models* that convert proofs to text have been made in the past (e.g., [30]). However, with the advent of Large Language Models (LLMs), this task has opened new

*The views and opinions expressed in this paper are those of the authors and do not necessarily reflect the official policy or position of the Bank of Italy.

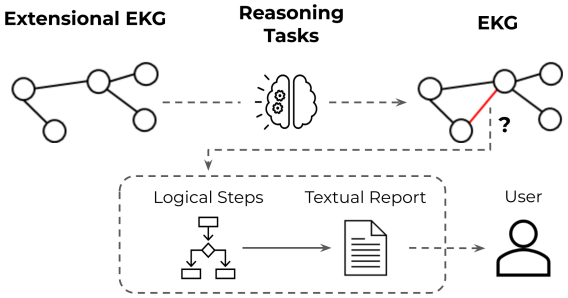


Figure 1: In EKGs, knowledge can be inferred by applying reasoning tasks, i.e., rule-based business applications that derive edges or nodes by performing logical steps. Converting them in a fluent textual form is essential for improving transparency in the inference mechanisms

possibilities, as they can be used to generate readable and comprehensible text, as we have recently demonstrated [4, 5]. The general approach to obtain a textual explanation is to generate a full materialization of the instance and trivially transform the provenance into natural language, leveraging the *data dictionary* or *domain glossary* containing a description of terms used in the reasoning task. This results in a *deterministic explanation* of inferred facts. However, explanations end up being too long and verbose, as they include all the inference steps, described one by one. On top of this one could then take advantage of LLMs and ask to perform its *summary* or *paraphrasis*, generating high-quality and readable explanations. Still, this approach has limitations, as both prompts may lead to omissions or, in some rare cases, even *hallucinations*.

LLMs and Data Privacy. Exploiting LLMs' potential in critical settings comes with the significant challenge of protecting confidential information [42]. Breaching data confidentiality refers to the inevitable disclosure of information that occurs when using most modern LLMs, whether for training or inference, as these require users to call LLMs via APIs or cloud-based server providers that offer the computational power to host and run such enormous generative AI models. In this context, anonymization techniques are a practical solution for preserving data confidentiality [50]. However, much of the data anonymization literature has focused on structured data, such as tables, while anonymization of unstructured data, such as textual explanations, remains unclear and challenging [62]. Therefore, developing a valid alternative that achieves the same level of textual quality as a pure LLM-based solution while complying with the highest data confidentiality standards raises a novel challenge.

Financial Enterprise Knowledge Graph. A relevant industrial case where the need for confidentiality and transparency is critical is the Enterprise Knowledge Graph developed by the Central Bank of Italy, which models relationships between financial intermediaries, empowering analysts to conduct complex studies about interesting financial patterns. Such applications are based on well-established business rules, originally laid out in official regulations, which have been encoded into a powerful and expressive logic-based declarative query language, i.e. Vadalog. Part of the Datalog family of languages, Vadalog has been successfully used for encoding multiple financial problems, for which presenting textual explanations, i.e., natural language business reports, while not disclosing the underlying data and information to third parties, would greatly increase transparency and accountability in internal decision-making processes.

Rule-based Knowledge Graph Applications. A KG application represents a relevant and recurring reasoning task that business experts have formalized in a logic language, such as the Bank of Italy's EKG. It includes long-standing problems such as the derivation of control relationships, i.e., who controls whom, or the modeling of shock propagation systems to conduct stress test simulations. Such tasks might involve complex and long reasoning processes that result from the incremental augmentation of the initial factual knowledge via the application of the reasoning rules, until fixpoint. The complexity of a KG application becomes particularly significant when *aggregation* operators and *recursion* are involved. Their presence might generate non-trivial *proofs*, whose length, due to recursion, is unpredictable.

The key idea of this paper is to exploit the rich expressiveness of logical inference processes adopted in deductive AI and the power of modern generative AI tools to build fluent and complete natural language explanations for Knowledge Graph applications, all while maintaining a privacy-preserving approach.

Template-based approach. To achieve our goal, we introduce **reasoning paths**, i.e., sets of Datalog rules that can capture the main interconnections between predicates and rules and that, if transformed into textual *explanation templates*, can be used to generate explanations that are comprehensible, fluent, and compact. We start from a preventive structural analysis of the *dependency graph* of a deployed KG application. By splitting it based on the syntactical features of the rules, we identify the main reasoning stories, which can then be verbalized to create explanation templates, also by employing LLMs to fully automatize the process. The explanation templates contain tokens that can be mapped back to the rules' literals: given a derived fact of interest, an explanation can be produced by combining one or more explanation templates according to the actual reasoning path followed during inference and by replacing the tokens with the materialized literals. The resulting explanations contain all relevant information required by analysts, with an option to include a human-in-the-loop step for preventive checks.

By pushing the LLM usage to the reasoning rules instead of the materialized instance, we propose a system that can be queried to generate textual explanations for KG applications that are as fluent and compact as those produced by pure LLM-based approaches, as confirmed by our expert user study, while still guaranteeing the highest standards of data confidentiality.

Our **contributions** can be summarized as follows.

- We present a novel approach based on **explanation templates** to create natural language reports for knowledge derived by rule-based Knowledge Graph applications.
- We implement and discuss our approach to financial applications developed in the EKG of the Bank of Italy.
- We perform an **experimental evaluation** via user studies to validate the effectiveness and accuracy of our approach in producing fluent explanations; we highlight our benefit compared to LLM-based solutions, and we illustrate the performances of our approach.

Overview. In the remainder of the paper is organized as follows. In Section 2 we discuss related work and Section 3 provides fundamental background. In Section 4, we present our approach to derive reasoning paths and generate fluent natural language

explanations, while in Section 5, we present some high-stakes applications. In Section 6, we illustrate the results of two user studies, which validate the effectiveness of our approach and the quality of the explanations. Our conclusions are drawn in Section 7.

2 RELATED WORK

In this section, we expand on relevant work dedicated to natural language explanations in logical inference processes, i.e., NL provenance models.

Developing a suitable and user-comprehensible form of data provenance has emerged as a critical research area in the database community, aiming to address the growing concerns regarding data quality, trustworthiness, and accountability. Numerous studies have been conducted to investigate different aspects of data provenance, namely tracking, storing, and presentation of the provenance [13, 23, 36, 40, 46, 54].

A recent line of research is studying how to present answers to queries in natural language [32, 48]. Some attempts have created natural language explanations by constructing them rule by rule [14, 39], however, showing no coherent prose and, thus, are hardly readable for business analysts. PROV [51] and its extension PROVglish [53] were the first architectures that have tried to transform provenance graphs into textual explanations. Their approach collects the necessary linguistic information informally encoded in the URIs, developing a deterministic solution based on part-of-speech (POS) tagging. The idea is that it is possible to build text generation rules using at least the classes of tags. While this resulted in richer and more fluent explanations compared to non-POS approaches, such a system is far from what could be considered a well-written business report, considering also that PROV is only capable of generating single-sentence explanations and is limited to short provenance graphs.

With NLProv [28–30], provenance information in natural language is built by leveraging users' NL question's inherent structure. They track specific phrases or words in the NL question and map them to corresponding parts of the formal query, recording which element of the formal query contributes to a specific part of the provenance information. By combining these mappings (text-to-query-parts and query-parts-to-provenance), they establish links between question phrases and relevant provenance details and utilize this association in reverse, translating the provenance information back into readable NL text. While being an elegant solution, the quality of resulting texts depends on input natural language questions, thus not applicable to contexts where textual explanations need to be generated over front-end applications where analysts directly interact with data as, for instance, in a graph-based environment, such as the one in place in the Bank of Italy [10].

More recently, we have demonstrated how to employ a pure LLM-based solution to obtain explanations of reasoning conducted over KG applications [4]. However, practically deploying that solution raised the issue of data confidentiality compliance. While using the most powerful generative AI tool to date, OpenAI's GPT family of models, meant sharing information with a third party, adopting an internal solution based on an open-source LLM, such as LLama, resulted in variable quality in the results, different in each run.

3 PRELIMINARIES

To guide our discussion, we first lay out the preliminary notions.

Relational Foundations. Let $\mathbf{C}$, $\mathbf{V}$ and $\mathbf{N}$ be disjoint countably infinite sets of *constants*, *variables* and *nulls*, 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, a *homomorphism* from ς_1 to ς_2 is a mapping $h : \mathbf{C} \cup \mathbf{N} \cup \mathbf{V} \rightarrow \mathbf{C} \cup \mathbf{N} \cup \mathbf{V}$ s.t. $h(t) = t$ if $t \in \mathbf{C}$, $h(t) \in \mathbf{C} \cup \mathbf{N}$ if $t \in \mathbf{N}$ and for each atom $a(t_1, \ldots, t_n) \in \varsigma_1$, then $h(a(t_1, \ldots, t_n)) = a(h(t_1), \ldots, h(t_n)) \in \varsigma_2$.

Dependencies. A Vadalog program Σ 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, otherwise, it is *extensional* (EDB) [1, 22, 38]. A fact corresponding to an intensional predicate is intensional, otherwise it is extensional.

Dependency Graph. The *dependency graph* of a Vadalog program Σ, denoted as $\mathcal{D}(\Sigma)$, is a directed graph where the set of vertices is the set of predicates appearing in a given set of rules Σ [37]. For each not necessarily distinct pair of predicates a and a' in Σ, there is an edge from a' to a iff Σ contains a rule where a' appears in the body and a appears in the head. A program Σ is *recursive* if the dependency graph $\mathcal{D}(\Sigma)$ is cyclic. A node a *depends on* a node a', denoted $a' \preceq a$, if there is a path from a' to a in $\mathcal{D}(\Sigma)$.

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* [61]. 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, modeled with comparison ($>, <, \geq, \leq, \neq$) and algebraic ($+, -, *, /$, etc.) operators.

Reasoning Task. 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 and Chase Graph. The semantics of a Vadalog program can be defined in an operational way with the *chase procedure* [44, 49]. 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. 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 graph with the facts from $chase(D, \Sigma)$ as

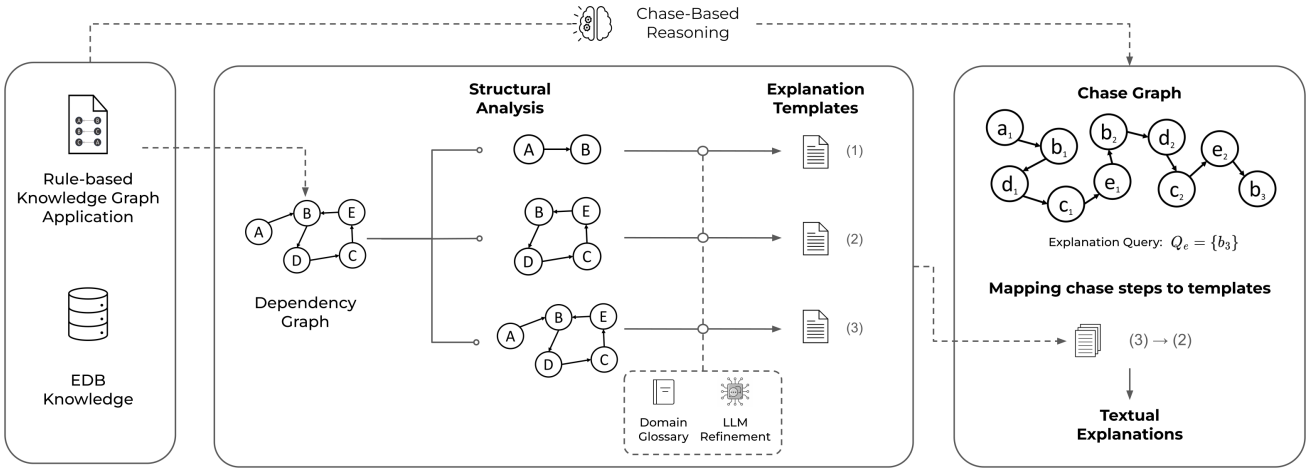


Figure 2: Overview of the proposed approach to produce textual explanations of facts. By leveraging rule-based encoding of knowledge graph applications, we get fluent templates that can be used to explain any fact derived by the chase procedure, without exposing factual knowledge to a third party.

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 [17]. In this paper, we only consider Vadalog programs involved in reasoning tasks whose termination is guaranteed. A comprehensive examination of reasoning termination in Vadalog has already been examined in dedicated works [6, 11].

4 TEMPLATE-BASED EXPLANATIONS

In this section, we present our approach to create easy-to-read template-based textual explanations for Knowledge Graph applications encoded in set logic-based rules Σ. Our approach is based on the Vadalog language and on a reasoning engine based on the chase procedure, as defined in the previous section. Without loss of generality, it can be generalized to Datalog programs.

Approach overview. Figure 2 summarizes our approach's steps towards deriving natural language explanations of facts entailed by Σ. We first pre-distill all the possible database-independent "reasoning stories", i.e., the full-blown theoretical of consequential chase steps in a compact way. This can be done via a *structural analysis* of the dependency graph of a Vadalog rule-based program, encoding business rules used to reason over a Knowledge Graph, to split it into *directed subgraphs*, according to the topology of $\mathcal{D}(\Sigma)$. Then, each of the subgraphs is transformed into natural language leveraging the syntax and grammar of the logic rules and the information contained in a *domain glossary*, producing *explanation templates*, which can be enhanced by leveraging powerful LLMs, such as GPT-based models. Thus, the portion of the chase graph representing the derivation of a certain fact can be mapped back to explanation templates via selection and, possibly, a combination of one or more explanation templates.

4.1 Structural Analysis

For any reasoning task as defined in Section 3, the chase procedure incorporates two forms of non-determinism in the chase step: the choice of which applicable rule to activate and, for such rule, which homomorphism θ to apply. Nevertheless, any potential chase step depends on the dependency graph, which contains all the necessary information to track any reasoning stories that might materialize. To produce coherent and fluent explanations, the challenge is to capture the main interconnections that characterize a dependency graph, which, if cyclic and displays aggregations, has an indefinite number of potential paths. To this end, we introduce *reasoning paths*, sequences of symbolic logical steps that generalize any root-to-leaf path in the chase graph (see Section 3), that can be used to generate templates. Roots in the dependency graph are nodes that do not depend on other nodes and appear in rules whose bodies do not contain intensional predicates. The leaf is a node denoting the intensional of interest, i.e., the goal of a Vadalog program. *Reasoning paths* can be either *simple reasoning paths* or *reasoning cycles*.

Definition 4.1. A node V of $\mathcal{D}(\Sigma)$ is critical when V is not extensional and either $deg^+(V) > 1$ or V is a leaf node.

Definition 4.2. A simple reasoning path $\Pi(\mathcal{D}(\Sigma))$ is a subgraph of $\mathcal{D}(\Sigma)$ that from roots conduct either to the leaf or to a critical node. A reasoning cycle $\Gamma(\mathcal{D}(\Sigma))$ is a subgraph of $\mathcal{D}(\Sigma)$ that connects a critical node with itself or with another critical node.

Both simple reasoning paths and reasoning cycles are computed by allowing only one visit per edge. Therefore, the reasoning paths are, by construction, finite. For a more compact notation, reasoning paths can also be represented as sequences of rules σ_i, i.e., $\Pi_i = \{\sigma_j, \ldots, \sigma_n\}$, by extracting labels of edges involved in the path.

Example 4.3. Let us consider the following set of rules, encoding a simplified version of a stress test simulation, one of the financial Knowledge Graph applications that allows analysts to perform so-called shock propagation exercises, simulating the effect of a shock over the financial market by the derivation of *Default* events, i.e., organizations that do not pass the stress test.

$$Shock(f, s), HasCapital(f, p_1), s > p_1 \rightarrow Default(f) \quad (\alpha)$$
$$Default(d), Debts(d, c, v), e = sum(v) \rightarrow Risk(c, e) \quad (\beta)$$
$$HasCapital(c, p_2), Risk(c, e), p_2 < e \rightarrow Default(c) \quad (\gamma)$$

A financial institution f defaults when an exogenous shock of s euro deteriorated its capital p_1 to the point it becomes negative (rule α). Whenever a financial institution defaults, it impacts its creditors via its exposures. The financial institutions c are at risk of failure, featuring a total e of loan exposures (rule β), measured in euro, to the defaulted entities. If the total e of exposures for the creditor c is higher than its capital p_2, then the creditor c defaults (rule γ). The

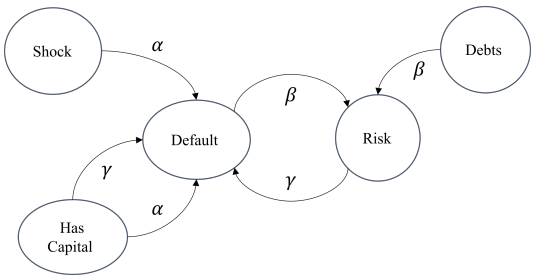
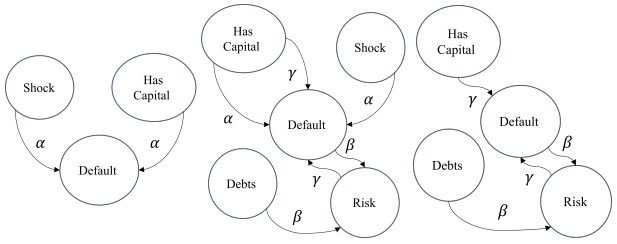


Figure 3: Dependency Graph of the set of rules from Example 4.3. Nodes represent all the atoms appearing in the set of rules, while rule-labeled edges connect the atoms in the body of the rule with the respective head. The leaf node is *Default* while root nodes are *Shock* and *HasCapital*. The dependency graph contains a critical node, i.e., the leaf node *Default* itself.



(a) Simple Reasoning Paths **(b) Reasoning Cycle**

Figure 4: Reasoning Paths derived by the structural analysis of Example 4.3. Simple Reasoning Paths, Π_1 and Π_2, represent paths that derive, respectively, defaults directly caused by an initial shock and direct defaults caused by exposures to a defaulted debtor. The Reasoning Cycle Γ_1 captures the case of a *cascade* default, indirectly caused by a chain of defaults.

set of rules aims to derive all the defaults, i.e., Default facts that an initial shock affecting one or more entities might trigger.

The dependency graph of the above set of rules is depicted in Figure 3. As it includes recursion, it is cyclic. In Figure 4 we derived the simple reasoning paths and the reasoning cycles from the dependency graph by applying their definitions.

Every path in the chase graph can be represented as an instantiation of a simple reasoning path and a set of adjacent reasoning cycles. To understand this, we start by considering a simple reasoning path Π, in its compact rule-based notation, from σ_s to σ_t, and a reasoning cycle Γ from σ'_s to σ'_t. They are *adjacent* if there is a homomorphism from the head of σ_t to a body atom of σ'_s. Intuitively, the two reasoning paths can be merged into a longer path from σ_s to σ'_t. Then, by considering a second reasoning cycle, Γ', from σ''_s to σ''_t, it is again *adjacent* to Γ if there is a homomorphism from the head of σ'_t to a body atom of σ''_s, and so on. Given a chase graph $\mathcal{G}(D, \Sigma)$, for every materialized source-to-leaf path π of $\mathcal{G}$, there exists a set of adjacent reasoning paths $\Pi^i, \Gamma^j \ldots, \Gamma^n$ that instantiates π. For any rule-based knowledge graph application, we call a generic explanation of a fact a *reasoning graph* on the chase graph.

Analysis of Aggregations. So far, we generated reasoning paths by limiting the structural analysis to the topology of the dependency graph. However, aggregations, namely, the *sum* and *prod* operators, play a central role and are widely used in real-world

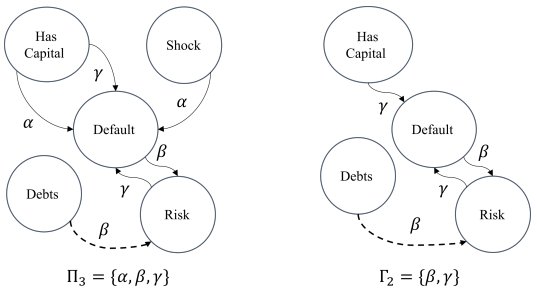


Figure 5: Additional Simple Reasoning Path and Reasoning Cycle for Example 4.3, capturing aggregation cases, i.e., when multiple *Debts* contribute to deriving the total *Risk* that an organization is exposed to.

KG applications. Textual explanations differ when an aggregation involves a single contributor to situations where multiple contributors are involved. Essentially, if there's only one contributor, it's similar to having - and can be explained as - a rule without the aggregation operator. To this aim, we expand the set of reasoning paths by analyzing edges, i.e., evaluating if rules contain an aggregation. If so, new reasoning paths, either simple reasoning paths or reasoning cycles, are added to the set. The new reasoning paths inherit the same topology as the source one, but we denote them by connecting contributors to an intensional with a dashed edge. Source reasoning paths will be used to capture reasoning stories whose aggregation is computed on one input fact, while additional ones will denote cases when the aggregator operates on multiple input facts.

Example 4.4. Continuing Example 4.3, two additional simple reasoning paths and two additional reasoning cycles can be identified, as depicted in Figure 5.

4.2 Explanation Templates

Reasoning paths can be transformed into *explanation templates* by applying a deterministic transformation of the Vadalog syntax into text. This is possible by means of a verbalizer, which algorithmically translates each rule into a natural language sentence of the form *"Since {body}, then {head}"*. We already used this module for producing on-the-fly textual explanations in [4] of an actual instance. In this paper, we apply the verbalizer directly to the rules that make up a reasoning path. We recall its main functioning. Each element of the Vadalog syntax is converted into its natural language counterpart. For example, *"and"* tokens are used for conjunctions, and specific keywords like *"is higher than"* replace the > built-in operator or aggregators such as *"sum"* is replaced by the sentence *"<result> is given by the sum of <contributors>"*. {Body} and {head} are obtained by verbalizing each predicate of the rules of the reasoning path. The textual verbalization contains *tokens* that can be directly mapped to the predicates' literals.

To this end, a domain glossary, a map of the predicates of our domain schema S into their natural language equivalent, is used. The domain glossary is essentially a *data dictionary* for Datalog-based contexts, i.e., a centralized repository of information about data - namely, *metadata* - such as meaning, relationships to other data, origin, usage, and format, an essential tool for any data-driven organization [52, 57]. Therefore, we suppose that, in any industrial context, a data dictionary is already stored and available.

Reasoning Path	Explanation Templates	Enhanced Templates
Π_1	*Since a shock amounting to <s> euros affects <f> and <f> is a financial institution with capital <p> and <s> is higher than <p> , then <f> is in default.*	*Since <f> has a capital of <p_1> and is hit by a shock of <s> euro, <f> is in default.*
Π_2	*Since a shock amounting to <s> euro affects <f>, and <f> has a capital of <p_1> and <f> is higher than <p_1> then <f> is in default. Since <d> is in default, and <d> has <v> euros of debts with <c>, then <c> is at risk of defaulting given its loan of <e> euros to a defaulted debtor. Since <c> is at risk of defaulting given its loan of <e> euros to a defaulted debtor and <c> is a financial institution with capital <p_2>, and <p_2> is lower than <e>, then <c> is in default.*	*<f> is in default due to a shock of <s> euros, being over its capital of <p_1> euros. With <e> debts to <d>, <c> is at risk due to having <v> euros of exposures to a defaulted debtor. <c> has and a capital of <p_2>, lower than <e>, thus also being in default.*
Π_3	*Since a shock amounting to <s> euro affects <f>, and <f> has a capital of <p_1> and <f> is higher than <p_1> then <f> is in default. Since <d> is in default, and <d> has <v> euros of debts with <c>, then <c> is at risk of defaulting given its loans of <e> euros to a defaulted debtor, with <e> given by the sum of <v>. Since <c> is at risk of defaulting given its loan of <e> euros to a defaulted debtor and <c> is a financial institution with capital <p_2>, and <p_2> is lower than <e>, then <c> is in default*	*Given that <s> is higher than its capital <p_1>, <f> defaults. Thus, <c> is at risk of defaulting since it lent <e> euros to a defaulted debtor, where <e> is the sum of loans of <v> euros. With a capital of <p_2>, lower than <e>, <c> defaults as well.*
Γ_1	*Since <d> is in default, and <d> has <v> euros of debts with <c>, then <c> is at risk of defaulting given its loan of <e> euros to a defaulted debtor. Since <c> is at risk of defaulting given its loan of <e> euros to a defaulted debtor and <c> is a financial institution with capital <p_2>, and <p_2> is lower than <e>, then <c> is in default.*	*Debtor <d> defaults on <v> euro debt to creditor <c>. Since <c> loaned <e> euros (more than <c>'s capital <p_2>) to defaulted <d>, <c> itself faces default.*
Γ_2	*Since <d> is in default, and <d> has <v> euros of debts with <c>, then <c> is at risk of defaulting given its loans of <e> euros to a defaulted debtor, with <e> given by the sum of <v>. Since <c> is at risk of defaulting given its loan of <e> euros to a defaulted debtor and <c> is a financial institution with capital <p_2>, and <p_2> is lower than <e>, then <c> is in default*	*Defaulted debtor <d> leaves lender <c> exposed with a total of <e> euros loan (sum of loans of <v> euros). This puts <c>, a financial institution with capital <p_2>, in default as well due to insufficient reserves.*

Figure 6: Explanation templates and their enhanced versions for the reasoning paths presented in Figures 4 and 5. Tokens are represented inside angle brackets. Explanation templates are obtained by deterministically verbalizing, through the domain glossary, rules involved in the reasoning path. Enhanced templates are obtained by applying LLMs to explanation templates.

Atom	Description
$HasCapital(f, p)$	<f> is a financial institution with capital of <p>.
$Shock(f, s)$	A shock amounting to <s> euro affects <f>.
$Default(f)$	<f> is in default.
$Debts(d, c, v)$	<d> has an amount <v> of debts with <c>.
$Risk(c, e)$	<c> is at risk of defaulting given its loan of <e> euros of exposures to a defaulted debtor.

Figure 7: Domain Glossary for Example 4.3, capturing the meaning of all atoms used in the Knowledge Graph application under analysis. Their description is provided by domain experts in the internal data dictionary.

Example 4.5. In Figure 7 we report the domain glossary for the stress test case described in Example 4.3.

By leveraging the domain glossary, explanation templates can be obtained by deterministically verbalizing the rules of interest, maintaining the predicates' variables as tokens in the text. As mentioned in the previous section, special attention is dedicated to aggregations, i.e., reasoning paths denoted by dashed edges. Loosely speaking, during the template generation process, the aggregator's transformation to natural language is truncated for reasoning paths not containing dashed edges by not verbalizing the aggregator. Instead, in the verbalization of dashed-denoted reasoning paths, the aggregator is converted to natural language, and corresponding tokens are allowed to accommodate an undefined number of contributors with a step of textual conjunction.

Enhancement of templates. The resulting deterministic explanation templates contain many repetitions, making the text redundant and not fluent. Moreover, they do not consider the interaction between rules in the same reasoning path. To cope with this, a human can intervene to generate enhanced versions of the templates. However, as the number of templates can grow exponentially with the complexity of the Vadalog program, and following recent works that demonstrated how powerful LLMs excel in text manipulation techniques, such as summarizing and paraphrasing tasks [43, 56], we can instead add a step on enhancement via LLMs, such as via a *gpt-3.5-turbo* model [12], which displayed very good results in text simplification tasks, even in specialized fields such as bio-medicine [55]. We prompt the LLM with the following request: *"Rephrase the following text:"* and add one of the explanation templates. The output of this step is a set of text-enhanced explanation templates, which are automatically double-checked in terms of the presence of all original tokens. To increase the textual richness of final explanations, this step can be repeated multiple times, generating different but interchangeable enriched versions of the same explanation template.

Example 4.6. Let us consider Example 4.3. The explanation templates and their enriched version for the reasoning paths identified via structural analysis are presented in Figure 6.

4.3 Mapping chase steps to templates

As previously discussed, any materialized path over an instance, captured in the *chase graph*, can be mapped to a set of adjacent reasoning paths and, consequently, to a set of explanation templates. The composition of explanation templates that should be used for a materialized chase path of interest is built by (i) finding the simple reasoning path Π_i that instantiates the highest number of the first j chase steps and, if the leaf node is not reached yet, (ii) adding the reasoning cycle Γ_j that instantiates the highest number of the following n chase steps. Step (ii) is repeated until the leaf node is reached.

Example 4.7. Let us consider again Example 4.3 and the portion of chase graph on an artificial EDB, from which $Default("C")$ can be derived, depicted in Figure 8.

Following the chase steps, i.e., the ordered set of activated rules, we have: $\pi = \{\alpha, \beta, \gamma, \beta, \gamma\}$. By progressively considering the ordered chase steps, we find that Π_1 applies to the first step. However, the simple reasoning path that could be applied to the highest number of chase steps is Π_2, which instantiates the first three ones, α, β, γ. Then, for the remaining ones, a reasoning cycle must be applied: here both available reasoning cycles Γ_1, and Γ_2 cover the pair β, γ: since there are multiple inputs for the aggregation in β, then Γ_2 is selected. Therefore, the materialized path under analysis can be explained by the *reasoning graph* obtained by combining the explanation templates associated with Π_2 and Γ_2.

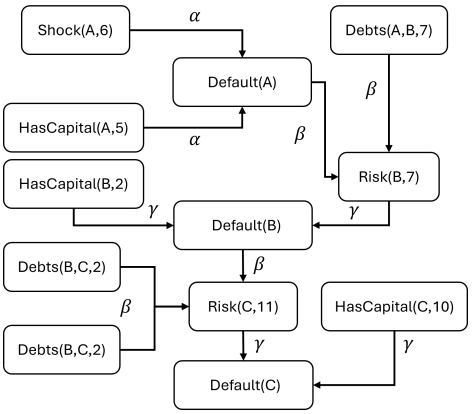


Figure 8: Portion of the chase graph deriving the default of entity "C", i.e., $Default(C)$. Nodes denote facts while rule-labeled edges illustrate the activated rule that has been activated to derive a new fact.

Finally, given an *explanation query* from users, i.e., a query asking for producing the explanation for a specific derived fact interest, the mapping of tokens with constants is trivially achieved by performing a template-wise substitution, replacing the tokens with the corresponding constants used in the portion of the chase graph which is used to derive the fact under analysis.

Example 4.8. The textual explanation for the explanation query $Q_e = \{Default(C)\}$ of Example 4.7 becomes: *"A" is in default due to a shock of 6M euros, being over its capital of 5M euros. With 7M euros debts to "A", "B" is at risk due to having 7M euros of exposure to a defaulted debtor. "B" has a capital of 2M euros, lower than 7M, thus also being in default. Defaulted debtor "B" leaves lender "C" exposed with a total of 11M euros loan (sum of loans of 2M and 9M euros). This puts "C", a financial institution with a capital of 10M euros, in default as well due to insufficient reserves.*

4.4 An automated pipeline

In the context of the Vadalog system, the chase-based reasoning engine based on the Vadalog language and adopted in the Bank of Italy, we implemented the structural analysis and the automatic template generation and selection by developing a dedicated module that can be activated over any deployed Knowledge Graph application [1]. The verbalizer was already available [4] and can be adapted to obtain the deterministic explanation templates. By integrating such components, we get a fully automated pipeline that implements our approach and enables its practical use over consolidated and new KG applications. Any explanation query for a fact can be obtained through this pipeline, avoiding any data sharing with LLM owners but, as we shall see, maintaining an overall comparable quality.

Dealing with Templates Hallucinations. As templates are passed to an LLM, no theoretical framework guarantees that its results are free of hallucinations. However, since templates for recurring KG applications can be pre-computed, they can also be checked by the Vadalog experts who defined the KG applications to understand whether the templates are fully informative. Such additional human-in-the-loop steps would support the goal of having complete explanations and do not undermine automation since it would be a once-for-all activity. A particular case of hallucinations is omissions, i.e., deletion of rule variables from the templates. To tackle this, an automatic preventive check controlling the presence of all variables throughout the template text has been implemented.

[1]The implementation and an example are available at: https://bit.ly/EDBT25-explain

5 FINANCIAL KNOWLEDGE GRAPH APPLICATIONS

We now present a couple of the most relevant Knowledge Graph applications that are being used in the financial Enterprise Knowledge Graph of the Bank of Italy [8], running on the Vadalog system. For each of these applications, generating natural language explanations is of great relevance for decision support for non-IT compliance staff and auditors, with the prospective business impact in terms of increased productivity and systemic stability [39]. However, adopting a generative AI solution in such a context is subject to the sharing of possibly confidential information that a central bank is not allowed to share with third parties, such as in the case of banking supervisory tasks [42].

Company Control. The *company control* program [9] finds chains of controls between companies and 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 knowledge graph with "*control*" edges, as follows:

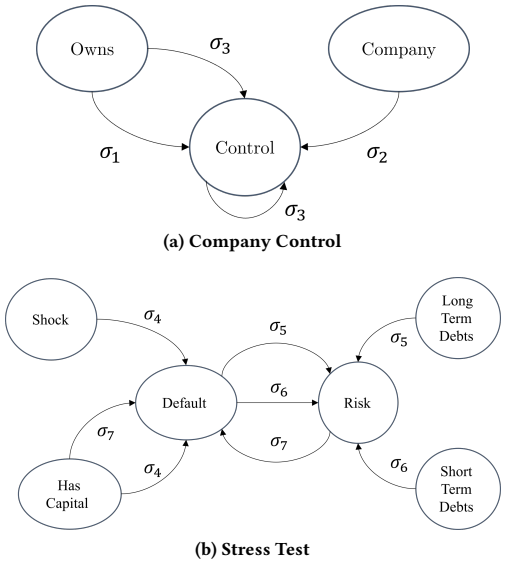
$$Own(x, y, s), s > 0.5 \rightarrow Control(x, y) \quad (\sigma_1)$$
$$Company(x) \rightarrow Control(x, x) \quad (\sigma_2)$$
$$Control(x, z), Own(z, y, s), ts = sum(s), ts > 0.5 \rightarrow Control(x, y) \quad (\sigma_3)$$

It follows the official definition: *"A company (or a person) x controls a company y, if: (i) x directly owns more than 50% of y; or, (ii) x controls a set of companies that jointly (i.e., summing the shares), and possibly together with x, own more than 50% of y"*. Although the program contains only three rules, complex corporate structures might occur, with companies that do not have immediate transparency of ownership and/or control. This is the case of a company with several layers of shareholders, which makes it difficult to see who the beneficial owners are and raises the question of their transparency. Discovering and describing such a long chain of controls is paramount for efficient financial market supervision.

Stress Tests. The following set of rules encodes a more interesting scenario of stress test simulation, presented in Example 4.3. It represents the propagation of a default shock over multiple channels, i.e., the short-term and long-term debt exposures, assessing how the resulting cascade defaults affect a network representing the interconnection in the financial system. The distinction between different channels is essential to understanding how an entity's exposures are distributed and what is their contribution to a cascade default.

$$Shock(f, s), HasCapital(f, p_1), s > p_1 \rightarrow Default(f) \quad (\sigma_4)$$
$$Default(d), LongTermDebts(d, c, v), e_l = sum(v) \rightarrow Risk(c, e_l, \text{"long"}) \quad (\sigma_5)$$
$$Default(d), ShortTermDebts(d, c, v), e_s = sum(v) \rightarrow Risk(c, e_s, \text{"short"}) \quad (\sigma_6)$$
$$Risk(c, e, t), HasCapital(c, p_2), l = sum(e), l > p_2 \rightarrow Default(c) \quad (\sigma_7)$$

A financial institution f defaults when an exogenous shock of s euro deteriorated its capital p_1 to the point it becomes negative (rule σ_4). Whenever a financial institution defaults, it impacts its



(a) Company Control

(b) Stress Test

Figure 9: Dependency Graphs of the Financial Knowledge Graph Applications

KG Application	Simple Reasoning Path	Reasoning Cycle
Company Control	$\Pi_1 = \{\sigma_1\}$ $\Pi_2^* = \{\sigma_1, \sigma_3\}$ $\Pi_3 = \{\sigma_2\}$ $\Pi_4^* = \{\sigma_2, \sigma_3\}$ $\Pi_5^* = \{\sigma_1, \sigma_2, \sigma_3\}$	$\Gamma_1^* = \{\sigma_3\}$
Stress Test	$\Pi_6 = \{\sigma_4\}$ $\Pi_7^* = \{\sigma_4, \sigma_5, \sigma_7\}$ $\Pi_8^* = \{\sigma_4, \sigma_6, \sigma_7\}$ $\Pi_9^* = \{\sigma_4, \sigma_5, \sigma_6, \sigma_7\}$	$\Gamma_2^* = \{\sigma_5, \sigma_7\}$ $\Gamma_3^* = \{\sigma_6, \sigma_7\}$ $\Gamma_4^* = \{\sigma_5, \sigma_6, \sigma_7\}$

Figure 10: Simple reasoning paths and reasoning cycles of the Financial Knowledge Graph applications we analyze. We use the * superscript to denote reasoning paths whose aggregation alternative version is also available.

creditors via long-term and short-term exposures. The financial institutions c are at risk of failure, featuring a total e_l of long-term loan exposures (rule σ_5) and/or e_s of short-term exposures (rule σ_6), measured in euro, to the defaulted entities. If the total l of exposures for the creditor c is higher than its capital p_2, then the creditor c defaults (rule σ_7). The set of rules aims to derive all the defaults that an initial shock affecting one or more entities might trigger.

Studying and analyzing how a financial shock propagates in the financial market, i.e., via which channel, is of critical importance for authorities [35], which can swiftly adopt countermeasures to prevent or, at least, mitigate a cascade effect.

Structural Analysis. The dependency graphs for the EKG applications are presented in Figure 9. All dependency graphs are cyclic, thus all potentially producing textual explanations of unpredictable length. It is important to notice that while, in general, recursion may lead to infinite chase sequences, this is not the case in the Vadalog context, where the application of chase steps that generate facts isomorphic to facts already in the chase is preempted while upholding correctness of the reasoning task [11]. On top of the dependency graphs, we can derive reasoning paths by applying the definitions. We represent them in Figure 10, adopting the compact rule-based notation.

Predicate	Description
$Owns(x, y, s)$	<x> owns <s> shares of <y>.
$Control(x, y)$	<x> exercises control over <y>.
$Company(x)$	<x> is a business corporation.
$HasCapital(f, p)$	<f> is a company with capital of <p> euros
$Shock(f, s)$	A shock amounting to <s> euro hits <f>
$Default(f)$	<f> is in default.
$LongTermDebts(d, c, v)$	<d> has an amount <v> of long-term debts with <c>.
$ShortTermDebts(d, c, v)$	<d> has an amount <v> of short-debts with <c>.
$Risk(c, e, l)$	<c> is at risk of defaulting given its <l>-term loans of <e> euros of exposures to a defaulted debtor.

Figure 11: Domain Glossary derived from the internal Data Dictionary for the Financial Knowledge Graph applications.

For the company control program, we identify two root nodes, *Owns* and *Company*, and a leaf node, *Control*, which is also critical. Therefore, we derive the simple reasoning paths $\Pi_1, \Pi_2, \Pi_3, \Pi_4$, which are all the potential paths that can be followed to connect the root nodes to the leaf node. Additionally, a simple reasoning path Π_5 can also be derived, as it originates in two distinct root nodes that jointly conduct to the leaf. The only reasoning cycle is the path connecting the leaf node to itself via rule σ_3.

We already derived reasoning paths for a simplified version of the stress test application in Section 4. Here, roots and the leaf node remain unchanged; however, more complexity arises in the intensional predicates, with the distinction of the two exposure channels, i.e., long and short-term debts. First, individual simple reasoning paths for each channel are derived (i.e., Π_7, Π_8), and, similarly to the company control case, a joint channel path starting from the root can also be detected, i.e., Π_9. The same applies to reasoning cycles, i.e., $\Gamma_2, \Gamma_3, \Gamma_4$.

Generation of Explanation Templates. By gathering the predicate description from the internal data dictionary, we present in Figure 11 the domain glossary for the above Financial Knowledge Graph applications which can be used to produce the deterministic explanation templates. With the domain glossary available, the generation of explanation templates follows the procedure presented in Section 4.2.

Representative Scenario. In Figure 12 we represent a portion of an artificial financial knowledge graph, resembling a real-world scenario of financial institutions that a business analyst might face to investigate. To this aim, the above rule-based applications might be run to (i) discover if and how these companies are in special control relationships and (ii) run a stress test to simulate how a shock impacts and propagates over the cluster of institutions under analysis. By running the company control application, new knowledge is derived via the chase procedure. The result is shown in Figure 13. For each derived fact, an explanation query Q_e can be run to generate an explanation, activating the template-based approach. For instance, the business analyst might be interested in understanding how the control between "B" and "D" has been derived and thus perform the following query: $Q_e = \{Control(B, D)\}$. The template-based system would track the facts in the underlying chase graph followed by the

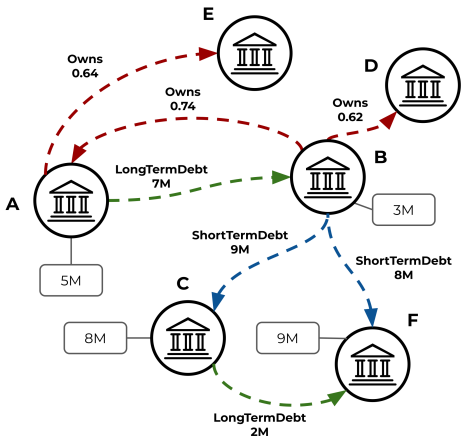


Figure 12: Portion of a synthetic extensional knowledge of a Financial Knowledge Graph. Rectangular nodes represent *HasCapital* facts. Red edges represent both *Owns* and *IntOwns* facts.

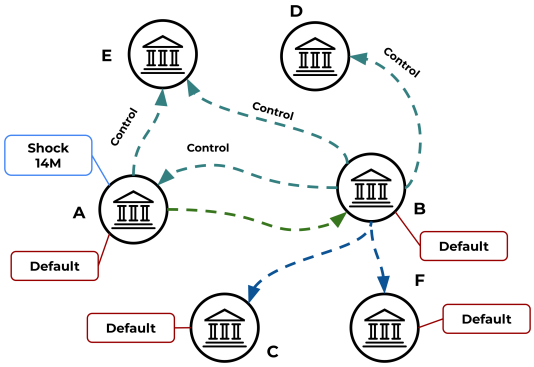


Figure 13: Derived knowledge applying the control and stress test applications. Auto-control edges, i.e., edges denoting the facts that companies control themselves, are omitted. We simulate a financial shock to entity *A* (of 15M$)

reasoning engine, as we did in Figure 8, and identify the corresponding reasoning path followed - that in this scenario is Π_2.

We also simulate a financial shock hitting entity A and we run the stress test program to understand its effect on the system. The derived knowledge, i.e., *Default* values, can be investigated by generating a natural language explanation of how the shock has propagated in the network. For instance, for $Q_e = \{Default(F)\}$, we get the following explanation: *"A financial shock amounting to 14 million euros impacts A's capital (worth 5 million euros), leading to its default. As a result, company B, which holds 7 million euros of long-term debts from A, faces risk due to its exposure to the defaulted company. With 7 million euros in exposures to the defaulted company, B's capital of 4 million euros is insufficient to absorb the loss, leading to B's default as well. With B defaulting, its short-term debt to creditor C, amounting to 9 million euros puts C at risk. Despite C's capital reserves of 8 million euros, its overall debt obligations worth 9 million euros led it to default. C and B both defaulted and they had, respectively, 2 and 8 million euros of long and short-term exposures with F, putting it at risk with a total debt of 11 million euros. Despite having 9 million euros in capital, the scale of its total debt led to F also defaulting on its financial obligation."*. This text has been obtained by utilizing explanation templates corresponding to reasoning paths $\{\Pi_7, \Gamma_3, \Gamma_4^*\}$, which instantiates the logical steps that allow the derivation of fact $Default(F)$.

6 EXPERIMENTAL EVALUATION

We evaluate our approach by demonstrating it can produce high-quality textual explanations. First, we devised two user-study experiments based on common approaches in the literature to (i) evaluate the comprehension of non-expert users when faced with our textual explanation and (ii) evaluate the effectiveness of our template-based approach in producing fluent texts for domain experts by comparing it with plain AI-generated texts [58]. Our experiments are centered around textual explanations of knowledge inferred from artificial data generated automatically for the KG applications presented in Section 5. Then, we verify that employing an LLM-based solution – the alternative approach to our template-based one – while being overall equivalent in terms of textual readability to our solution, is subject to omissions, especially when the proofs of facts become longer, as it is common in financial scenarios, for instance in case of very long control chains or contagion resulting from the propagation of shocks. Finally, we discuss the performances of generating explanations with our template-based approach.

6.1 Users Comprehension

We first evaluate the comprehension of our explanations with users that are not expert of the financial domain. We created a set of five multi-choice questions related to various instances of the financial applications described in Section 5. Each question contained a "business report" that we sampled from the pool of all textual explanations generated in our artificial KG. In particular, we selected a case of control through aggregation over multiple entities (1), a simple stress test scenario (2), control via recursion (3), a complex stress test involving recursion and aggregation (4), and a case of control combining recursion and aggregation (5). For each question, we generated three visualizations in the form of graphs, similar to Figures 12 and 13, where one was corresponding to the explanation, while the other two contained errors. The idea is that if non-expert users can identify the correct visual graph-based explanation, the proposed textual explanation is comprehensive, i.e., it helps users to understand the inference mechanism over the KG.To generate the errors, we identified four *archetypes*, similarly to [26]: (I) the presence of false edges, (II) incorrect value of a property, (III) incorrect order of aggregation values, and (IV) incorrect chain in case of recursion. Such error types encapsulate the main problems that make the task of generating a fluent and accurate explanation complex, as outlined in Section 4.1.

We recruited 24 participants with no or little financial background, and all fluent in English. This resulted in the collection of 120 answers. Figure 14 summarizes the results for each case study analysed by our tested users. In general, we achieved a very high accuracy of 96%, meaning that in almost all cases, our users detected the correct KG based on our explanation. Regarding errors made by users, no clear pattern (i.e., archetype that systematically leads to user errors) can be identified, meaning that our explanations were comprehensive and accurate.

6.2 Expert User Study

For each simulated scenario, we presented three possible textual explanations of the same proof, one produced using our approach and the other two generated by a pure LLM solution, i.e., by

Case Study	Error Archetype				Correct Answers
	Wrong Edge	Wrong Value	Incorrect Aggregation	Incorrect Chain	
1	8%	0%	0%	0%	92%
2	0%	0%	0%	0%	100%
3	4%	0%	0%	0%	96%
4	0%	0%	0%	0%	100%
5	0%	0%	4%	4%	92%

Figure 14: Results of the comprehension user study, presenting the relative number of users that answered correctly and the corresponding errors, if any. Users were presented with five cases and, for each, tested to select the visual KG that captured the knowledge of each text. Multiple KG visualizations were presented, with only one being the correct version and the others containing one of the error archetypes.

prompting the deterministic explanation of the portion of the chase graph of interest, obtained by applying the verbalizer to the instance, as we did in [4]. We consider ChatGPT as our benchmark LLM model and two prompts, one asking for a paraphrase to get a fluent and complete version of the deterministic explanation and the other asking for a summary to obtain a not redundant text. The prompts are:

(1) *"Generate a paraphrased version of the following text: ..."*
(2) *"Generate a summarized version of the following text: ..."*

We had 14 expert users, i.e., Central Bank employees with a solid economic background and capable of understanding the context of applying our business rules. We did not provide details about how these texts were generated, but we just hinted that they all resulted from the application of automatic text tool generators. While other studies asked their participants to evaluate several quality dimensions, we presented the above-mentioned dimensions to participants and asked them to assess the textual quality by assigning an overall mark to facilitate and speed up the evaluation process, as textual explanations of complex reasoning inferences quickly become long.

Each participant was asked to analyze four scenarios: two of them for the company control application (a short control chain and a long one with multiple layers of intermediate controls), one from the stress test application and one from the close link application, another financial application from our domain [2]. For each of these scenarios, three textual explanations were generated and graded. This resulted in collecting 168 individual data points, 56 for each methodology, i.e., GPT with paraphrasing prompt, GPT with summarization prompt, and the template-based approach. The order of texts was randomly shuffled, and the methodology used to generate the text was not shown in the input form. Participants graded each text on a 5-value Likert scale.

In Figure 15 we show an example output of the explanation methods we tested. We report the user study results in Figure 16, in terms of average Likert values and corresponding standard deviations for the methodologies that have been considered.

We were expecting no significant difference in the overall quality of the explanations generated with our system compared with the ones of GPT. Following previous studies on comparing responses on a Likert scale [25, 27], we conducted pairwise Wilcoxon signed-rank tests considering two explanation methods at a time. We found no significant differences between any of the tested pairs. In particular, the p-value for the two-sample

Deterministic Explanation

Since Irish Bank is a company, and Irish Bank owns 83% of Fondo Italiano shares, and 83% is over 50, then Irish Bank controls Fondo Italiano. Since Irish Bank is a company, and Irish Bank owns 54% of FrenchPLCs shares, and 54 is over 50, then Irish Bank controls FrenchPLC. Since Irish Bank controls FrenchPLC, and FrenchPLC owns 21% of Madrid Credit shares, and Irish Bank controls Fondo Italiano, and Fondo Italiano owns 36% of Madrid Credit shares, then IrishBank owns a total 57% of Madrid Credit shares. Since Irish Bank owns a total 57% of Madrid Credit shares, and 57 is over 50, then IrishBank controls Madrid Credit.

GPT Paraphrasis of Deterministic Explanation

Irish Bank has ownership of 83% of Fondo Italiano, which means it has control over the company. Additionally, Irish Bank owns 54% of French PLC, establishing its control over that company as well. Since FrenchPLC possesses 21% of Madrid Credit, and Fondo Italiano holds 36% of Madrid Credit, it follows that Irish Bank has ownership of a combined 57% of Madrid Credit. Hence, Irish Bank exercises control over Madrid Credit.

GPT Summary of Deterministic Explanation

Irish Bank owns a majority stake in Fondo Italiano and French PLC, enabling it to control both companies. Additionally, French PLC owns shares in Madrid Credit, and since Irish Bank controls FrenchPLC and Fondo Italiano, it ultimately controls a significant portion of Madrid Credit. Thus, Irish Bank has overall control of Madrid Credit as well.

Template-based Approach

Irish Bank owns 83% Fondo Italiano and 54% of French PLC through Fondo Italiano. Thus, Irish Bank controls Fondo Italiano and FrenchPLC. Irish Bank, which controls FrenchPLC and Fondo Italiano that own, respectively, 21% and 36% of Madrid Credit, thereby owns 57% of Madrid Credit, effectively giving Irish Bank control over it.

Figure 15: Example of textual explanations for the same fact (i.e., Irish Bank exercises control over Madrid Credit). Users were asked to evaluate the last three versions.

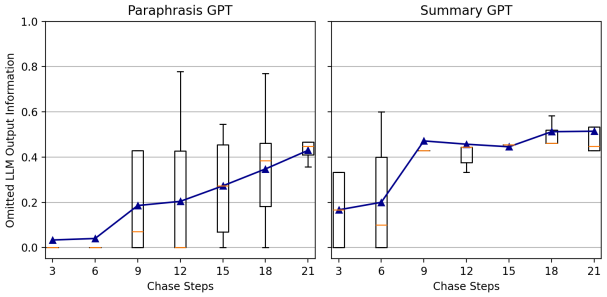
	Paraphrasis	Summary	Templates
Mean	3.78	3.765	3.69
Std. Dev.	1.09	1.25	0.94

Figure 16: Mean Likert value and standard deviation for each methodology

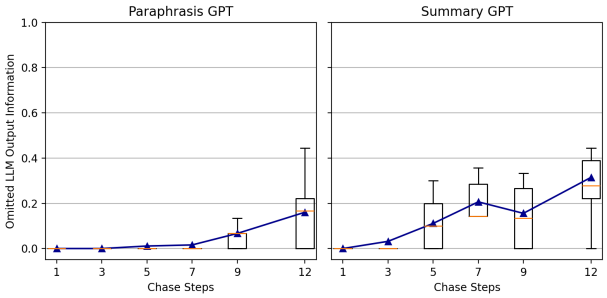
Wilcoxon test between Likert values of GPT-based paraphrasis and our templates is $p_1 = 0.5851$; similarly, the p-value for GPT-based summarization and our templates is $p_2 = 0.404$. Both values are far from being significant, confirming our expectations. This user study confirms that our approach produces texts of the same quality as GPT's, without sharing the underlying data with third parties.

6.3 Completeness of textual explanations

As a last experiment, we discuss one of the main technical limitations of relying on an LLM-based solution in our context, i.e., using it over the deterministic explanations to get human-readable versions. The automatic generation of business reports requires that the outcome guarantees high trustworthiness. If a tool for textual generation omits important information or, in worst cases, generates hallucinations, it cannot be approved for use in critical text generation tasks, such as those employed by the Central Bank. In our experiment, we test ChatGPT, the same model we employed to generate the templates, to produce textual explanations of proofs for the KG applications we presented in Section 5. We show how it is subject to omissions, which undermines the completeness criterion of business reports. Such behavior does not influence our template-based approach, which,



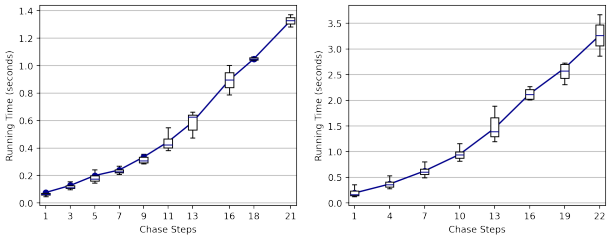
(a) Company Control



(b) Stress Test

Figure 17: Relative average proportion of missing information in the output text of ChatGPT – our benchmark for generating explanations – asked to paraphrase and summarize the deterministic verbalization of proofs of increasing sizes, i.e., longer explanations. The length of inference is measured in terms of the number of chase steps; information loss is measured as the ratio between the number of constants present in the textual explanation and the number of facts required by the correct inference. Boxplots refer to ten distinct sampled proofs with equal length. Within our system, we can avoid such omissions by leveraging the template-based technique and a once-for-all human-in-the-loop step.

by construction, contains all constants used during the inference process, as they are captured by tokens. To this aim, we applied the company control and the stress test KG applications over artificially generated data, as individual shares and loan information are confidential. We sampled subsets of distinct edges of proofs of increasing length, measured in the number of logical steps required to derive a conclusion. We asked ChatGPT to generate a paraphrasis and a summarized explanation for each proof by prompting it with the deterministic explanation we obtained via our verbalizer. We then measured the ratio of omitted information for each ChatGPT output, i.e. the relative number of constants displayed in the final textual explanation w.r.t. the constants used in the proof for deriving a conclusion. For each considered proof length, we obtained a distribution of 10 ratios of omitted information, as represented by Figure 17 with the boxplot. As expected, the average ratio grows with the proof length, meaning that, as the inference process grows, ChatGPT tends to lose information that might be relevant for analysts. In general, we observed the paraphrasing task is less subject to such behavior, compared to the summarization one, but still, some information in the textual output is missing. More specifically, for the company control application, omissions refer, in most cases, to ownership share amounts that are not reported in the final text, both in the case of summarization and paraphrases; instead, in the case of stress test, no specific patterns in the omissions could be identified.



(a) Company Control (b) Stress Test

Figure 18: Average running times required for generating explanations of increasing inference length using our approach. Boxplots are distributions of 15 distinct proofs with the same chase step length.

6.4 Performances

We finally analyze the running times of our template-based approach when the proof length increases, i.e., the time required to select, parse, and combine templates when the explanation becomes more complex and more inference steps are required. We ran the experiment on a Windows 11 machine with AMD Ryzen 5 5500U and 8 GB 3200 MHz DDR4 memory. Results are in Figure 18 and display the average running times of translating proofs of different lengths into textual explanations. As expected, running times increase with the number of inference steps required. However, the complexity of the KG applications also plays a role since the stress test application - which is syntactically more complex, having multiple rules with aggregations - displays higher running times. However, in both scenarios, running times remain low and acceptable for a user-friendly experience, with a maximum of around 3 seconds in the case of most complex stress test cases, requiring over 20 inference steps.

6.5 Discussion and limitations

For generating the explanation templates and for paraphrasing and summarizing in our experiments, we used and tested ChatGPT, the most widespread generative AI model and the most natural choice for non-IT business analysts. Another option for producing explanations would be to have an internal, lighter, fine-tuned language model for generating explanations. However, such fine-tuning would (i) require a training set of domain-specific explanations and (ii) a new fine-tuning whenever a new KG application is deployed. Instead, our approach is database-independent and directly applicable to any new application. Also, our approach focuses on the financial domain, which is the one of interest for our applications. However, the quality of results does not depend on any training data but on the rephrasing power of LLMs. Therefore, we believe our approach can achieve similar results in any other domain, once equipped with an internal data dictionary. Other prompts could have been used for both experiments; however, while prompt engineering can greatly influence results, no prompt guarantees perfect consistency [60], which, in our case, refers to the absence of omissions.

7 CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel approach for generating high-quality textual explanations for knowledge inferred over complex financial Enterprise Knowledge Graph applications, i.e.,

containing recursion and aggregations that generate non-trivial provenances. Our approach leverages the intrinsic simplicity and transparency of Datalog-based languages, such as Vadalog, and their syntax to generate fluent and compact textual templates that capture the main interconnections between business rules that express problems of interest for analysts under the form of Knowledge Graph application and without relying on any concrete database. This is possible as we exploit commonly available resources of corporations, such as data dictionaries. To fully automatize the process, we employed modern Large Language Models in a self-sustained, low-cost framework to create the templates, which also avoids the breach of data confidentiality. We tested our approach over three important KG applications and we conducted a user study to show how, at least for the financial domain, our approach creates textual explanations that are fluent and readable, being competitive with pure LLM-based methodologies that generate business reports, i.e., reasoning explanations, but require the share of the actual instance. We also analyzed how directly employing generative AI to produce explanations can lead to omissions, especially when proofs get longer, harming the completeness of business reports. In contrast, by being completely controllable and tractable, our template-based approach contains instead all the necessary information that a business analyst might require. Although limited to the financial domain, the approach can be easily replicated over other Knowledge Graph applications for corporations working in other domains, as the quality of results depends on internal data dictionaries, and no training or fine-tuning of LLMs is involved. In future work, we will test our system in other domains and test whether the advantages over plain LLM systems are still relevant, such as in the financial one.

Resources. The implementation of our approach, which is based on the Vadalog system, is available in a GitHub repository (https://github.com/andreac0/Template-Based-Inference-EDBT). Synthetic financial data to run our applications are instead available at: https://bit.ly/edbt25Data.

ACKNOWLEDGMENTS

Stefano Ceri is supported by the PNRR-PE-AI FAIR project, funded by the NextGenerationEU program. Andrea Colombo kindly acknowledges INPS for the funding of his Ph.D. program. 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] Serge Abiteboul, Richard Hull, and Victor Vianu. 1995. *Foundations of databases*. Vol. 8. Addison-Wesley Reading.

[2] Paolo Atzeni, Luigi Bellomarini, Michela Iezzi, Emanuel Sallinger, and Adriano Vlad. 2020. Weaving Enterprise Knowledge Graphs: The Case of Company Ownership Graphs. In *Proceedings of the 23rd International Conference on Extending Database Technology*. openproceedings.org, Copenhagen, Denmark, 555–566.

[3] Jean-François Baget, Michel Leclère, and Marie-Laure Mugnier. 2010. Walking the Decidability Line for Rules with Existential Variables. *KR* 10 (2010), 466–476.

[4] Teodoro Baldazzi, Luigi Bellomarini, Stefano Ceri, Andrea Colombo, Andrea Gentili, and Emanuel Sallinger. 2024. 'Please, Vadalog, tell me why': Interactive Explanation of Datalog-based Reasoning. In *Proceedings 28th International Conference on Extending Database Technology, EDBT 2024, Paestum, Italy, March 28-28, 2024*. OpenProceedings.org, Paestum, Italy, 834–837.

[5] Teodoro Baldazzi, Luigi Bellomarini, Stefano Ceri, Andrea Colombo, Andrea Gentili, Emanuel Sallinger, and Paolo Atzeni. 2024. Explaining Enterprise Knowledge Graphs with Large Language Models and Ontological Reasoning. In *The Provenance of Elegance in Computation - Essays Dedicated to Val Tannen (Open Access Series in Informatics (OASIcs))*, Antoine Amarilli and Alin Deutsch (Eds.), Vol. 119. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 1:1–1:20. https://doi.org/10.4230/OASIcs.Tannen.1

[6] Teodoro Baldazzi, Luigi Bellomarini, Emanuel Sallinger, and Paolo Atzeni. 2021. Eliminating Harmful Joins in Warded Datalog+/-. In *RuleML+RR (Lecture Notes in Computer Science)*, Vol. 12851. Springer, Cham, 267–275.

[7] Pablo Barceló and Reinhard Pichler. 2012. *Datalog in Academia and Industry: Second International Workshop, Datalog 2.0*. Vol. 7494. Springer, Vienna, Austria.

[8] 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. 2022. Reasoning on company takeovers: From tactic to strategy. *Data Knowl. Eng.* 141, C (sep 2022), 24. https://doi.org/10.1016/j.datak.2022.102073

[9] Luigi Bellomarini, Marco Benedetti, Andrea Gentili, Rosario Laurendi, Davide Magnanimi, Antonio Muci, and Emanuel Sallinger. 2020. COVID-19 and Company Knowledge Graphs: Assessing Golden Powers and Economic Impact of Selective Lockdown via AI Reasoning. *CoRR* abs/2004.10119 (2020). arXiv:2004.10119 https://arxiv.org/abs/2004.10119

[10] Luigi Bellomarini, Marco Benedetti, Andrea Gentili, Davide Magnanimi, and Emanuel Sallinger. 2023. KG-Roar: Interactive Datalog-Based Reasoning on Virtual Knowledge Graphs. *Proceedings of the VLDB Endowment* 16, 12 (2023), 4014–4017.

[11] Luigi Bellomarini, Emanuel Sallinger, and Georg Gottlob. 2018. The Vadalog System: Datalog-Based Reasoning for Knowledge Graphs. *Proc. VLDB Endow.* 11, 9 (may 2018), 975–987. https://doi.org/10.14778/3213880.3213888

[12] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901. https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[13] Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. 2001. Why and Where: A Characterization of Data Provenance. In *Database Theory — ICDT 2001*, Jan Van den Bussche and Victor Vianu (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 316–330.

[14] Pedro Cabalar, Jorge Fandinno, and Brais Muñiz. 2020. A System for Explainable Answer Set Programming. *Electronic Proceedings in Theoretical Computer Science* 325 (2020), 124–136. https://doi.org/10.4204/eptcs.325.19

[15] Andrea Calì, Georg Gottlob, and Michael Kifer. 2013. Taming the Infinite Chase: Query Answering under Expressive Relational Constraints. *J. Artif. Intell. Res.* 48 (2013), 115–174.

[16] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. 2009. Datalog±: A Unified Approach to Ontologies and Integrity Constraints. In *ICDT 2009*. Association for Computing Machinery, New York, NY, USA, 14–30. https://doi.org/10.1145/1514894.1514897

[17] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. 2012. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Semant.* 14 (2012), 57–83. https://doi.org/10.1016/j.websem.2012.03.001

[18] Andrea Calì, Georg Gottlob, and Andreas Pieris. 2010. Advanced Processing for Ontological Queries. *Proc. VLDB Endow.* 3, 1 (2010), 554–565.

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

[20] Andrea Calì, Georg Gottlob, Thomas Lukasiewicz, Bruno Marnette, and Andreas Pieris. 2010. Datalog+/-: A Family of Logical Knowledge Representation and Query Languages for New Applications. In *2010 25th Annual IEEE Symposium on Logic in Computer Science*. 228–242. https://doi.org/10.1109/LICS.2010.27

[21] Luciano Caroprese, Eugenio Vocaturo, and Ester Zumpano. 2022. Argumentation approaches for explanaible AI in medical informatics. *Intelligent Systems with Applications* 16 (2022), 200109. https://doi.org/10.1016/j.iswa.2022.200109

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

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

[24] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. 2001. Complexity and Expressive Power of Logic Programming. *ACM Comput. Surv.* 33, 3 (sep 2001), 374–425. https://doi.org/10.1145/502807.502810

[25] Joost FC de Winter and Dimitra Dodou. 2019. Five-point likert items: t test versus Mann-Whitney-Wilcoxon (Addendum added October 2012). *Practical Assessment, Research, and Evaluation* 15, 1 (2019), 11.

[26] Matt Dennis, Kees Van Deemter, Daniele Dell'Aglio, and Jeff Z Pan. 2017. Computing authoring tests from competency questions: Experimental validation. In *The Semantic Web–ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21–25, 2017, Proceedings, Part I 16*. Springer, Vienna,

Austria, 243–259.
[27] Ben Derrick and Paul White. 2017. Comparing two samples from an individual Likert question. *International Journal of Mathematics and Statistics* 18, 3 (2017), 1–13.
[28] Daniel Deutch, Nave Frost, and Amir Gilad. 2016. Nlprov: Natural language provenance. *Proceedings of the VLDB Endowment* 9, 13 (2016), 1537–1540.
[29] Daniel Deutch, Nave Frost, and Amir Gilad. 2017. Provenance for Natural Language Queries. *Proc. VLDB Endow.* 10, 5 (jan 2017), 577–588. https://doi.org/10.14778/3055540.3055550
[30] Daniel Deutch, Nave Frost, and Amir Gilad. 2018. Natural Language Explanations for Query Results. *SIGMOD Rec.* 47, 1 (sep 2018), 42–49. https://doi.org/10.1145/3277006.3277017
[31] Daniel Deutch, Nave Frost, and Amir Gilad. 2018. Provenance for Non-Experts. *IEEE Data Eng. Bull.* 41, 1 (2018), 3–14.
[32] Esra Erdem and Umut Oztok. 2015. Generating explanations for biomedical queries. *Theory and Practice of Logic Programming* 15, 1 (2015), 35–78.
[33] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. 2005. Data exchange: semantics and query answering. *Theor. Comput. Sci.* 336, 1 (2005), 89–124. https://doi.org/10.1016/j.tcs.2004.10.033
[34] Jorge Fandinno and Claudia Schulz. 2019. Answering the "why" in answer set programming–A survey of explanation approaches. *Theory and Practice of Logic Programming* 19, 2 (2019), 114–203.
[35] Fabio Fornari and Livio Stracca. 2012. What does a financial shock do? First international evidence. *Economic Policy* 27, 71 (2012), 407–445.
[36] Boris Glavic, Renée J. Miller, and Gustavo Alonso. 2013. *Using SQL for Efficient Generation and Querying of Provenance Information.* Springer Berlin Heidelberg, Berlin, Heidelberg, 291–320. https://doi.org/10.1007/978-3-642-41660-6_16
[37] Sergio Greco and Cristian Molinaro. 2016. *Datalog and Logic Databases.* Springer International Publishing. https://doi.org/10.1007/978-3-031-01854-1
[38] Todd J Green, Shan Shan Huang, Boon Thau Loo, Wenchao Zhou, et al. 2013. Datalog and recursive query processing. *Foundations and Trends® in Databases* 5, 2 (2013), 105–195.
[39] Benjamin N. Grosof, Janine Bloomfield, Paul Fodor, Michael Kifer, Isaac Grosof, Miguel Calejo, and Terrance Swift. 2015. Automated Decision Support for Financial Regulatory/Policy Compliance, using Textual Rulelog. In *Proceedings of the RuleML 2015 Challenge, the Special Track on Rule-based Recommender Systems for the Web of Data, the Special Industry Track and the RuleML 2015 Doctoral Consortium hosted by the 9th International Web Rule Symposium (RuleML 2015) (CEUR Workshop Proceedings)*, Nick Bassiliades, Paul Fodor, Adrian Giurca, Georg Gottlob, Tomás Kliegr, Grzegorz J. Nalepa, Monica Palmirani, Adrian Paschke, Mark Proctor, Dumitru Roman, Fariba Sadri, and Nenad Stojanovic (Eds.), Vol. 1417. CEUR-WS.org, Berlin, Germany. https://ceur-ws.org/Vol-1417/paper8.pdf
[40] Melanie Herschel and Marcel Hlawatsch. 2016. Provenance: On and Behind the Screens. In *SIGMOD 2016*. Association for Computing Machinery, New York, NY, USA, 2213–2217. https://doi.org/10.1145/2882903.2912568
[41] Jason I. Hong. 2023. Teaching the FATE Community about Privacy. *Commun. ACM* 66, 8 (jul 2023), 10–11.
[42] Michael Ioannidis. 2023. The principle of confidentiality in banking supervision. *ESCB Legal Conference 2020* 95 (2023), 223.
[43] Sameer Jain, Vaishakh Keshava, Swarnashree Mysore Sathyendra, Patrick Fernandes, Pengfei Liu, Graham Neubig, and Chunting Zhou. 2023. Multi-Dimensional Evaluation of Text Summarization with In-Context Learning. In *ACL 2023, Toronto, Canada, July 9-14, 2023*, Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 8487–8495. https://aclanthology.org/2023.findings-acl.537
[44] David S. Johnson and Anthony C. Klug. 1984. Testing Containment of Conjunctive Queries under Functional and Inclusion Dependencies. *J. Comput. Syst. Sci.* 28, 1 (1984), 167–189.
[45] Amruta Kale, Tin Nguyen, Jr. Harris, Frederick C., Chenhao Li, Jiyin Zhang, and Xiaogang Ma. 2023. Provenance documentation to enable explainable and trustworthy AI: A literature review. *Data Intelligence* 5, 1 (03 2023), 139–162. https://doi.org/10.1162/dint_a_00119 arXiv:https://direct.mit.edu/dint/article-pdf/5/1/139/2074289/dint_a_00119.pdf
[46] David Koop, Marta Mattoso, and Juliana Freire. 2018. *Provenance in Workflows.* Springer New York, New York, NY, 2912–2916. https://doi.org/10.1007/978-1-4614-8265-9_80745
[47] Markus Krötzsch and Veronika Thost. 2016. Ontologies for Knowledge Graphs: Breaking the Rules. In *The Semantic Web – ISWC 2016*, Paul Groth, Elena Simperl, Alasdair Gray, Marta Sabou, Markus Krötzsch, Freddy Lecue, Fabian Flöck, and Yolanda Gil (Eds.). Springer International Publishing, Cham, 376–392.
[48] Seokki Lee, Bertram Ludäscher, and Boris Glavic. 2018. Provenance Summaries for Answers and Non-Answers. *Proc. VLDB Endow.* 11, 12 (aug 2018), 1954–1957. https://doi.org/10.14778/3229863.3236233
[49] David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv. 1979. Testing Implications of Data Dependencies. *ACM TODS* 4, 4 (1979), 455–469. https://doi.org/10.1145/320107.320115
[50] Abdul Majeed and Sungchang Lee. 2021. Anonymization Techniques for Privacy Preserving Data Publishing: A Comprehensive Survey. *IEEE Access* 9 (2021), 8512–8545. https://doi.org/10.1109/ACCESS.2020.3045700
[51] Heather S. Packer and Luc Moreau. 2015. Sentence Templating for Explaining Provenance. In *Provenance and Annotation of Data and Processes*, Bertram Ludäscher and Beth Plale (Eds.). Springer International Publishing, Cham, 278–280.
[52] Sabbir M. Rashid, James P. McCusker, Paulo Pinheiro, Marcello P. Bax, Henrique O. Santos, Jeanette A. Stingone, Amar K. Das, and Deborah L. McGuinness. 2020. The Semantic Data Dictionary – An Approach for Describing and Annotating Data. *Data Intelligence* 2, 4 (10 2020), 443–486. https://doi.org/10.1162/dint_a_00058 arXiv:https://direct.mit.edu/dint/article-pdf/2/4/443/1857498/dint_a_00058.pdf
[53] Darren P. Richardson and Luc Moreau. 2016. Towards the Domain Agnostic Generation of Natural Language Explanations from Provenance Graphs for Casual Users. In *Provenance and Annotation of Data and Processes*, Marta Mattoso and Boris Glavic (Eds.). Springer International Publishing, Cham, 95–106.
[54] Sudeepa Roy and Dan Suciu. 2014. A Formal Approach to Finding Explanations for Database Queries. In *SIGMOD 2014*. Association for Computing Machinery, New York, NY, USA, 1579–1590. https://doi.org/10.1145/2588555.2588578
[55] Chantal Shaib, Millicent Li, Sebastian Joseph, Iain Marshall, Junyi Jessy Li, and Byron Wallace. 2023. Summarizing, Simplifying, and Synthesizing Medical Evidence using GPT-3 (with Varying Success). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Toronto, Canada, 1387–1407. https://aclanthology.org/2023.acl-short.119
[56] Sanja Štajner, Kim Cheng Sheang, and Horacio Saggion. 2022. Sentence Simplification Capabilities of Transfer-Based Models. *Proceedings of the AAAI Conference on Artificial Intelligence* 36, 11 (June 2022), 12172–12180. https://doi.org/10.1609/aaai.v36i11.21477
[57] P. P. Uhrowczik. 1973. Data Dictionary/Directories. *IBM Systems Journal* 12, 4 (1973), 332–350. https://doi.org/10.1147/sj.124.0332
[58] Chris van der Lee, Albert Gatt, Emiel van Miltenburg, and Emiel Krahmer. 2021. Human evaluation of automatically generated text: Current trends and best practice guidelines. *Computer Speech & Language* 67 (2021), 101151. https://doi.org/10.1016/j.csl.2020.101151
[59] Victor Vianu. 2021. Datalog Unchained. In *Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS'21)*. Association for Computing Machinery, New York, NY, USA, 57–69. https://doi.org/10.1145/3452021.3458815
[60] Li Wang, Xi Chen, XiangWen Deng, Hao Wen, MingKe You, WeiZhi Liu, Qi Li, and Jian Li. 2024. Prompt engineering in consistency and reliability with the evidence-based guideline for LLMs. *npj Digital Medicine* 7, 1 (2024), 41.
[61] Yisu Remy Wang, Mahmoud Abo Khamis, Hung Q. Ngo, Reinhard Pichler, and Dan Suciu. 2022. Optimizing Recursive Queries with Progam Synthesis. In *SIGMOD 2022*. Association for Computing Machinery, New York, NY, USA, 79–93. https://doi.org/10.1145/3514221.3517827
[62] Emily M Weitzenboeck, Pierre Lison, Malgorzata Cyndecka, and Malcolm Langford. 2022. The GDPR and unstructured data: is anonymization possible? *International Data Privacy Law* 12, 3 (03 2022), 184–206. https://doi.org/10.1093/idpl/ipac008 arXiv:https://academic.oup.com/idpl/article-pdf/12/3/184/45690911/ipac008.pdf