Datalog-based Reasoning for Banking Supervision: is This All a Fantasy?*

Luigi Bellomarini[1], Marta Bernardini[1], Livia Blasi[1,2], Costanza Catalano[1], Rosario Laurendi[1], Andrea Gentili[1], Davide Magnanimi[1,3], Marco Mastrogirolamo[1] and Emanuel Sallinger[2]

[1]*Banca d'Italia, Italy*
[2]*TU Wien, Austria*
[3]*Politecnico di Milano, Italy*

Abstract

Banking supervision is a pillar of economic stability, and rule-based AI systems promise to enhance its effectiveness through scalability and explainability. In this paper, we present a concrete case study from the Central Bank of Italy, where we developed and deployed KG4VIG, a rule-based framework built on the Datalog$^{\pm}$ family of languages and Knowledge Graphs. Focusing on the task of computing shareholding relationships in a financial network, we show how our declarative reasoning approach overcomes the limitations of matrix-based methods and provides an explainable and scalable solution.

Keywords

Knowledge graphs, reasoning, Datalog, Integrated Ownership, banking supervision.

1. Introduction and Contribution

Banking supervision is one of the most critical aspects for ensuring the continued functioning of our economic systems and society as a whole—as demonstrated by the devastating effects of some of the largest bank failures in history. The promises of information technology in general, and symbolic, rule-based AI systems in particular, to address these challenges are high. The main question we are going to answer in this paper is "is this all a fantasy?" and which points translate into reality, shown through a concrete approach in real-world use in the central bank space.

In an influential paper, Ehrlinger and Wöß [1] review many definitions of Knowledge Graphs (KG) and propose a formulation that distills a promising approach to put symbolic rule-based AI into action: "*[a KG] acquires and integrates information into an ontology and applies a reasoner to derive new knowledge*". Besides data integration and wrangling, ontologies are an essential ingredient: they shape the business meaning of the KG and enable *reasoning*, i.e., reading, analyzing and inferring the business concepts.

Still, how should we model ontologies? A prominent approach comes from the logic, database, and Knowledge Representation and Reasoning (KRR) communities, which have devoted increasing effort to express ontologies as rules in the Datalog$^{\pm}$ family [2] of languages, or "fragments". Datalog$^{\pm}$ extends the applicability of the Datalog language of databases [3] to reasoning tasks by supporting the features needed in this space, such as a full interplay between existential quantification and recursion. In the academic discourse, we see—and contribute to the study of—a proliferation of fragments, which aim at striking a balance between expressive power and computational complexity of reasoning, with different strategies incorporated into the execution engines. In this line of research, within a collaboration among the Central Bank of Italy, the University of Oxford, and TU Wien, namely, the *KG Labs*, we developed

RuleML+RR'25: Companion Proceedings of the 9th International Joint Conference on Rules and Reasoning, September 22–24, 2025, Istanbul, Turkiye.

*The views and opinions expressed in this paper are those of the authors and do not necessarily reflect the official policy or position of Banca d'Italia.

luigi.bellomarini@bancaditalia.it (L. Bellomarini); marta.bernardini@bancaditalia.it (M. Bernardini); livia.blasi@bancaditalia.it (L. Blasi); costanza.catalano@bancaditalia.it (C. Catalano); rosario.laurendi@bancaditalia.it (R. Laurendi); andrea.gentili@bancaditalia.it (A. Gentili); davide.magnanimi@bancaditalia.it (D. Magnanimi); marco.mastrogirolamo@bancaditalia.it (M. Mastrogirolamo); sallinger@dbai.tuwien.ac.at (E. Sallinger)

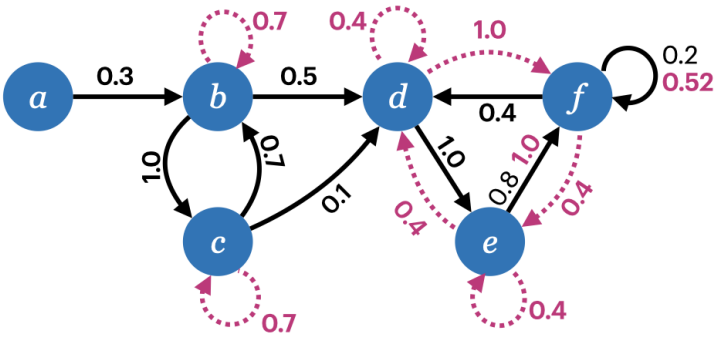

Figure 1: An ownership graph. Purple edges and weights refer to the IO in each SCC. When no double color is shown, the IO coincides with the edge weight, as for (b, c).

Vadalog [4], a Datalog-based reasoner based on *Warded Datalog*$^{\pm}$ [5], and gradually extended its ecosystem with features of practical utility such as algebraic operations and monotonic aggregations [6], conditions and equality-generating dependencies [7], and parallel scalable execution modes [8].

From the academic point of view, here we wonder: is this Datalog- rule-based ecosystem, stemming from decades of database and AI research, really actionable for real-world problems?

Contribution. In this paper, we demonstrate through an industrial case in the financial domain that rule-based reasoning with Vadalog and KGs can effectively address real-world challenges, offering scalable and explainable solutions. We present a perspective on Kg4Vig, a framework we developed and now in production at the Central Bank of Italy to support banking supervision.

Kg4Vig is a *banking supervision platform* with numerous features. In this paper, we focus on one specific task: providing analysts with the *Integrated Ownership* (IO) of a bank within a financial KG, where nodes represent financial entities (be they banks, companies, intermediaries, or people), and edges are shareholding relationships between them. The IO of an entity x over y is defined as the total ownership—both direct and indirect—that x holds on y, and serves as a measure of the influence of an entity over another, by quantifying the cumulative ownership involvement between them. While a matrix-based formulation of this problem exists, it suffers from computational limitations. We propose a declarative rule-based solution, formalized in Vadalog and implemented within Kg4Vig. We argue for the scalability and explainability of our technique for IO, and underscore the key role of tailor-made UI/UX components for successful rule-based applications.

Overview. In Section 2 we present the reasoning setting. In Section 3, we focus on the implementation and UI aspects. Section 4 concludes the paper.

2. The Industrial Case: Rule-based Integrated Ownership

Let $G = (V, E, \omega)$ be a graph that models the property relationships between financial entities—or companies hereinafter—where V is the set of nodes representing companies, $E \subset V \times V$ is the set of directed edges for ownership relationships, and $\omega : E \to [0, 1]$ is a weight function representing the share magnitude; an edge $e = (u, v)$ of weight $\omega(e)$ indicates that company u owns $\omega(e)\%$ of shares of v. An example of such property graph is in Figure 1 (only black edges and weights). The definition of IO relies on the concept of *Baldone walk* [9, 10]; a Baldone walk from A node u to A node v is a walk in which the starting node u may only appear as the first or (if $u = v$) the last node in the walk. The weight of a Baldone walk is the product of the weights of its edges. In Figure 1, the walk $\{a, b, c, b, c, b\}$ is a Baldone walk from a to b of weight $0.3 \times 1 \times 0.7 \times 1 \times 0.7 = 0.147$. Instead, the walk $\{b, c, b, d\}$ is not a Baldone walk. As shown in the example, Baldone walks may contain cycles. The integrated ownership $\mathcal{I}_{uv}$ between u and v is defined as the sum of the weights of all the Baldone walks from u to v. By setting W the adjacency matrix of G, $\mathcal{I} = \{\mathcal{I}_{uv}\}_{u,v \in V}$ and I the identity matrix, Glattfelder [11] shows that

$$\mathcal{I} = (\text{diag}((I - W)^{-1}))^{-1}(I - W)^{-1}W\,. \tag{1}$$

Unfortunately, applying Eq.(1) involves computing the inverse of the matrix $(I - W)$, where the best state-of-the-art algorithm doing so has computational complexity of $O(n^{2.372})$ [12], a computationally infeasible task in large real-world graphs like ours, having $\sim$ 182M nodes.

In KG4VIG, we solve the IO problem with a *graph factorization approach*: first, we compute the IO between all pairs of nodes within the same strongly connected component (SCC) with Eq. (1). Then, with the following VADALOG rules, we merge the SCC-local IOs and compute the IO between nodes in different SCCs.

$$\textit{Edge}(x, y, scc\_X, scc\_Y, w\_XY, w\_YY), sccX = sccY, \\ \rightarrow \textit{Own}(x, y, scc\_X, scc\_Y, w\_XY, w\_YY) \tag{2}$$

$$\textit{Edge}(x, y, scc\_X, scc\_Y, w\_XY, w\_YY), scc\_X \neq scc\_Y, \\ w = sum(w\_XY/(1.0 - w\_YY)) \rightarrow \textit{Own}(x, y, scc\_X, scc\_Y, w, w\_YY) \tag{3}$$

$$\textit{Edge}(x, z, scc\_X, scc\_Z, w\_XZ, w\_ZZ), \textit{Edge}(z, y, scc\_X, scc\_Y, w\_ZY, w\_YY), \\ scc\_X = scc\_Z, scc\_Z \neq scc\_Y, scc\_X \neq scc\_Y, \\ w = sum(w\_XZ * w\_ZY/(1.0 - w\_YY)) \rightarrow \textit{Own}(x, y, scc\_X, scc\_Y, w, w\_YY) \tag{4}$$

$$\textit{Own}(x, z, scc\_X, scc\_Z, w\_XZ, w\_ZZ), \textit{Edge}(z, y, scc\_X, scc\_Y, w\_ZY, w\_YY), \\ scc\_X \neq scc\_Y, scc\_X \neq scc\_Z, scc\_Y = scc\_Z, \\ w = sum(w\_XZ * w\_ZY) \rightarrow \textit{OwnD}(x, y, scc\_X, scc\_Y, w, w\_YY) \tag{5}$$

$$\textit{OwnD}(x, z, scc\_X, scc\_Z, w\_XZ, w\_ZZ), \textit{Edge}(z, y, scc\_X, scc\_Y, w\_ZY, w\_YY), \\ scc\_X \neq scc\_Y \neq scc\_Z \\ w = sum(w\_XZ * w\_ZY/(1.0 - w\_YY)) \rightarrow \textit{Own}(x, y, scc\_X, scc\_Y, w, w\_YY) \tag{6}$$

$$\textit{Own}(x, z, scc\_X, scc\_Z, w\_XZ, w\_ZZ), \textit{Edge}(z, y, scc\_X, scc\_Y, w\_ZY, w\_YY), \\ scc\_X \neq scc\_Y \neq scc\_Z, \\ w = sum(w\_XZ * w\_ZY/(1.0 - w\_YY)) \rightarrow \textit{Own}(x, y, scc\_X, scc\_Y, w, w\_YY) \tag{7}$$

The *Edge* predicate captures the identities of x and y, the SCC they belong to (scc_X and scc_Y), the weight w_XY, and the (SCC-locally computed) integrated ownership of y over itself $w\_YY = \mathcal{I}_{yy}$. If x and y belong to the same SCC, then w_XY is set equal to $\mathcal{I}_{xy}$, as by Eq. (1); if they belong to different SCCs, then $w\_YY = \omega(x, y)$, i.e., the ownership shares of x in y, if there are any. This is equivalent to adding edges between any pair of nodes (x, y) belonging to the same SCC with weight $\mathcal{I}_{xy}$ (see Figure 1, black *and* purple edges and weights). The contributions are later summed to get the integrated ownerships between every two companies.

When x and y belong to the same SCC (**Rule 2** - base case) the resulting Own's w simply equals w_XY itself. When x and y belong to different SCCs and are connected by an edge (**Rule 3** - base case), the resulting w of *Own* is computed by the sum of all w_XY normalized by $(1 - w\_YY)$. This normalization factor takes into account the sum of the weights of all walks from y to y itself in the graph, which in matrix terms is equal to $\sum_{i=0}^{+\infty} W_{yy}^{i} = (I - W)_{yy}^{-1}$; these walks are the ones that, concatenated with the edge (x, y), form the Baldone walks from x to y. Indeed, by massaging Eq.(1), it can be shown that $1/(1 - w\_YY) = 1/(1 - \mathcal{I}_{yy}) = (I - W)_{yy}^{-1}$. In Figure 1 this is the case for $x = c$ and $y = d$, which produces the contribution $w = 0.1/(1 - 0.4) = 0.1\bar{6}$ to the computation of $\mathcal{I}_{cd}$.

The last four rules are recursive cases of the first two. **Rule 4** states that if x and z belong to the same SCC (and hence there is an edge between the two of weight $w\_XZ = \mathcal{I}_{xz}$) and there is an edge from z to y where y belongs to a different SCC then the term w for *Own* between x and y is the sum of $w\_XZ * w\_ZY$ normalized by $(1 - w\_YY)$. In Figure 1 this is the case for $x = c$, $z = b$ and $y = d$, producing the contribution $w = (0.7 \times 0.5)/(1 - 0.4) = 0.5\bar{8}$ to the computation of $\mathcal{I}_{cd}$. Summing up the two contributions coming from Rule 3 and Rule 4 (as together they consider all the Baldone walks from c to d in the graph), we get that $\mathcal{I}_{cd} = 0.1\bar{6} + 0.5\bar{8} = 0.75$.

In **Rule 5**, a new predicate appears, *OwnD*, which is in turn used in Rule 6. Rule 5 tackles the case where an ownership path is computed between x and z belonging to different SCCs, and then z is

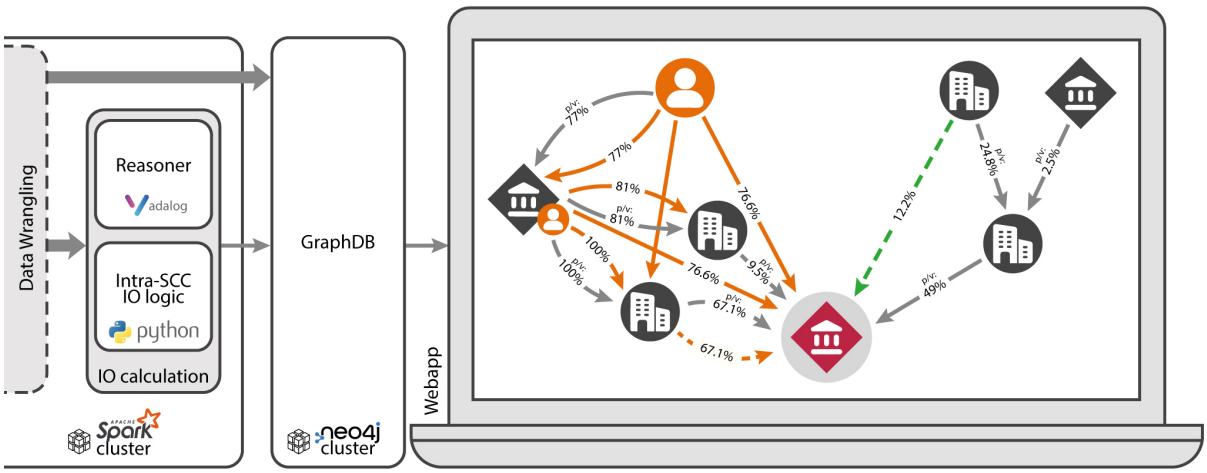

Figure 2: An IO-centered view on the architecture of KG4VIG: from data ingestion to web visualization.

connected to y in the same SCC; the term w for *OwnD* between x and y is the sum of $w\_XZ * w\_ZY$; this time it is not normalized, as by definition w_ZY is already equal to $\mathcal{I}_{zy}$. In Figure 1 this is the case for $x = a$, $z = b$ and $y = c$, producing $\mathcal{I}_{ac} = \mathcal{I}_{ab} \times 1 = 1 \times 1 = 1$. Since in our setting every pair of nodes is connected in a SCC, with Rule 5 we already cover all the nodes y belonging to the same SCC of z.

Therefore, the next step of the recursion must necessarily leave such SCC as otherwise it would re-visit nodes; this is the role of **Rule 6**. In it, all three nodes x, y, z must belong to different SCCs, with *OwnD* between x and z computed by Rule 5 and an edge from z to y; then the term w for *Own* between x and y is the sum of $w\_XZ * w\_ZY$ normalized by $(1 - w\_YY)$. In Figure 1, this is the case for $x = a$, $z = c$ and $y = d$, which produces $w = \mathcal{I}_{ac} \times 0.1/(1 - 0.4) = 1 \times 0.1/0.6 = 0.1\bar{6}$ to the computation of $\mathcal{I}_{ad}$.

Finally, **Rule 7** is as Rule 6 but with the predicate *Own* in input, covering the last case where all the three nodes x, z, y belong to different SCCs. In Figure 1 this is the case for $x = a$, $z = b$ and $y = d$, producing the contribution $w = \mathcal{I}_{ab} \times 0.5/(1 - 0.4) = 1 \times 0.5/0.6 = 0.8\bar{3}$ to the computation of $\mathcal{I}_{ad}$, that together with what computed by Rule 6 is equal to $\mathcal{I}_{ad} = 0.1\bar{6} + 0.8\bar{3} = 1$. Notice that $\mathcal{I}_{ab} = 0.3/(1 - 0.7) = 1$ is computed by Rule 3 in the base case.

Correctness and scalability. Our observation is that the IO between two nodes that belong to the same SCC $\mathcal{C}$ involves only walks in $\mathcal{C}$, i.e., the application of Eq. (1) can be restricted to the adjacency matrix $W_{\mathcal{C}}$ of $\mathcal{C}$. For example, in Figure 1, we have three SCCs: $\{a\}$, $\{b, c\}$ and $\{d, e, f\}$; the application of Eq. (1) to $\mathcal{C} = \{b, c\}$ produces $\mathcal{I}_{bb} = 0.7$, $\mathcal{I}_{bc} = 1$, $\mathcal{I}_{cb} = 0.7$ and $\mathcal{I}_{cc} = 0.7$ using $W_{\mathcal{C}} = [0, 1; 0.7, 0]$.

Our KG comprises $\sim$ 64k SCCs, whose largest component has $\sim$ 1k nodes, while all the others are under 50 nodes, making Eq.(1) efficiently computable in each SCC. Moreover, such computations are parallelizable among the SCCs as the calculations are independent of each other. The whole IO computation can be parallelized among weakly connected components of the graph, as they do not share any Baldone walk.

3. Implementation and Visualization

While this paper only focuses on a specific application perspective, we remark that the KG4VIG system is fully engineered and currently adopted in production in the Central Bank of Italy.

Implementation. An IO-centered view on its architecture is shown in Figure 2. A *data wrangling* component builds the KG from the enterprise data stores and materializes it into a *Neo4J cluster*.

In particular, they perform specific step in the data transformation pipeline that goes from improving the data quality and consistency (*data cleaning*) and harmonizing heterogenous data schemas into

a unified global one (*schema integration*), up to the construction of a unified graph view on domain entities and their relationships by integrating data from different data sources (*entity resolution* and *data fusion*). The database is deployed across three geographically distributed nodes to ensure load balancing, end-users' query performances, and fault tolerance.

VADALOG is used for data wrangling as well as the execution of business rules, like those of IO (Section 2). The rules are compiled into an abstract higher-level algebraic representation, namely, an *execution plan*, which is finally translated into a set of Spark Jobs. An underlying *Apache Spark Cluster* (16 cores, 32 threads, 256 GB of RAM each) executes these jobs, and the results are stored back into the KG. A dedicated web app provides access to the users. The data wrangling phase, as well as performance-critical tasks like IO, require the native use of *PySpark* and dedicated graph libraries such as *GraphFrames*, which provide primitives for distributed data processing within a Spark ecosystem.

To compute IO, all the SCCs are preliminarily identified using built-in methods from *GraphFrames* and then, they are transformed into a matrix representation, where Eq.(1) is solved in *Numpy*. The illustrated VADALOG program is then applied on the results from this phase. The reasoning engine retrieves and parses the ontology (i.e., set of rules) from its internal repository, applies a series of logic optimizations, and construct an access plan that maps each rule to a corresponding data transformation operation. This access plan is eventually compiled into a query plan tailored for the target execution environment. Within our framework, the reference execution environment is represented by the Apache Spark cluster; hence, the compiled query plan leverages Spark primitives to distribute the data processing workload across it. Finally, the computed IO from both stages are persisted back into the Neo4J graph database, thereby augmenting the KG with new inferred knowledge. These results are accessible to end-users through a tailored web application that facilitates the user interaction with the KG.

Notably, all the reasoning and Python computations are executed in parallel across SCCs, maximizing the use of cluster resources. The overall computational time for the IO is ~ 3 hours; notice that the use of Eq.(1) for the whole graph would make the machine to run out of memory space before termination.

Visualization. As rule-based developers and adopters, a tempting perspective lies in offering general-purpose solutions, with generic graphical metaphors for nodes and edges cooperating with *rule-based reasoning widgets* [13]. Yet, we learnt that a production-ready rule-based system calls for a tailor-made user interface, as domain experts wish for applications that can speak their business language and cater specifically to their business scenarios, while offering scalability and explainability.

Scalable logic fragments like Warded Datalog$^{\pm}$, supported by efficient execution engines, offer an extremely promising avenue. Thanks to the combination of KGs and rule-based reasoning, high interpretability of results can also be achieved. UIs can be hinged on the visual metaphor of *extensional vs intensional components*: the former shows the business data in the form of a property graph, carefully rendered with graph visualization libraries, the latter highlights the knowledge derived from the reasoning process. For example (Figure 2), the derived edges denoting IO and other business-relevant amounts, are shown in dashed line, while the extensional edges for the same measures (i.e., in the figure, the control edges, denoted in orange in both cases) are shown in solid line. The informative use of line styles, added to an accessible color palette to denote the different measures, as well as the careful use of shapes and symbols, facilitate the readability of the information shown by the application.

User interaction is also crucial: our approach consists in an incremental expansion of the currently visualized graph, which follows and mimics the rule application. The involving feeling of driving the analysis comes as a consequence. In the IO case, explainability is supported by interactive node exploration. The analyst may request a visualization of entities that directly or indirectly own the subject, with each step of the calculation made explicit as computed by VADALOG, trace ownership chains, and find the ultimately controlling bank. Side-tabs provide legends on colors and symbols, as well as information on the datasets of provenance and computed IO, reminding the user the computational principles underneath.

4. Conclusion

We presented a real-world application for the Central Bank of Italy, demonstrating the practical use of rule-based technology built on a state-of-the-art Datalog-based language and system. The successful deployment of such tools proves that their use is not a fantasy. While the gap between theory and practice, especially in the realm of logic programming, often seems wide, we showed that careful design choices and concrete experience in industrial contexts can bridge it effectively. The step-by-step inference mechanism of Datalog is already inherently explainable, but tight integrations with LLMs are starting to show the possibility to develop *GraphRAG* [14] approaches to build textual explanations of the reasoning conclusions [15], a goal for our future work.

Acknowledgments

This work was partially supported by the Vienna Science and Technology Fund (WWTF) [10.47379/VRG18013, 10.47379/NXT22018, 10.47379/ICT2201] and the Austrian Science Fund [10.55776/COE12].

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] L. Ehrlinger, W. Wöß, Towards a definition of knowledge graphs, in: SEMANTiCS, 2016.
[2] A. Calì, G. Gottlob, T. Lukasiewicz, B. Marnette, A. Pieris, Datalog$^{\pm}$: A family of logical knowledge representation and query languages for new applications, in: LICS, 2010.
[3] S. Abiteboul, R. Hull, V. Vianu, Foundations of databases, volume 8, 1995.
[4] L. Bellomarini, D. Benedetto, G. Gottlob, E. Sallinger, Vadalog: A modern architecture for automated reasoning with large knowledge graphs, Inf. Syst. 105 (2022) 101528.
[5] G. Gottlob, A. Pieris, Beyond SPARQL under OWL 2 QL entailment regime: Rules to the rescue, in: IJCAI, AAAI Press, 2015.
[6] A. Shkapsky, M. Yang, C. Zaniolo, Optimizing recursive queries with monotonic aggregates in deals, in: ICDE, IEEE Computer Society, 2015, pp. 867–878.
[7] L. Bellomarini, D. Benedetto, M. Brandetti, E. Sallinger, Exploiting the power of equality-generating dependencies in ontological reasoning, Proc. VLDB Endow. 15 (2022) 3976–3988.
[8] L. Bellomarini, D. Benedetto, M. Brandetti, E. Sallinger, A. Vlad, The vadalog parallel system: Distributed reasoning with Datalog+/-, Proc. VLDB Endow. 17 (2024) 4614–4626.
[9] S. Baldone, F. Brioschi, S. Paleari, Ownership measures among firms connected by cross-shareholdings and a further analogy with input-output theory, in: JAFEE, 1998.
[10] D. Magnanimi, M. Iezzi, Ownership graphs and reasoning in corporate economics, in: EDBT/ICDT Workshops, 2022.
[11] J. Glattfelder, Owership networks and corporate contol: mapping economic power in a globalized world, Phd thesis, ETH Zurich, 2010.
[12] J. Alman, R. Duan, V. V. Williams, Y. Xu, Z. Xu, R. Zhou, More asymmetry yields faster matrix multiplication, in: SODA, 2025, pp. 2005–2039.
[13] L. Bellomarini, M. Benedetti, A. Gentili, D. Magnanimi, E. Sallinger, KG-roar: Interactive Datalog-based reasoning on virtual knowledge graphs, VLDB 16 (2023) 4014–4017.
[14] Y. Hu, Z. Lei, Z. Zhang, B. Pan, C. Ling, L. Zhao, GRAG: graph retrieval-augmented generation, in: NAACL (Findings), ACL, 2025, pp. 4145–4157.
[15] A. Colombo, T. Baldazzi, L. Bellomarini, E. Sallinger, S. Ceri, Template-based explainable inference over high-stakes financial knowledge graphs, in: EDBT, 2025, pp. 503–515.