Towards Hybrid Logic-based and Embedding-based Reasoning on Financial Knowledge Graphs

Adriano Vlad[1,2], Sahar Vahdati[3], Mojtaba Nayyeri[4], Luigi Bellomarini[5] and Emanuel Sallinger[1,2]

[1]*University of Oxford, Department of Computer Science, Oxford, UK*
[2]*TU Wien, Faculty of Informatics, Vienna, Austria*
[3]*Institute for Applied Informatics (InfAI), Leipzig University, Leipzig, Germany*
[4]*University Of Bonn, Bonn, Germany*
[5]*Banca d'Italia, Italy*

Abstract
Warded Datalog+/- is a Datalog-based KRR language that guarantees decidability and tractability of the ontological reasoning task, thanks to its favourable theoretical properties. The Vadalog reasoning system exploits Warded Datalog+/- to provide a practical implementation of different reasoning tasks via basic isomorphism checks. However, these can be prohibitive in space and time complexity especially in the economic and financial context which is characterised by extreme-scale data stores and complex societal network dynamics. Recently, Knowledge Graph Embeddings (KGEs) have gained great interest in the scientific community and have extensively improved learning and knowledge discovery techniques. In this paper, we present and provide an experimental evaluation of Vada-ER, a framework that jointly uses logic-based reasoning and KGEs to provide a scalable alternative to basic isomorphism checks in ontological reasoning. With our work, we aim to improve the synergy between the reasoning and the embedding technologies and communities.

Keywords
Knowledge Graph Embeddings, Learning and Reasoning, Logic-based Reasoning, Embedding-based Reasoning, Datalog, Vadalog, Finance, Financial Knowledge Graphs

1. Introduction

Reasoning techniques are applied to a plethora of different problems in the computer science realm, especially in downstream tasks of Artificial Intelligence (AI). Languages of the Datalog$^\pm$ family [1] are used as a common thread across general-purpose systems for information integration and extraction, as well as data-oriented computing in general. With the increasing scale of information encountered today, new requirements on reasoning flexibility and scalability have emerged.

Financial Knowledge Graphs. The economic and financial context in particular are characterised by extreme-scale data stores and complex societal network dynamics. These can be modeled and captured using KGs and analysed with the power of automated reasoning. These graphs are central objects in corporate economics [2, 3, 4, 5] and are used by central banks, financial authorities and national statistical offices for banking supervision, creditworthiness evaluation, anti-money laundering, insurance fraud detection, economic and statistical research and more.

Ontological Reasoning. The ontological reasoning task, consisting in evaluating a query Q under a database D and a set of domain-describing constraints Σ, requires high scalability to co-exist with sufficient expressive power. Since the presence of existential quantification in Σ, essential for high expressive power, leads to undecidability of the reasoning tasks, multiple fragments of Datalog$^\pm$ arose, each adopting different syntactic restrictions to guarantee decidability and tractability. A particularly relevant fragment is Warded Datalog$^\pm$, where ontological reasoning is PTIME in data complexity [6].

Termination Strategies. From a theoretical point of view, the semantics of ontological reasoning tasks are operationally defined via the application of the CHASE procedure [7]. In order to answer Q, the chase extends D with new facts until all the constraints in Σ—and Q itself—are satisfied. The chase introduces new fresh symbols, *labelled (or marked) nulls*, as placeholders for the "objects" introduced by existential quantification. Practically, in the presence of recursion, the task could be non-terminating due to the possible introduction of infinitely many new labelled nulls. To cope with this, the state-of-the-art reasoners such as Vadalog [8] adopt so-called *termination strategies*, techniques that prevent the generation of facts that are superfluous to answer Q and yet could lead to non-termination.

Isomorphism Check. The key for guaranteeing termination of a chase-based procedure in reasoners is, in fact,

Published in the Workshop Proceedings of the EDBT/ICDT 2022 Joint Conference (March 29-April 1, 2022), Edinburgh, UK
adriano.vlad@gmail.com (A. Vlad); vahdati@infai.org (S. Vahdati); nayyeri@iai.uni-bonn.de (M. Nayyeri); luigi.bellomarini@bancaditalia.it (L. Bellomarini); sallinger@dbai.tuwien.ac.at (E. Sallinger)

defining points at which the chase may be terminated prematurely, by dynamically pruning derivation branches, while at the same time upholding correctness of the reasoning task. For instance, in Warded Datalog$^{\pm}$ the chase is restricted by isomorphism check, in the sense that when two facts have the same predicate name, same constants in the same positions and there exists a bijection between the labelled nulls—i.e., they are isomorphic—just one of them is created in the process, without loss of information for query answering. Yet, standard isomorphism check could be inefficient, due to the impractical memorisation of all the produced facts, in many situations prohibitive in space and time. In fact, there are attempts to limit the search space and so optimise memory usage, for instance, with Warded Datalog$^{\pm}$ it is possible to restrict the search of isomorphic copies to specific portions of the chase derivation graph, individuated by the fragment characteristics, namely the connected components of the *warded forest* [8]. However, isomorphism check and this form of local search cannot be effectively exploited with very high fragmentation of the warded forest, for example induced by the presence of many joins on extensional atoms [8].

Knowledge Graph Embeddings. Recently, Knowledge Graph Embedding models (KGEs) have gained great interest in the scientific community. Achieving state-of-the-art performance and accuracy, they are used for AI tasks such as link prediction, question answering, and recommendation systems. However, despite their potential, they are mainly employed for knowledge preparation learning rather than actual logic-based reasoning tasks. Although there still is a perceived disconnect between the two areas, injecting logical rules into embedding-oriented approaches seems promising [9, 10, 11] and beneficial in terms of both scalability and explainability [12].

Contribution. In this paper we aim to bridge logic and embeddings to enable faster reasoning over large scale datasets and provide the following contributions:

- We present Vada-ER, a framework which jointly uses logic-based reasoning and KGEs to achieve faster and more efficient query answering.
- We introduce the distance termination strategy which employs embeddings for chase termination purposes. We provide an experimental evaluation and show improvements in performance.
- We propose a chase graph generator methodology capturing both semantic and pattern features.
- We introduce chase graph embeddings to improve the traditional logic-based approach and enable embedding-based reasoning for a novel hybrid solution for query answering.

2. Preliminaries

In this section, we lay out the preliminary concepts of this work including: chase, chase graph, query answering, and knowledge graph embeddings.

Chase. The *chase* is considered among the fundamental algorithmic tools for a variety of database problems [13, 14, 15, 16]. We consider a set Σ of tuple-generating dependencies *TGDs* [16] of the form $\forall\bar{x}\forall\bar{y}(\phi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\psi(\bar{x},\bar{z}))$ where ϕ and ψ are conjunctions of atoms. Let us have a database $D = \{\mathcal{E}, \mathcal{R}, \mathcal{F}^D\}$, where $\mathcal{E} = \{e\}$ is a set of entities, $\mathcal{R} = \{r_n\}$ is a set of n-ary relations, and $\mathcal{F}^D = \{f = r_n(e_1,\dots,e_n) | r_n \in \mathcal{R}, e_i \in \mathcal{E}\}$ is a set of facts. The chase is a procedure $\mathcal{P}(\Sigma, \mathcal{D})$ that takes in input a database D and a set Σ of constraints and applies the TGDs until all of them are satisfied (if terminating) and possibly generates nulls (labelled/marked nulls) to satisfy existential quantification [14]. Each step of the chase can be referred to as *chase step*.

A *chase graph* [14, 15] is the representation of a chase procedure in the form of a graph. The nodes of the graph represent the facts $\mho = r_n(e_1,\dots,e_n)$, while the links represent the application of the rule, the dependency of the generated facts.

Query answering. Ontological query answering is one of the fundamental tasks in databases and knowledge representation. Given a set of existential rules Σ and an n-ary predicate *Ans*, the evaluation of a query $Q = (\Sigma, \textit{Ans})$ over a database D is defined as $Q(D) = \{\bar{t} \in \mathrm{dom}(D)^n \mid \mathrm{Ans}(\bar{t}) \in \Sigma(D)\}$. Therefore, it is the problem of answering queries with respect to both D and all the facts entailed from D via Σ [17].

Knowledge graph embeddings. Embedding-based approaches obtain low dimensional representation for symbolic data (e.g., vector representation of nodes and edges in a graph) and aim at preserving the characteristics of original representation. A Knowledge Graph (KG) is defined as $\mathcal{K} = \{\mathcal{E}, \mathcal{R}, \mathcal{F}\}$, where $\mathcal{E}, \mathcal{R}$ are a set of entities (e.g. *Bob, Apple*) and (2-ary) relations (e.g. *InvestorIn*) respectively, and $\mathcal{F} = \{r_2(h,t) = (h,r,t)\} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a set of triples/facts.

Embeddings: For a given graph, $\gtrapprox$ is a mapping that represents nodes as vectors in a d_e-dimensional space such that $\gtrapprox : \mathcal{E} \rightarrow V_e^{d_e}$, where V_e can be real $\mathbb{R}$, complex $\mathbb{C}$ etc. The embedding of an entity $e \in \mathcal{E}$ is noted in boldface form i.e. $\boldsymbol{e} = \gtrapprox(e)$ where $\boldsymbol{h}$ denotes the head and $\boldsymbol{t}$ the tail entities. Depending on the formulation of a KGE model, each relation $r \in \mathcal{R}$ can be represented as a vector $v : \mathcal{R} \rightarrow V^{d_r}$, matrix $v : \mathcal{R} \rightarrow V^{(d_r^1 \times d_r^2)}$, n-ary tensor $v : \mathcal{R} \rightarrow V^{d_r^1 \times d_r^2 \times \dots \times d_r^n}$, (where V can be real $\mathbb{R}$, complex $\mathbb{C}$, etc), or a function (e.g the weights of a neural network) $v : \mathcal{R} \rightarrow V^{d_r}$. We use $\theta = \{\theta_e, \theta_r\}$ (initially randomized) to represent the embeddings of all entities (θ_e) and relations (θ_r) in a KG. In case a relation is represented as a function, θ_r denotes the parameters of the function.

Other related work. There is, to the best of our knowledge, very little related work in the concrete area of using embeddings to improve performance of the chase. One example is [18]. For space reasons, in this short paper, we refer to [8] for related work on Datalog$^\pm$ in general, and to [19] for related work to embeddings.

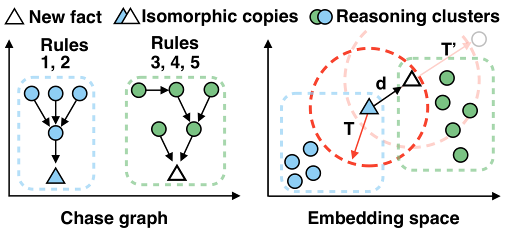

Figure 1: Distance Termination Strategy

3. The VADA-ER Framework

As discussed in the introduction, *termination strategies* are a key technique to ensure chase termination in Datalog$^\pm$ languages, such as Warded Datalog$^\pm$, when the chase starts producing isomorphic copies. We propose an alternative and novel termination strategy based on embedding distance instead of isomorphism check.

Distance Termination Strategy. Our embedding-based chase termination approach relies on the distance between fact embeddings (i.e. $dist(\boldsymbol{f_a}, \boldsymbol{f_b})$, $\boldsymbol{f_a}, \boldsymbol{f_b} \in \mathbb{R}^d$, $f_a, f_b \in \mathcal{F}$, where $\mathcal{F}$ is the set of all facts obtained during the chase) rather than basic isomorphism checks in ontological reasoning. Given a Vadalog program, our Vada-ER framework builds the full chase graph and learns fact embeddings $\boldsymbol{E} = \{\boldsymbol{f} \mid f \in \mathcal{F}\}$. These are trained to represent semantic and reasoning similarities. Let us explain the procedure with an example.

Example 1. *(isomorphic copies in chase) Program P contains set of entities, relations, facts $D = \{\mathcal{E}, \mathcal{R}, \mathcal{F}^{\mathcal{D}}\}$, rules Σ and a query Q.*

$\mathcal{E} = \{$*Apple*, *Bob*, *Sequoia*$\}$,

$\mathcal{R} = \{$*Company*, *InvestorFrom*, *InvestorIn*$\}$,

HasPitched, *SuccesfulPitch*$\}$,

$\mathcal{F}^D = \{$*Company*(*Apple*), *InvestorFrom*(*Bob*, *Sequoia*),

InvestorIn(*Bob*, *Apple*), *HasPitched*(*Apple*),

SuccesfulPitch(*Apple*)$\}$

$\Sigma = \{$

1 : *Company*(x), *InvestorFrom*(z, y),

InvestorIn$(z, x) \to$ *InvestmentOfFrom*(x, z, y).

2 : *InvestmentOfFrom*$(x, z, y) \to \exists k$ *Funding*(x, y, k).

3 : *Company*$(x) \to \exists y$ *MeetingWith*(x, y).

4 : *MeetingWith*(x, y), *HasPitched*$(x) \to$ *PitchedTo*(x, y).

5 : *PitchedTo*(x, y), *SuccesfulPitch*$(x) \to \exists k$ *Funding*(x, y, k).$\}$

$Q = \{$*Funding*$(x, y, z) \to$?.$\}$

In order to answer the query Q in this example, Vada-ER generates the query chase graph, sketched on the left-hand side of Figure 1. For each new fact f_{new} produced during the chase procedure, instead of isomorphism check, Vada-ER computes the embedding distance from every existing fact f and sets $d_{min} = \min_{f \in \mathcal{F}_i} dist(\boldsymbol{f}, \boldsymbol{f_{new}})$ (where $\mathcal{F}_i$ is the set of all facts in the ith iteration of the chase) as the shortest distance. If d_{min} is larger than a certain threshold T, f_{new} is generated and added to the chase graph. We observe that the first two rules lead to the generation of *Funding*(*Apple*, *Sequoia*, $k1$). At this point in the chase, Vada-ER checks if a similar fact was already generated by calculating the distance from the fact embedding of *Company*(*Apple*), *InvestorFrom*(*Bob*, *Sequoia*) and *InvestmentOfFrom*(*Apple*, *Bob*, *Sequoia*). The shortest distance from the already generated facts is still larger than the threshold and hence the fact can be added to the chase. However, *Funding*(*Apple*, *Sequoia*, $k2$) is produced in the connected component resulting from rules 3, 4, 5. Its embedding results are very close to the one produced in rule 2 as they have same predicate name and constants in the same positions and there is a bijection of the labelled nulls. Therefore, Vada-ER stops to avoid generating the duplicate.

To measure whether two fact embeddings are too close for termination purposes, Vada-ER employs the so called *Distance Threshold* which is defined below.

Definition 1. ***(Distance Threshold)*** *Given a Vadalog program $P = \{D, \Sigma\}$ and fact embeddings $\boldsymbol{E} = \{\boldsymbol{f} \in \mathbb{R}^d | f \in \mathcal{F}\}$ and given a fact f_{new} to be generated in the current chase step (i^{th} step), we define the* distance threshold T_i *as the minimal distance at which the closest fact to f_{new} in the embedding space $\boldsymbol{E}$ must be so that f_{new} is generated. Then f_{new} is generated when $dist(\boldsymbol{f_{new}}, \boldsymbol{f}) > T_i$ and dropped when $dist(\boldsymbol{f_{new}}, \boldsymbol{f}) \leq T_i$.*

Approximating chase trees. In order to decrease the probability of generating duplicates with the number of chase steps in our methodology, we propose to use a cumulative distance threshold which increases by Δ_i (the *distance threshold step*) in each iteration (i), i.e. $T_i = T_{i-1} + \Delta_i$. This technique allows us to decrease, at each chase step, the probability of producing new facts and obtain a homogeneous approximation of each chase tree. In order to update the distance threshold at each step, we propose the equation $T_i = T_0 + \sum_{n=1}^{i} e^{-n*T_0}$ where T_0 is the initial distance threshold, chosen as a parameter.

The traditional isomorphism check is performed between all facts in the same connected component in the warded forest [8] and, if this has too many connected components, this either leaves many isomorphic copies behind or has to be applied to the whole chase graph, which is prohibitive in cost. Using the embedding distance instead, Vada-ER can position facts in a multidimensional space that, if properly indexed, enables an efficient full check beyond the mere single connected component and guaranteeing low computational complexity, e.g. logarithmic in case of kd-trees.

Preprocessing Phase. Vada-ER carries out two preprocessing steps to generate the embeddings used in the distance termination strategy. These are the *Chase Graph Generator* and *Embedding Layer*. The *Chase Graph Generator* layer generates the chase graph for the execution of a given Vadalog program and a set of facts representing the underlying knowledge. This step itself includes the creation of a *semantic chase graph* and *pattern chase graph* which are then unified in a *hybrid chase graph*. The latter is to be fed to the *Embedding Layer* to learn fact embeddings used for the termination purposes explained above. Each of these steps is introduced below.

Semantic chase graph. The first step of the preprocessing phase is the *Chase Graph Generator* which constructs what we call the *hybrid chase graph*. In a first step, we take the standard chase graph CG (i.e., where facts are nodes and rule applications are edges) and enhance it with the following additional edges: for each rule application, all atoms in the body are also connected via edges. We call this the *semantic chase graph* SCG, given that it improves the semantic understanding of the reasoning process for the later embedding step.

Pattern chase graph. As this yields separate components of the graph for each ground instantiation, and most embedding methods are not able to learn the connections between such separate components automatically, we supplement it by the following *pattern chase graph* PCG. Two facts are pattern-isomorphic when they have the same predicate name and there is a bijection between the constant values and a bijection between the labelled nulls. Let f^c denote the isomorphism class of fact f. Then

$$PCG = \{(f_a^c, f_b^c) \mid (f_a, f_b) \in CG\}$$

Hybrid chase graph. We construct the *hybrid chase graph* HCG as the union of the semantic and pattern chase graphs, and in addition connect each node of the latter representing an isomorphism class with all of its instantiations in the semantic chase graph. That is, $HCG = SCG \cup PCG \cup CCG$ with

$$CCG = \{(f_a, f_a^c) \mid (f_a, \_) \in CG \vee (\_, f_a) \in CG\}$$

We call those additional edges CCG (as in connecting chase graph). We annotate each of the edges stemming from SCG, PCG and CCG with edge types r_S, r_P and r_C, respectively – this allows the embedding model to distinguish such edges.

Embedding Layer. The second step of the preprocessing phase is performed by the *Embedding Layer* which takes in input the hybrid chase graph HCG and returns a vector for each fact $\boldsymbol{E} = \{\boldsymbol{f} \mid (f, \_) \in HCG \vee (\_, f) \in HCG\}$ produced during the chase so that similar facts in terms of semantic and pattern features are closer in the embedding space. This is the prerequisite to successfully use embeddings for reasoning purposes. We obtain an embedding space in which nodes belonging to the same chase tree are close to each other, grouped in what we call *reasoning clusters*. For this purpose various suitable state-of-the-art embedding models are *DeepWalk* [20], *RotatE* [21] and *TransE* [22].

Vada-ER advantages. This methodology has several advantages compared to the usual termination strategies. It wins at run time, as the learning procedure is carried on during the preprocessing steps and the model used at run time. Moreover, our approach is not only capable of recognising two isomorphic facts but even finding semantic and pattern similarity between facts that may lead to isomorphism in following steps.

4. Experiments on Financial KGs

In our experiments, we measure Vada-ER execution time and recall using both real-world and complex synthetic Vadalog programs [8]. Among these are st-Connectivity, a well known program also in the financial context, and a synthetic program with which we test our approach in complex economic and financial settings. On the one hand, as the input to the st-Connectivity program we use a graph generated by means of the Barabasi Albert algorithm with node parameter equal to 100 and 2 attached edges per new node. These parameters lead to roughly 2 thousand nodes in output and 5.5 thousand edges. On the other hand, the synthetic program automatically generates its input and has more rules than the previous program, making it significantly more complex. It counts 27 recursive linear rules, 63 non-recursive linear rules, and 10 join rules. It has a prevalence of linear rules and 20% of the total rules have existential quantification. Moreover 30% of the linear and non-linear rules are recursive.

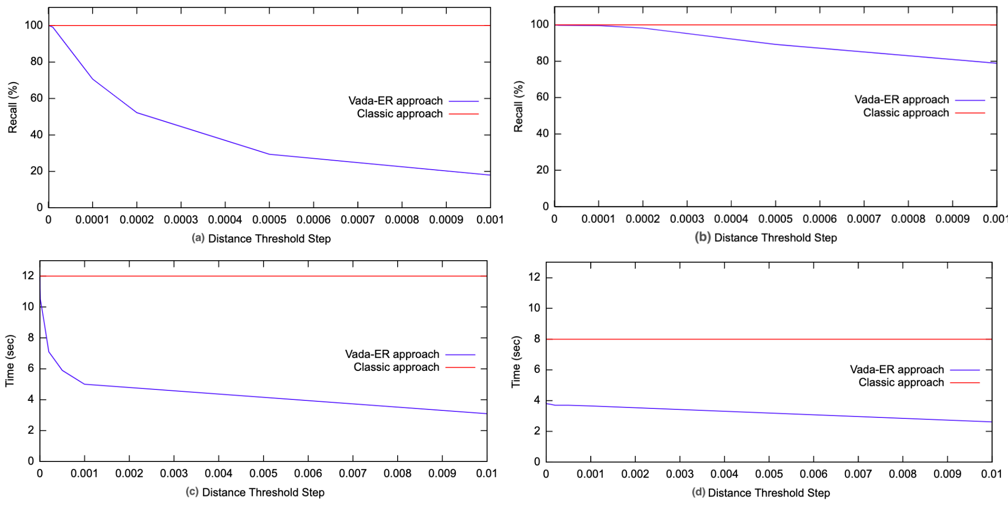

Figure 2: Impact of the distance threshold step on the distance termination strategy.

In these experiments, we investigate the impact of the threshold step, the characteristic parameter of our approach, on Vada-ER execution time. In this setting, the recall measures how many of the facts that should be generated are indeed generated and it is computed comparing Vada-ER execution with the traditional one that uses standard isomorphism check. The recall here is not always total because errors in the embedding model cause false matches, which can result in expected facts of the chase not being produced. The recall grows (blue line) with decreasing threshold steps Δ_i. In the synthetic scenario in Figure 2(a), Vada-ER achieves maximum recall for very small Δ_i. The same step in the real-world case in Figure 2(b) leads to maximum recall in one third of the time. The execution time (blue line) for the synthetic scenario in Figure 2(c) also grows slightly more than linearly with decreasing Δ, as smaller values correspond to allowing the generation of more nodes. Vada-ER achieves a 70% recall in two thirds of the execution time of a classic logic-based approach (red line) and maximum recall still remaining faster than traditional methods. The real-world scenario in Figure 2(d) shows even faster execution times. In summary, we see how the threshold adjusts the tradeoff between performance and recall: zero threshold means total recall, pure isomorphism check, low performance; high threshold means approximate check, lower recall, high performance.

5. Conclusion

This work has taken one foundational step towards bridging logic-based and embedding-based reasoning, by answering the question: can embedding-based reasoning speed up logic-based reasoning, and at what cost? We proposed Vada-ER, a hybrid reasoning framework deployed in the Vadalog system having at its core the injection of logical theories into embedding-oriented approaches (via the chase) and vice versa (via termination strategies). Our novel approach improves the chase termination strategies of logic-based KG reasoning systems using ML-techniques, achieving both scalability and explainability, fundamental while reasoning over complex financial KGs.

References

[1] A. Calì, G. Gottlob, A. Pieris, Advanced processing for ontological queries, Proc. VLDB Endow. 3 (2010) 554–565.

[2] F. Barca, M. Becht, The control of corporate Europe, OUP Oxford, 2001.

[3] A. Chapelle, A. Szafarz, Controlling firms through the majority voting rule, Physica A: Statistical Mechanics and its Applications 355 (2005) 509–529.

[4] J. B. Glattfelder, Ownership networks and corporate control: mapping economic power in a globalized world, Ph.D. thesis, ETH Zurich, 2010.

[5] A. Romei, S. Ruggieri, F. Turini, The layered structure of company share networks, in: 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), IEEE, 2015, pp. 1–10.

[6] G. Gottlob, A. Pieris, Beyond SPARQL under OWL 2 QL entailment regime: Rules to the rescue, in: IJCAI, 2015, pp. 2999–3007.

[7] D. Maier, A. O. Mendelzon, Y. Sagiv, Testing impli-

cations of data dependencies, ACM Transactions on Database Systems 4 (1979) 455–468.

[8] L. Bellomarini, G. Gottlob, E. Sallinger, The Vadalog system: Datalog-based reasoning for knowledge graphs, arXiv preprint arXiv:1807.08709 (2018).

[9] S. Guo, Q. Wang, L. Wang, B. Wang, L. Guo, Jointly embedding knowledge graphs and logical rules, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, pp. 192–202.

[10] V. T. Ho, D. Stepanova, M. H. Gad-Elrab, E. Kharlamov, G. Weikum, Rule learning from knowledge graphs guided by embedding models, in: International Semantic Web Conference, Springer, 2018, pp. 72–90.

[11] J. Zhang, J. Li, Enhanced knowledge graph embedding by jointly learning soft rules and facts, Algorithms 12 (2019) 265.

[12] F. Yang, Z. Yang, W. W. Cohen, Differentiable learning of logical rules for knowledge base reasoning, in: NIPS, 2017, pp. 2319–2328.

[13] A. V. Aho, Y. Sagiv, J. D. Ullman, Efficient optimization of a class of relational expressions, ACM Transactions on Database Systems (TODS) 4 (1979) 435–454.

[14] A. Calì, G. Gottlob, M. Kifer, Taming the infinite chase: Query answering under expressive relational constraints, JAIR 48 (2013) 115–174.

[15] R. Fagin, P. G. Kolaitis, R. J. Miller, L. Popa, Data exchange: semantics and query answering, Theoretical Computer Science 336 (2005) 89–124.

[16] T. Gogacz, J. Marcinkowski, A. Pieris, All-instances restricted chase termination, in: Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, 2020, pp. 245–258.

[17] A. Calì, G. Gottlob, A. Pieris, Towards more expressive ontology languages: The query answering problem, Artificial Intelligence 193 (2012) 87–128.

[18] P. Atzeni, L. Bellomarini, M. Iezzi, E. Sallinger, A. Vlad, Augmenting logic-based knowledge graphs: The case of company graphs, in: KR4L, 2021.

[19] P. Atzeni, L. Bellomarini, M. Iezzi, E. Sallinger, A. Vlad, Weaving enterprise knowledge graphs: The case of company ownership graphs, in: EDBT, 2020.

[20] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 701–710.

[21] Z. Sun, Z.-H. Deng, J.-Y. Nie, J. Tang, Rotate: Knowledge graph embedding by relational rotation in complex space, arXiv preprint arXiv:1902.10197 (2019).

[22] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Advances in neural information processing systems, 2013, pp. 2787–2795.