Datalog-based Reasoning for Knowledge Graphs

Luigi Bellomarini[1,2], Georg Gottlob[1,3], and Emanuel Sallinger[1,3]

[1] University of Oxford
[2] Banca d'Italia
[3] TU Wien

Abstract. This is a short abstract based on the recently published VLDB 2018 paper [6] – the main technical paper describing the Vadalog system. It describes the central motivations and gives useful pointers on the architecture and the main algorithms.

Introduction and motivation. The importance of capitalizing and exploiting corporate knowledge has been clear to decision makers since the late 1970s, when this idea was gradually made concrete in the context of *expert systems*, software frameworks able to harness such knowledge and provide answers to structured business questions. Through *deductive database systems* – database systems that have advanced reasoning capabilities – and in particular the language *Datalog*, the area became of high interest to the database community in the 1980s and 1990s. Nevertheless, even though the importance of harnessing knowledge certainly has grown steadily since then, culminating in today's desire of companies to exploit *knowledge graphs*, the interest in deductive databases has faltered due to immature technology and overly complicated KRR formalisms.

Yet, we are currently assisting to a resurgence of *Datalog* in academia and industry [1, 4, 7–10]: companies like LogicBlox have proven that a fruitful exchange between academic research and high-performance industrial applications can be achieved based on Datalog [1], and companies like LinkedIn have shown that the interest in Datalog permeates industry [16]. Meanwhile, the recent interest in machine learning brought renewed visibility to AI, raising interest and triggering investments in thousands of companies world-wide, which suddenly wish to collect, encapsulate and exploit their corporate knowledge in the form of a knowledge graph.

In this context, it has been recognized that to handle the complex knowledge-based scenarios encountered today, such as reasoning over large knowledge graphs, Datalog has to be extended with features such as existential quantification. Yet, Datalog-based reasoning in the presence of existential quantification is in general undecidable. Many efforts have been made to define decidable fragments. Warded Datalog$^{\pm}$ is a very promising one, as it captures PTIME complexity while allowing ontological reasoning. Yet, so far, no implementation of Warded Datalog$^{\pm}$ was available.

We recently introduced the Vadalog system, a Datalog-based system for performing complex logic reasoning tasks, such as those required in advanced knowledge graphs; it is Oxford's contribution to the VADA research programme [18, 13], a joint effort of the universities of Oxford, Manchester and Edinburgh and around 20 industrial partners such as Facebook, BP, and the NHS (UK national health system). The Vadalog system proposes the first implementation of Warded Datalog$^{\pm}$, a high-performance Datalog$^{\pm}$ system utilising an aggressive termination control strategy. In this paper, we summarise the key aspects of the Vadalog system, while pointing to the original technical paper for the details and a comprehensive experimental evaluation.

Reasoning over knowledge graphs. The term *knowledge graph* (KG) has no standard definition. It can be seen as referring only to Google's Knowledge Graph, to triple-based models, or to multi-attributed graphs, which represent n-ary relations [15, 17]. As shown by Krötzsch [14], in order to support rule-based reasoning on such data structures, it is sometimes necessary to use tuples of arity higher than three at least for intermediate results. In this paper, we adopt a general notion of KGs by allowing relations of arbitrary arity, to support all of these models and modes of reasoning.

Example 1. An example of a simple knowledge graph reasoning setting is given in [15]:

$$\text{Spouse}(x, y, \mathit{start}, \mathit{loc}, \mathit{end}) \rightarrow \text{Spouse}(y, x, \mathit{start}, \mathit{loc}, \mathit{end})$$

This rule expresses that when a person x is married to a person y at a particular location, starting date and end date, then the same holds for y and x. That is, the graph of persons and their marriage relations is symmetric.

As stated in [15], most modern ontology languages are not able to express this example. Beyond this simple example, there are numerous requirements for a system that allows ontological reasoning over KGs. Navigating graphs is impossible without powerful recursion; ontological reasoning is impossible without existential quantification in rule heads [12]. Yet reasoning with recursive Datalog is undecidable in the presence of existential quantification, so some tradeoffs have to be accepted. While a comprehensive analysis of various requirements was given in [5], let us isolate three concrete characteristics for a language that supports reasoning over KGs:

1. **Recursion over KGs**. Should be at least able to express full recursion and joins, i.e., should at least encompass Datalog. Full recursion in combination with arbitrary joins allows to express complex reasoning tasks over KGs. Navigational capabilities, empowered by recursion, are vital for graph-based structures.
2. **Ontological Reasoning over KGs**. Should at least be able to express SPARQL reasoning under the OWL 2 QL entailment regime and set semantics. OWL 2 QL is one of the most adopted profiles of the Web Ontology Language, standardized by W3C.
3. **Low Complexity**. Reasoning should be tractable in data complexity. This is a minimal requirement for allowing scalability over large volumes of data.

Beyond these specific requirements, the competition between powerful recursion, powerful existential quantification and low complexity has spawned fruitful research throughout the community to address the mentioned Datalog undecidability in the presence of existential quantification. This has been done under a number of different names, but which we shall here call *Datalog*$^{\pm}$, the "+" referring to the additional features (including existential quantification), the "-" to restrictions that have to be made to obtain decidability. Many languages within the Datalog$^{\pm}$ family of languages have been proposed and intensively investigated [2, 3, 7–10]. Depending on the syntactic restrictions, they achieve a different balance between expressiveness and computational complexity.

Figure 1 gives an overview of the main Datalog$^{\pm}$ languages. In fact, most of these candidates, including Linear Datalog$^{\pm}$, Guarded Datalog$^{\pm}$, Sticky Datalog$^{\pm}$ and Weakly Sticky Datalog$^{\pm}$ do not fulfil (1). Datalog itself does not fulfil (2). Warded and Weakly Frontier Guarded Datalog$^{\pm}$ satisfy (1) and (2), thus are expressive enough. However,

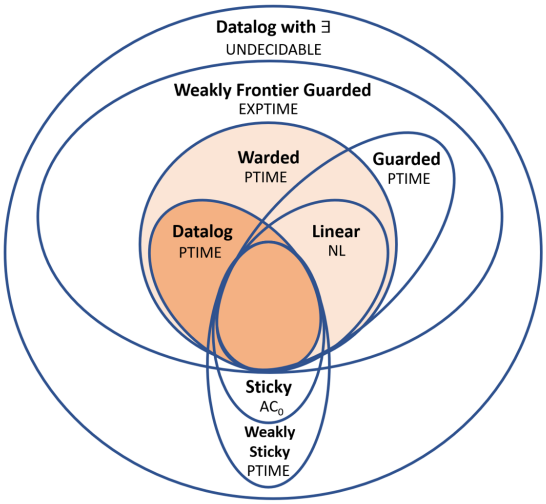

Fig. 1: Syntactic containment of Datalog$^{\pm}$ languages. Annotations (non-bold) denote data complexity. All names that do not explicitly mention Datalog refer to the respective Datalog$^{\pm}$ languages. E.g., "Sticky" refers to "Sticky Datalog$^{\pm}$".

the expressiveness of Weakly Frontier Guarded Datalog$^{\pm}$ comes at the price of it being EXPTIME-complete [3]. Thus it does not fulfil (3). Thus, in total, the only known language that satisfies (1), (2) and (3) is Warded Datalog$^{\pm}$. Yet, while Warded Datalog$^{\pm}$ has very good theoretical properties, the algorithms presented in [12] are alternating-time Turing machine algorithms, far away from a practical implementation.

Before summarising the characteristics of the Vadalog system, let us propose an example of a Datalog program, which we consider representative of the complexity of a typical reasoning setting involving KGs. The example is at the same time close to a classical scenario [11] as well as relevant for an application of the KG of one of our current industrial partners; in particular, for detecting specific risks related to issuers or guarantors for funds that are non-eligible due to their financing arrangement: in Europe there is prohibition on guaranteed debt instruments issued by a "closely-linked entity".

Example 2. Consider a set of rules about ownership relationships among a large number of companies. The extensional knowledge is stored in a database in the form of tuples of a relation Own($comp_1$, $comp_2$, w): company $comp_1$ directly owns a fraction w of company $comp_2$, with $0 \leq w \leq 1$. In addition, with a set of two rules, we intensionally represent the concept of *company control* (also studied in [11]): a company x controls a company y if company x directly owns more than half of the shares of company y or if x controls a set S of companies that jointly own more than half of y. Note that the `msum` operator, i.e., aggregation in the form of *monotonic sum*, is not a standard feature of Datalog, but an extension that some Datalog-based systems support [6].

$$\text{Control}(x, x).$$
$$\text{Own}(x, y, w), w > 0.5 \rightarrow \text{Control}(x, y)$$
$$\text{Control}(x, y), \text{Own}(y, z, w), v = \texttt{msum}(w, \langle y \rangle), v > 0.5 \rightarrow \text{Control}(x, z).$$

Here, for fixed x, the aggregate construct $\texttt{msum}(w, \langle y \rangle)$ forms the sum over all values w such that for some company y, $\text{Control}(x, y)$ is true, and $\text{Own}(y, z, w)$ holds, i.e., company y directly owns fraction w of company z. Now, with the described knowledge graph, many questions can be asked, such as: (i) obtain all the pairs (x, y) such that company x controls company y; (ii) which companies are controlled by x? Which companies control x? (iii) does company x control company y?

The Vadalog system. The Vadalog system is built around the Vadalog language, with Warded Datalog$^{\pm}$ as its logical core. It is currently used as the core deductive database system of the overall *Vadalog Knowledge Graph Management System* described in [5], as well as at various industrial partners, including the finance, security, and media intelligence industries. The **main contributions** are:

- A **novel analysis** of Warded Datalog$^{\pm}$, which culminates in the **first practical algorithm for Warded Datalog$^{\pm}$**. In the Vadalog system, the core principle behind the reasoning process is performing an *aggressive termination control*, which amounts to adopting dynamic programming strategies to preempt the generation of already explored areas of the KG as well as isomorphic ones, guaranteeing at the same time termination (as the fragment is decidable) and efficiency (as all the "informative" KG areas can be explored in PTIME). To achieve this goal, it is essential to recognise these KG areas as early as possible.
  While the naïve solution would imply a full caching of all the generated facts, the challenge here is guaranteeing limited memory footprint. We identify a number of *guide data structures* that closely exploit the underpinnings of Warded Datalog$^{\pm}$ and allow to abstract large KG areas by means of single representative facts (or patterns thereof). More in particular, we propose structures that play a complementary role for exploiting the periodicity of the KG: *warded forest*, which actually allows *vertical pruning* of isomorphic trees based on root isomorphism, and *(lifted) linear forest*, which allows *horizontal pruning* of entire pattern-isomorphic trees.
- A **system and architecture** that implements this algorithm in a relational database-inspired operator pipeline architecture. The pipeline's operators rely on *termination strategy wrappers* which transparently prevent the generation of facts that may lead to non-termination while ensuring the correctness of the result. The system is completed by a wide range of standard and novel optimisation techniques such as the dynamic construction of in-memory indices, stream-optimized "slot machine" joins, monotonic aggregation, materialization points, etc.
- The technical paper also presents a full-scale **experimental evaluation** of the Vadalog system on a variety of real-world and synthetic scenarios that thoroughly validate the effectiveness of our techniques on Warded Datalog$^{\pm}$ in absolute terms and comparatively with the top existing systems, which are outperformed by our reasoner.

Acknowledgements. This work is supported by the EPSRC programme grant VADA EP/M025268/1, the Vienna Science and Technology Fund (WWTF) grant VRG18-013, and the EU Horizon 2020 grant 809965.

References

1. M. Aref, B. ten Cate, T. J. Green, B. Kimelfeld, D. Olteanu, E. Pasalic, T. L. Veldhuizen, and G. Washburn. Design and implementation of the LogicBlox system. In *SIGMOD*, pages 1371–1382, 2015.
2. J. Baget, M. Leclère, and M. Mugnier. Walking the decidability line for rules with existential variables. In *KR*. AAAI Press, 2010.
3. J. Baget, M. Mugnier, S. Rudolph, and M. Thomazo. Walking the complexity lines for generalized guarded existential rules. In *IJCAI*, pages 712–717. IJCAI/AAAI, 2011.
4. P. Barceló and R. Pichler, editors. *Datalog in Academia and Industry - Second International Workshop, Datalog 2.0, Vienna, Austria, September 11-13, 2012. Proceedings*, volume 7494 of *Lecture Notes in Computer Science*. Springer, 2012.
5. L. Bellomarini, G. Gottlob, A. Pieris, and E. Sallinger. Swift logic for big data and knowledge graphs. In *IJCAI*, pages 2–10, 2017.
6. L. Bellomarini, E. Sallinger, and G. Gottlob. The vadalog system: Datalog-based reasoning for knowledge graphs. *PVLDB*, 11(9):975–987, 2018.
7. A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. *J. Artif. Intell. Res.*, 48:115–174, 2013.
8. A. Calì, G. Gottlob, and T. Lukasiewicz. A general Datalog-based framework for tractable query answering over ontologies. *J. Web Sem.*, 14:57–83, 2012.
9. A. Calì, G. Gottlob, T. Lukasiewicz, B. Marnette, and A. Pieris. Datalog+/-: A family of logical knowledge representation and query languages for new applications. In *LICS*, pages 228–242, 2010.
10. A. Calì, G. Gottlob, and A. Pieris. Towards more expressive ontology languages: The query answering problem. *Artificial Intelligence*, 193:87–128, 2012.
11. S. Ceri, G. Gottlob, and L. Tanca. *Logic programming and databases*. Springer, 2012.
12. G. Gottlob and A. Pieris. Beyond SPARQL under OWL 2 QL entailment regime: Rules to the rescue. In *IJCAI*, pages 2999–3007, 2015.
13. N. Konstantinou, M. Koehler, E. Abel, C. Civili, B. Neumayr, E. Sallinger, A. A. A. Fernandes, G. Gottlob, J. A. Keane, L. Libkin, and N. W. Paton. The VADA architecture for cost-effective data wrangling. In *SIGMOD Conference*, pages 1599–1602. ACM, 2017.
14. M. Krötzsch. Efficient rule-based inferencing for OWL EL. In *IJCAI 2011*, pages 2668–2673, 2011.
15. M. Marx, M. Krötzsch, and V. Thost. Logic on MARS: ontologies for generalised property graphs. In *IJCAI 2017*, pages 1188–1194, 2017.
16. W. E. Moustafa, V. Papavasileiou, K. Yocum, and A. Deutsch. Datalography: Scaling datalog graph analytics on graph processing systems. In *BigData*, pages 56–65. IEEE, 2016.
17. J. Urbani, C. J. H. Jacobs, and M. Krötzsch. Column-oriented datalog materialization for large knowledge graphs. In *AAAI 2016*, pages 258–264, 2016.
18. VADA. Project. `http://vada.org.uk/`, 2016.