A Datalog-Based Language for Querying RDF Graphs*

Marcelo Arenas[1], Georg Gottlob[2], and Andreas Pieris[3]

[1] Pontificia Universidad Católica de Chile marenas@ing.puc.cl
[2] Department of Computer Science, University of Oxford georg.gottlob@cs.ox.ac.uk
[3] Institute of Information Systems, Vienna University of Technology
pieris@dbai.tuwien.ac.at

RDF is the W3C recommendation data model to represent information about World Wide Web resources, while SPARQL is the standard language for querying RDF data, since its standardization in 2008. One of the distinctive features of Semantic Web data is the existence of vocabularies with predefined semantics: the *RDF Schema (RDFS)* and the *Web Ontology Language (OWL)*, which can be used to derive logical conclusions from RDF graphs; hence, an RDF query language equipped with reasoning capabilities to deal with these vocabularies is desirable. In addition, navigational capabilities are vital for data models with an explicit graph structure such as RDF [1, 3, 9, 15], while recursive definitions are a key feature for graph query languages [5, 14]. Having an RDF query language available that combines the above key functionalities is of paramount importance for the development of the Semantic Web. This has been recognized by the W3C, which led to the release of SPARQL 1.1 in 2013 [10, 12], that is, an extended version of the 2008 language with reasoning capabilities to deal with RDFS and OWL vocabularies, and a mechanism to express navigation patterns through regular expressions. However, there are still useful queries that cannot be expressed in SPARQL 1.1, due to the lack of general recursion [14].

To the best of our knowledge, the only language that supports the above features, focussing on the profile OWL 2 QL of OWL 2, while its query evaluation problem is tractable in data complexity, is the recently introduced rule-based language TriQ-Lite, the lite version of the highly expressive triple query (TriQ) language [2]. This language is based on Datalog$^{\exists,\neg s,\bot}$, that is, Datalog extended with existential quantification in rule-heads, stratified negation, and negative constraints with the falsum ($\bot$) in rule-heads. Unfortunately, TriQ-Lite suffers from a serious drawback, which may revoke its advantage as an expressive RDF query language, namely it is not a *plain* language. A query language is called plain if it allows the user to write a query as a single program in a simple non-composite syntax. An example of a plain query language is Datalog, where the user simply needs to define a single Datalog program that captures the intended query. The property of plainness provides conceptual simplicity, which is considered to be a key condition for a query language to be useful in practice. Although TriQ-Lite is based on an extension of Datalog, the way its syntax and semantics are defined significantly deviates from the standard way of defining Datalog-like languages, and thus does not inherit the plainness of Datalog. In fact, TriQ-Lite is a composite language, where the user is forced to split the query in several modules $\Pi_1, \ldots, \Pi_n$ so that each Π_i can be expressed by the fragment of Datalog$^{\exists,\neg s,\bot}$ that is underlying TriQ-Lite, while each pair (Π_i, Π_{i+1}) is bridged via a set Q_i of conjunctive queries.

* This short paper is based on the recent works [2, 11].

From the above discussion, we conclude that an RDF query language that fulfills certain desiderata, which in turn guarantee its applicability in real Semantic Web applications, is currently missing. These desiderata are the following:

1. **Plainness:** simple syntax and semantics, with the aim of simplifying the definition of queries;
2. **Reasoning Capabilities:** express every SPARQL query under the entailment regime for OWL 2 QL;
3. **Recursive Definitions:** general form of recursion must be supported, and ideally Datalog must be incorporated;
4. **Efficiency:** query evaluation must be data tractable, and feasible by the use of standard database technology.

At this point, we would like to expose an additional (conceptual) shortcoming of SPARQL 1.1, which must be taken into account during the designing of an RDF query language. Under the OWL 2 direct semantics entailment regime, the evaluation of a basic graph pattern over an RDF graph adopts the so-called active domain semantics, i.e., it uses the notion of entailment in OWL 2 QL, but allowing variables and blank nodes to take only values from the RDF graph. As discussed in [2], this forces the user to encode part of the reasoning in the actual query, which undoubtedly leads to unnatural and complex queries. This is illustrated in the following example:

Example 1. Consider the OWL 2 QL ontology $\mathcal{O}$ which states that Tom is a person and each person has a father, and let G be the RDF graph that represents $\mathcal{O}$. Assume that we want to retrieve the elements of G that have a father. One may be tempted to claim that this query can be expressed via the graph pattern $P = (?X, \text{father}, B)$, where B is a blank node. However, the answer to P over G is empty since there are no elements a, b in G such that the triple (a, father, b) is implied by the ontology. To obtain the expected answer, we have to consider the graph pattern $(?X, \text{rdf:type}, \exists\text{father})$, which means that we are forced to implicitly encode the fact that the triple $(\text{Tom}, \text{rdf:type}, \exists\text{father})$ is inferred by the ontology. ∎

Notably, TriQ-Lite provides the definition of the more natural entailment regime without the active domain restriction. This is certainly an additional desideratum:

5. **Reasoning-Query Decoupling:** the entailment regime without the active domain restriction must be definable in order to decouple the reasoning from the query.

In this work, we focus on TriQ-Lite, which is a language in evolution, and we investigate how it can be transformed into a plain language without sacrificing any of the other desiderata. The outcome of our study is TriQ-Lite 1.0, the new version of TriQ-Lite, which is based on Datalog$^{\exists,\neg sg,\bot}$; we use the superscript $\neg sg$ (instead of $\neg s$) since, for our purposes, it suffices to focus on negation that, apart from being stratified, is also grounded, i.e., it can be used with predicates that can only store constants. The proposed formalism is part of Datalog$^{\pm}$, that is, a family of logical KR languages [7]. Although several interesting Datalog$^{\pm}$ languages can be found in the literature (see, e.g., [4, 6, 8, 13, 16]), none of them fulfills all desiderata. Even though Datalog$^{\pm}$ languages inherit the plainness of Datalog, either they are not expressive enough for satisfying desiderata 2, 3 and 5, or very expressive and thus intractable. Hence, our key

Desideratum	How it is Achieved
Plainness	Standard Datalog-like language
Reasoning capabilities	$\exists$-quantification and $\bot$ in rule-heads
Recursive definitions	Incorporate full Datalog$^{\neg s}$
Efficiency	Reduction to UCQ evaluation
Reasoning-Query decoupling	Expressive joins in rule-bodies, and $\exists$-quantification in rule-heads

Table 1. TriQ-Lite 1.0.

challenge was to define a Datalog$^{\pm}$ language that achieves the right balance between expressivity and complexity.

Contribution. We propose a new syntactic paradigm, which is underlying TriQ-Lite 1.0, called *wardedness*, that can be informally described as follows: all the *dangerous* body-variables, i.e., variables that may be bound by the program to non-constant values, and at the same time are propagated to the rule-head, occur in exactly one body-atom, called *ward*, that can interact with the rest of the rule-body only via harmless join variables, i.e., variables that are bound by the program to database constants. For example, the program Π, where the dangerous variables are the variables marked by tilde,

$$\begin{aligned} \mathit{triple}(\tilde{X}, U, \tilde{Y}), \mathit{inv}(U, V) &\rightarrow \mathit{triple}(\tilde{Y}, V, \tilde{X}) \\ \mathit{type}(\tilde{X}, Y), \mathit{rest}(Y, U) &\rightarrow \exists Z\, \mathit{triple}(\tilde{X}, U, Z) \\ \mathit{triple}(\tilde{X}, U, Y), \mathit{rest}(Z, U) &\rightarrow \mathit{type}(\tilde{X}, Z) \end{aligned}$$

is warded; for each rule, all the tilde variables occur in a single body-atom $\underline{a}$, while all the other variables of $\underline{a}$ are harmless.

Our technical results can be summarized as follows (for details see [11]):

- We introduce TriQ-Lite 1.0, which is based on warded Datalog$^{\exists,\neg sg,\bot}$, and show that it fulfills all desiderata; the technical reasons are given in Table 1.
- We show that for the reasoning-query decoupling, apart from expressive joins among body-variables that are bound to non-constant values (this exposes one of the main limitations of the language underlying TriQ-Lite), it is vital to allow for existentially quantified variables in rule-heads.
- We provide a formal justification for the necessity of introducing a new Datalog-based formalism. We establish, via a model-theoretic argument, that existing tractable formalisms are not able to encode the OWL 2 direct semantics entailment regime, and at the same time ensure the reasoning-query decoupling.
- Finally, we substantiate the design choices made in the definition of our formalism. In fact, we show that very mild extensions lead to EXPTIME-hardness.

Let us say that warded Datalog$^{\exists,\neg sg,\bot}$, the formalism underlying TriQ-Lite 1.0, is well-suited as a general-purpose KR language, since it extends Datalog with features that allow us to express OWL 2 QL and OWL 2 RL ontologies.

Acknowledgements. Arenas is funded by the Millenium Nucleus Center for Semantic Web Research under grant NC120004. Gottlob is supported by the EPSRC Programme

Grant EP/M025268/ “VADA: Value Added Data Systems – Principles and Architecture”. Pieris is supported by the Austrian Science Fund (FWF), projects P25207-N23 and Y698, and Vienna Science and Technology Fund (WWTF), project ICT12-015.

References

1. Alkhateeb, F., Baget, J.F., Euzenat, J.: Extending SPARQL with regular expression patterns (for querying RDF). J. Web Sem. 7(2), 57–73 (2009)
2. Arenas, M., Gottlob, G., Pieris, A.: Expressive languages for querying the semantic web. In: PODS. pp. 14–26 (2014)
3. Arenas, M., Gutierrez, C., Pérez, J.: Foundations of RDF databases. In: RW. pp. 158–204 (2009)
4. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: On rules with existential variables: Walking the decidability line. Artif. Intell. 175(9-10), 1620–1654 (2011)
5. Barceló, P.: Querying graph databases. In: PODS. pp. 175–188 (2013)
6. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. J. Artif. Intell. Res. 48, 115–174 (2013)
7. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. J. Web Sem. 14, 57–83 (2012)
8. Calì, A., Gottlob, G., Pieris, A.: Towards more expressive ontology languages: The query answering problem. Artif. Intell. 193, 87–128 (2012)
9. Fionda, V., Gutierrez, C., Pirrò, G.: Semantic navigation on the web of data: specification of routes, web fragments and actions. In: WWW. pp. 281–290 (2012)
10. Glimm, B., Ogbuji, C.: SPARQL 1.1 entailment regimes (2013), w3C Recommendation 21 March 2013, `http://www.w3.org/TR/sparql11-entailment/`
11. Gottlob, G., Pieris, A.: Beyond SPARQL under OWL 2 QL entailment regime: Rules to the rescue. In: IJCAI. pp. 2999–3007 (2015)
12. Harris, S., Seaborne, A.: SPARQL 1.1 query language (2013), w3C Recommendation 21 March 2013, `http://www.w3.org/TR/sparql11-query/`
13. Leone, N., Manna, M., Terracina, G., Veltri, P.: Efficiently computable Datalog$^{\exists}$ programs. In: KR (2012)
14. Libkin, L., Reutter, J.L., Vrgoc, D.: Trial for RDF: adapting graph query languages for RDF data. In: PODS. pp. 201–212 (2013)
15. Pérez, J., Arenas, M., Gutierrez, C.: nSPARQL: a navigational language for RDF. J. Web Sem. 8(4), 255–270 (2010)
16. Thomazo, M., Baget, J.F., Mugnier, M.L., Rudolph, S.: A generic querying algorithm for greedy sets of existential rules. In: KR (2012)