Towards a Positive Query Language for RDF

Marcelo Arenas and Martín Ugarte

Pontificia Universidad Católica de Chile

1 Introduction

When querying incomplete databases, the possibility of optionally obtaining information -when available- is a distinctive feature. In particular, this feature is present in SPARQL, the language recommended by the W3C for querying Semantic Web data. Unfortunately, its implementation has been shown to have adverse consequences. Two of the most notable are an increase in evaluation complexity and the possibility of creating negative queries (queries for which there can be a loss of answers when adding new data to a dataset). Many approaches for settling these issues have been presented, being that of restricting SPARQL to *well-designed* graph patterns [2] the most widely adopted. *Well-designed* graph patterns are indeed positive and their evaluation complexity is lower, but they are not capable of expressing disjunction. Moreover, it is hard to prove properties concerning expressibility due to the syntactic-based definition of well-designedness. It remains an open problem whether well-designed patterns are capable of expressing every disjunction-free (UNION-free) positive pattern. In 2007, Polleres presented an extension of the previous fragment aiming to allow disjunction [4]. However, under his notion there are some natural positive graph patterns which become not well-designed.

In this paper we present an introduction to our ongoing project on restricting SPARQL to positive patterns while maintaining desired properties concerning expressibility and evaluation complexity.

2 Definitions

We present a proposal for modifying SPARQL in order to obtain a positive and simple query language while maintaining the possibility of querying for optional information. We introduce a new operator called not-subsumed (NS), and disallow the original OPT operator. The not-subsumed operator was originally defined for the relational model as the *minimum union* operator [1].

In order to save space we do not present the definitions of RDF and SPARQL. We refer the reader to [2], as the following definitions can be seen as an extension to the main definitions presented in that paper. Recall that an RDF graph is a set of triples in $\mathbf{I} \times \mathbf{I} \times \mathbf{I}$, where $\mathbf{I}$ is an infinite set of IRIs. Moreover, given a graph pattern P and an RDF graph G, the evaluation of P over G, denoted by $[\![P]\!]_G$, is a set of partial mappings from var(P) to $\mathbf{I}$, where var(P) is the set of variables mentioned in P.

Definition 1. *Given mappings μ and μ', μ is said to be subsumed by μ', denoted by $\mu \preceq \mu'$, if $dom(\mu) \subseteq dom(\mu')$ and $\mu(?X) = \mu'(?X)$ for every $?X \in dom(\mu)$.*

Since SPARQL allows for adding optional information, the classical way to define positive queries does not fit. In [2] the authors address this issue by defining the notion of weak-monotonicity. A graph pattern P is weakly-monotone if for every two RDF graphs G_1 and G_2 such that $G_1 \subseteq G_2$ it is the case that for every $\mu_1 \in [\![P]\!]_G$ there exists a $\mu_2 \in [\![P]\!]_{G_2}$ such that $\mu_1 \preceq \mu_2$.

Next we formally define the syntax and semantics of the NS operator:

Definition 2. *If P is a graph pattern, then* $\mathrm{NS}(P)$ *is a graph pattern. Moreover, for every RDF graph G:*

$$[\![\mathrm{NS}(P)]\!]_G = \{\mu \in [\![P]\!]_G \mid \textit{there is no } \mu' \in [\![P]\!]_G \textit{ such that } \mu \preceq \mu' \textit{ and } \mu \neq \mu'\}.$$

It is important to notice that the operator NS is capable of simulating the OPT operator since P_1 OPT $P_2 \equiv \mathrm{NS}(P_1$ UNION $(P_1$ AND $P_2))$. Thus, it is not of our interest to allow nested NS operators, as we want to disallow non weakly-monotone patterns.

Definition 3. *A simple pattern is a graph pattern of the form either* $\mathrm{NS}(P)$ *or P, where P is generated using only AND, UNION and FILTER operators.*

Unfortunately, the set of all simple patterns is not capable of expressing disjunctions of graph patterns querying for optional information. Thus, we need to add one more level of disjunction:

Definition 4. *A pattern of the form* P_1 UNION P_2 UNION $\cdots$ UNION P_n *where every P_i ($1 \leq i \leq n$) is a simple graph pattern is called an ns-pattern.*

3 Results

We present the results of our ongoing research. We intend to argue why our modified version of SPARQL is an improvement from the fragments mentioned in the introduction. A first important result is that every graph pattern in de fragments defined above is indeed weakly-monotone.

Theorem 1. *Every ns-pattern is weakly-monotone.*

Another study focus is the combined complexity [5] of evaluating graph patterns for the fragments defined in the previous section. The evaluation problem is formally defined as

$$\textsc{Eval} = \{(G, P, \mu) \mid \begin{array}{l} G \text{ is an RD graph, } P \text{ is a pattern} \\ \text{and } \mu \text{ is a mapping such that } \mu \in [\![P]\!]_G\} \end{array}$$

Theorem 2. $\textsc{Eval}$ *is* $\mathrm{P}_{\|}^{\mathrm{NP}}$*-complete when restricted to ns-patterns.*

The complexity class $\mathrm{P}_{\|}^{\mathrm{NP}}$ contains every language that can be decided in polynomial time with parallel access to an NP-oracle. We know that $\textsc{Eval}$ is PSPACE-complete for the full SPARQL fragment and is coNP-complete when restricted to well-designed graph patterns ([2] and [3], respectively).

One final important result is to compare the expressive power of our fragment against that of well-designed queries.

Theorem 3. *There are ns-patterns that cannot be expressed as a disjunction of* UNION *-free weakly-monotone patterns.*

The above theorem and the fact that coNP $\subseteq \mathrm{P}_{\|}^{\mathrm{NP}} \subseteq$ PSPACE may indicate that our fragment is more expressive than the fragment of well-designed graph patterns. Moreover, if that was the case, we would know that our fragment is more expressive than disjunctions of well-designed graph patterns.

4 Conclusions

We conclude that the defined fragment is a good candidate for changing the way SPARQL allows asking for optional information. This fragment presents an improvement in syntax simplicity over the previous proposal of well-designed queries. From Theorem 3 and Theorem 2 we can also expect it to be an improvement in terms of expressive power.

References

1. Galindo-Legaria, C.A.: Outerjoins as disjunctions. In: SIGMOD Conference. pp. 348–358 (1994)
2. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of sparql. In: ISWC. pp. 30–43 (2006)
3. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of sparql. ACM Trans. Database Syst. 34(3), 16:1–16:45 (Sep 2009)
4. Polleres, A.: From sparql to rules (and back). In: Proceedings of the 16th international conference on World Wide Web. pp. 787–796. WWW '07, ACM (2007)
5. Vardi, M.Y.: The complexity of relational query languages (extended abstract). In: Proceedings of the fourteenth annual ACM symposium on Theory of computing. pp. 137–146. STOC '82, ACM (1982)