Certain Answers for SPARQL?

Claudio Gutierrez[1], Daniel Hernández[1], Aidan Hogan[1], and Axel Polleres[2]

[1] Center for Semantic Web Research,
Department of Computer Science, University of Chile
[2] Vienna University of Economics and Business, Vienna, Austria

Abstract. The standard semantics of SPARQL and the standard semantics of RDF differ fundamentally, sometimes leading to unintuitive answers. In this paper, we thus motivate an alternative semantics for SPARQL based on *certain answers*, taking into account the existential nature of blank nodes, the open-world assumption of RDF, and perhaps even the lack of a unique name assumption. We propose that SPARQL is a natural use-case for applying existing techniques that approximate certain answers in relational database settings.

1 Introduction

The foundational "simple semantics" of RDF is based on three premises: (I) an open world assumption (OWA) where anything not present in the data is assumed to be neither true nor false, (II) a lack of unique name assumption (no UNA) where multiple names can refer to the same thing, and (III) the presence of existential variables in the form of blank nodes. On the other hand, SPARQL – the standard query language for RDF – appears not to respect any of these premises. This may lead, in some cases, to unintuitive solutions.

Taking an example relating to (III), the Wikidata knowledge-base uses existential blank nodes in RDF. For example, in the case of Nicole Brown Simpson, it is stated that she has a killer, but that the killer is unknown, encoded by the following triple with blank node `_:b`: (`w:NicoleBrownSimpson`, `w:killedBy`, `_:b`). Now consider the following SPARQL query over Wikidata (including this triple):

```
SELECT * WHERE {  ?person rdf:type w:Person .
           MINUS { w:NicoleBrownSimpson w:killedBy ?person } }
```
(1)

The query will return the IRIs of all persons in Wikidata: the SPARQL engine will do a comparison to check if `_:b = w:X` – where `w:X` denotes each person – and will return FALSE in every case since blank nodes in data act like constants. Semantically speaking however, the RDF graph does not provide any evidence that these people in the results are not the killer.

Leaving aside blank nodes, let's turn to (I) and assume `w:NicoleBrownSimpson` has no `w:killedBy` edge. SPARQL will again return all people in the data based on the lack of existence of the edge *in the data* (effectively taking an implicit closed world assumption (CWA)). Conversely, the OWA semantics of RDF asserts that the relation may still hold even if the relevant triple is not given.

Likewise, with respect to (II), the comparison `w:x = w:y` will always return FALSE in SPARQL, whereas in the RDF graph, these *may* refer to the same thing: SPARQL sees unequal terms (effectively adopting a UNA) while in RDF, these terms may be referring to the same thing (no UNA).

There is another option, however: SPARQL *could* return answers only where it is *certain* that the person did not kill Nicole Brown Simpson. In other words, nobody would be returned for this query since anybody could have killed her. In the comparison `_:b = w:X`, we could say UNKNOWN instead of FALSE to acknowledge that `_:b` *could* refer to `w:X`; this is similar to how SQL handles the case of `null`s, using three-valued logic. Likewise, checking the (non-)existence of some relation not supported by the data could return UNKNOWN to align with RDF's OWA; and so too could (in)equalities over IRIs, aligning with no-UNA.

In this paper, we do not develop new results, but rather position SPARQL and RDF (with blank nodes) as a natural application of recent results in the database literature relating to *certain answers*. Our proposed aim is to have a keyword, such as `SELECT CERTAIN` [5], which users could opt to use to ensure that all answers returned by the query are true for all models of the RDF graph, such as to ensure that no results are returned in the case(s) previously outlined.

While some authors have looked at certain answers for SPARQL in the context of OWA [2] and OWL entailments [1], to the best of our knowledge, no work has yet looked at RDF graphs with blank nodes, nor considered certain answers more generally in the context of the simple semantics of RDF.

2 Simple Entailment and SPARQL

Given two RDF graphs G and H, we say that G (simple) *entails* H, denoted $G \models H$, if every *model* of G is a model of H; intuitively, H says nothing new over G [3]. If $G \models H$ and $H \models G$, then we say that G and H are (simple) *equivalent*, denoted $G \equiv H$. In the simple semantics of RDF, blank nodes act as existential variables, while IRIs and literals acts as ground terms. With respect to checking entailment, let μ denote a mapping from blank nodes to other RDF terms and let $\mu(G)$ denote the image of an RDF graph G under μ. Then $G \models H$ if and only if there exists a μ such that $\mu(H) \subseteq G$ (denoted $H \to G$) [3].

Example 1. Let us consider two RDF graphs, G on the left, H on the right, where b_* are blank nodes, and all other terms are IRIs.

w	x	b .
w	y	b .
w	y	z .

w	x	b_1 .
w	y	b_2 .
b_3	y	b_2 .

In this case, $G \models H$ with $\mu(\mathrm{b}_1) = \mathrm{b}$, $\mu(\mathrm{b}_2) = \mathrm{b}$, $\mu(\mathrm{b}_3) = \mathrm{w}$, $\mu(H) \subseteq G$. □

Given a SPARQL query Q and two graphs G and H such that $G \equiv H$, it is often the case that (abusing notation) $Q(G) \not\leftrightarrow Q(H)$. More generally, it does not hold that $G \models H$ implies $Q(H) \to Q(G)$ in SPARQL, even when considering just basic graph patterns (aka. conjunctive queries) with basic inequalities.

Example 2. Take G and H ($G \models H$) from Example 1 and the following query Q:

```
SELECT * WHERE { ?w :x ?b1 ; y ?b2 . FILTER(?b1 != ?b2) }
```

Under standard SPARQL query evaluation, $Q(H)$ returns two results – (w,b_1,b_2) and (w,b_2,b_1) – whereas $Q(G)$ is empty. Clearly $Q(H) \not\rightarrow Q(G)$. □

SPARQL query evaluation is counter to the underlying RDF semantics: one can get more answers in SPARQL from an RDF graph with less information (in terms of entailment). Even if a query uses a feature like `MINUS`, the simple semantics of RDF adopts OWA, such that if $G \models H$, any information given by G that is not given by H is considered UNKNOWN, rather than FALSE. We are thus interested in exploring an alternative semantics for SPARQL where we could guarantee, for example, that if $G \models H$, then $Q(H) \rightarrow Q(G)$: that in some sense SPARQL answers are "monotonic" with respect to simple entailment in RDF.

3 Certain Answers for SPARQL?

An intuitive idea is to define the semantics of SPARQL with respect to all possible models of the RDF graph, where an answer is included in the result if and only if it is "valid" for *all* models. This idea corresponds closely with the notion of *certain answers* for incomplete databases.

More specifically, let $\mathbf{C}$ denote a set of constants and $\mathbf{N}$ a set of nulls where both sets are disjoint. Let us likewise assume, for simplicity, a single relation name R with arity k. An *instance* I of R is a k-ary relation $R_I \subseteq \mathbf{C}^k$ with only constants. A *naive instance* I of R is a k-ary relation $R_I \subseteq (\mathbf{CN})^k$ that may contain constants and nulls. Let $\nu : \mathbf{N} \rightarrow \mathbf{C}$ be a mapping from nulls to constants, and abusing notation, let $\nu(R_I)$ denote the image of R_I under ν. A *representative* of a naive instance I is an instance I' where $\nu(R_I) \subseteq R_{I'}$. Each representative fills in the incomplete information of I in a particular way. We denote the set of all representatives of I as $\text{rep}(I)$.

Example 3. The following are instances of a relation R where numbers indicate constants and terms of the form $\bot_i$ indicate a labelled null.

R		
1	$\bot_2$	$\bot_1$
2	$\bot_2$	3

R		
1	2	4
2	2	3

R		
1	3	3
2	3	3
4	5	6

The leftmost instance I_1 is naive since it contains nulls, whereas I_2 and I_3 are also instances but are not naive. Instances I_2 and I_3 are representatives of I_1. □

Now, given a (possibly naive) database instance I and a query Q, the *certain answers* can be defined as:

$$\text{certain}(Q, I) = \bigcap \{Q(I') \mid I' \in \text{rep}(I)\}$$

In other words, a certain answer is an answer for all representatives.

Importantly, representatives in certain answers act somewhat like models in simple entailment, leading to an interesting correspondence: an RDF graph $G \models H$ if and only if $\text{rep}(G) \subseteq \text{rep}(H)$ (assuming an intuitive encoding of G and H as a ternary relation) [4]. From this follows:

Remark 1. Given two RDF graphs G and H, then $G \models H$ if and only if $\text{certain}(Q, G) \subseteq \text{certain}(Q, H)$. □

In other words, if G simple-entails H, then the certain answers for any query Q over H will be "contained in" those for G. This semantic property holds no matter what type of query Q or base query-evaluation $Q(G)$ we consider.

Example 4. Consider again query Q from Example 2 run over the same two graphs. We previously saw that $Q(H) \not\to Q(G)$. On the other hand, $\text{certain}(Q, H)$ is empty because there exists $H' \in \text{rep}(H)$ such that b_1 and b_2 map to the same constant and where $Q(H')$ is thus empty, and thus $\text{certain}(Q, H)$ is empty. It now holds that $\text{certain}(Q, H) \subseteq \text{certain}(Q, G)$. □

To evaluate certain answers for SPARQL, we could apply recent results by Libkin [5], who proposes a sound evaluation of certain answers for SQL by changing how (in)equality expressions and atomic formulae are evaluated in the presence of nulls. We very briefly give an example to illustrate.

Example 5. Under Libkin's proposal, for query (1) in the introduction, the `w:killedBy` pattern would be evaluated as UNKNOWN (with or without the triple mentioning `_:b`), which would lead to no results being returned as certain answers. In the case of Example 2, the inequality comparison would likewise return UNKNOWN, leading to $\text{certain}(Q, H)$ returning no results.

Implementing and experimentally evaluating Libkin's proposals [5] for SPARQL would thus be a natural next step towards a `SELECT CERTAIN` feature. There are also novel questions to tackle in the SPARQL setting, including the effect of bag semantics, non-well-designed queries, etc., on certain answer evaluation.

Acknowledgements: This work was supported by FONDECYT Grant no. 11140900 and by the Millennium Nucleus Center for Semantic Web Research under Grant NC120004.

References

1. S. Ahmetaj, W. Fischl, R. Pichler, M. Simkus, and S. Skritek. Towards reconciling SPARQL and certain answers. In *World Wide Web (WWW)*, pages 23–33, 2015.
2. M. Arenas and J. Pérez. Querying semantic web data with SPARQL. In *Principles of Database Systems (PODS)*, pages 305–316, 2011.
3. C. Gutierrez, C. A. Hurtado, A. O. Mendelzon, and J. Pérez. Foundations of Semantic Web databases. *J. Comput. Syst. Sci.*, 77(3):520–541, 2011.
4. A. Hogan, M. Arenas, A. Mallea, and A. Polleres. Everything you always wanted to know about blank nodes. *J. Web Sem.*, 27:42–69, 2014.
5. L. Libkin. SQL's three-valued logic and certain answers. In *International Conference on Database Theory (ICDT)*, pages 94–109, 2015.