Negation in SPARQL

Renzo Angles[1] and Claudio Gutierrez[2]

[1] Dept. of Computer Science, Universidad de Talca, Chile
[2] Dept. of Computer Science, Universidad de Chile, Chile

Abstract. This paper studies negation in the SPARQL query language. We study the current proposals and implementations under basic desirable assumptions. We show that current implementations do not fulfill basic desiderata for negation. Finally, we show that SPARQL 1.1 does not implement (arithmetic) difference of bags, thus keeping the expressive power in the realm of classical relational algebra of sets.

1 Introduction

The incorporation of negation features into SPARQL, the standard query language for RDF, has always been problematic. First of all, since the philosophy and semantics of RDF is open world, one should have a positive language to query it. Thus, how to include forms of negation? Coincidently –though not clear if by a well agreed strategy– the design of SPARQL 1.0 and 1.1 have followed such desiderata. In fact, as we will show in this short paper, these languages are essentially positive, in the sense that negation reduces to filtering.

In this short note, we will review how different types of negation have been incorporated into SPARQL and study them systematically. In order to organize the discussion, we will base our analysis on the following desiderata for negation:

1. Consistency with the overall semantics defined;
2. Consistency with classical (intuitive) properties of negation; and
3. Well behaveness when combining it with the rest of the operators.

Contributions. We show that the above desiderata is not fulfilled for SPARQL. Although most of the SPARQL operators have been defined under bag semantics, the definitions of negation provided by SPARQL 1.0 (negation by failure) and SPARQL 1.1 (operators MINUS and NOT EXISTS) do not conform with standard negation for bags (i.e. arithmetic difference) [4]. We show that the difference of solution mappings, defined in SPARQL 1.0, can be used to provide a well-founded notion of negation for set semantics. In contrast, we show that the MINUS operator defined by SPARQL 1.1, which works only for set semantics, does not meet the classical intuitive properties of classical negation for sets, e.g. $A - B = A - (A - (A - B))$ and $A - B = A - (A \cap B)$.

The paper is organized as follows. First, and for the sake of completeness, we study negation in SPARQL 1.0. Then, in Section 2, we study the various approaches to negation introduced in SPARQL 1.1. Finally, we sketch the proof that in SPARQL 1.1 arithmetic (bag) negation cannot be simulated.

2 Negation in SPARQL 1.0

SPARQL 1.0 [5] does not provide an explicit operator to express the negation of graph patterns, even when its algebra defines the difference of two multisets of solution mappings. As a way to patching the absence of it, the SPARQL specification establishes that this operation can be implemented as a combination of an optional graph pattern and the bound operator. In this section we analyze in depth the scope and limitations of this approach.

First, consider the following basic definitions. A *solution mapping* is a partial function from variables to RDF terms. The empty solution mapping, denoted μ_0, is the mapping whose domain of variables is empty. Two solution mappings μ_1, μ_2 are compatible, denoted $\mu_1 \sim \mu_2$, if $\mu_1(?X) = \mu_2(?X)$ for every variable $?X$ shared by μ_1 and μ_2. A *multiset (bag) of solution mappings* is denoted by Ω. SPARQL defines the following operations between two multisets of solution mappings Ω_1, Ω_2:

- $\Omega_1 \bowtie \Omega_2 = \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2, \mu_1 \sim \mu_2\}$ where $\text{card}_{\Omega_1 \bowtie \Omega_2}(\mu) = \sum_{\mu=\mu_1\cup\mu_2} \text{card}_{\Omega_1}(\mu_1) \times \text{card}_{\Omega_2}(\mu_2)$
- $\Omega_1 \cup \Omega_2 = \{\mu \mid \mu \in \Omega_1 \vee \mu \in \Omega_2\}$ where $\text{card}_{\Omega_1 \cup \Omega_2}(\mu) = \text{card}_{\Omega_1}(\mu) + \text{card}_{\Omega_2}(\mu)$
- $\Omega_1 \setminus \Omega_2 = \{\mu_1 \in \Omega_1 \mid \forall \mu_2 \in \Omega_2, \mu_1 \nsim \mu_2\}$ where $\text{card}_{\Omega_1 \setminus \Omega_2}(\mu) = \text{card}_{\Omega_1}(\mu)$
- $\Omega_1 ⟕ \Omega_2 = (\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 \setminus \Omega_2)$

We have that $\Omega_\emptyset = \{\mu_0\}$ is the multiset of solution mappings satisfying that $\Omega_1 \bowtie \Omega_\emptyset = \Omega_1$ for any other multiset Ω_1 (i.e. the join identity). Note that the join ($\bowtie$), union ($\cup$) and left-outer join ($⟕$) operators are defined under bag semantics, whereas the minus ($\setminus$) operator works under set semantics.

The evaluation of a graph pattern P over a dataset D with active graph G is denoted by $[\![P]\!]_G^D$ (or just $[\![P]\!]$ when D and G are clear from the context). Given two graph patterns P_1 and P_2, we have that $[\![(P_1 \text{ AND } P_2)]\!] = [\![P_1]\!] \bowtie [\![P_2]\!]$, $[\![(P_1 \text{ UNION } P_2)]\!] = [\![P_1]\!] \cup [\![P_2]\!]$ and $[\![(P_1 \text{ OPT } P_2)]\!] = [\![P_1]\!] ⟕ [\![P_2]\!]$.

Let us introduce the DIFF operator as an explicit way to express the difference of graph patterns (this operator is not defined in SPARQL 1.0 nor in SPARQL 1.1). The *difference of two graph patterns*, denoted $(P_1 \text{ DIFF } P_2)$, is defined as

$$[\![(P_1 \text{ DIFF } P_2)]\!] = [\![P_1]\!] \setminus [\![P_2]\!].$$

A naive implementation. As mentioned in the SPARQL 1.0 specification (see [5], Sec. 11.4.1), the graph pattern $(P_1 \text{ DIFF } P_2)$ can be simulated by an optional graph pattern that introduces a variable of P_2 not occurring in P_1 and testing if such variable is unbounded. This implementation, called *negation as failure* in logic programming, can be represented as a graph pattern of the form

$$((P_1 \text{ OPT } P_2) \text{ FILTER}(\neg \text{bound}(?X))) \tag{1}$$

where $?X$ is a variable of P_2 not occurring in P_1. Hence, this query returns the mappings of $[\![(P_1 \text{ OPT } P_2)]\!]$ satisfying that variable $?X$ is unbounded, i.e. $?X$ does not match P_2. There are two problems with this solution:

				$G_0 \neq \emptyset$		$G_0 = \emptyset$	
	$[\![P_1]\!]$	$[\![P_2]\!]$	$[\![(P_1 \text{ DIFF } P_2)]\!]$	$[\![P_3]\!]$	$[\![P_4]\!]$	$[\![P_3]\!]$	$[\![P_4]\!]$
1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
2	$\emptyset$	$\Omega_\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
3	$\emptyset$	Ω_2	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
4	$\Omega_\emptyset$	$\emptyset$	$\Omega_\emptyset$	$\Omega_\emptyset$	$\Omega_\emptyset$	$\Omega_\emptyset$	$\Omega_\emptyset$
5	$\Omega_\emptyset$	$\Omega_\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\Omega_\emptyset$	$\emptyset$
6	$\Omega_\emptyset$	Ω_2	$\emptyset$	$\emptyset$	$\emptyset$	–	–
7	Ω_1	$\emptyset$	Ω_1	Ω_1	Ω_1	–	–
8	Ω_1	$\Omega_\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	–	–
9	Ω_1	Ω_1	$\emptyset$	$\emptyset$	$\emptyset$	–	–
10	Ω_1	Ω_2	$\Omega_1 \setminus \Omega_2$	$\Omega_1 \setminus \Omega_2$	$\Omega_1 \setminus \Omega_2$	–	–
11	Ω_1	Ω_3	$\emptyset$	$\emptyset$	$\emptyset$	–	–

Table 1. Comparison of two implementations of difference of SPARQL graph patterns. Assume that $\Omega_\emptyset = \{\mu_0\}$ (join identity), and $\Omega_1, \Omega_2, \Omega_3$ are sets of mappings distinct of $\Omega_\emptyset$. Additionally, $\text{dom}(\Omega_1) \cap \text{dom}(\Omega_2) \neq \emptyset$ and $\text{dom}(\Omega_1) \cap \text{dom}(\Omega_3) = \emptyset$. P_3 and P_4 are the graph patterns presented in Equations 2 and 3 respectively. Note that, $[\![P_4]\!]$ is equivalent to $[\![(P_1 \text{ DIFF } P_2)]\!]$ in all the cases, whereas $[\![P_3]\!]$ fails when $G_0 = \emptyset$ (i.e. the default graph G_0 is the empty graph).

- Variable $?X$ cannot be an arbitrary variable. For example, P_2 could be in turn an optional pattern $(P_3 \text{ OPT } P_4)$ where only variables in P_3 are relevant to evaluate the condition $(\neg \text{bound}(?X))$.
- If P_1 and P_2 do not have variables in common, then there is not variable $?X$ to check unboundedness.

Angles and Gutierrez [1] identified these problems and proposed a solution for them. The solution was shown to have a small bug.

Perez's solution. A second approach which fixed the bag was presented in [2], where the authors propose that $(P_1 \text{ DIFF } P_2)$ can be simulated by the pattern

$$((P_1 \text{ OPT } (P_2 \text{ AND } (?X_1\ ?X_2\ ?X_3))) \text{ FILTER} \neg \text{bound}(?X_1)) \tag{2}$$

where $?X_1, ?X_2, ?X_3$ are fresh variables mentioned neither in P_1 nor in P_2.

This solution works well when the empty graph pattern is not allowed in the grammar. In the presence of it, the counterexample is given when P_1 and P_2 are empty graph patterns and the default graph is the empty graph. In this case, we have that $[\![(P_1 \text{ DIFF } P_2)]\!]^D_{G_0} = \emptyset$ whereas $[\![(2)]\!]^D_{G_0} = \{\Omega_\emptyset\}$.

Polleres's solution. A third proposal was sketched by Axel Polleres in the mailing list of the SPARQL W3C Working Group, and discussed with the authors of this paper during the year 2009. The proposed solution is defined formally in the following proposition.

	$[\![P_1]\!]$	$[\![P_2]\!]$	$[\![P_1]\!] \setminus [\![P_2]\!]$	$[\![P_1]\!] \setminus ([\![P_1]\!] \ltimes [\![P_2]\!])$	$[\![P_1]\!] \setminus ([\![P_1]\!] \setminus ([\![P_1]\!] \setminus [\![P_2]\!]))$
1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
2	$\emptyset$	$\Omega_\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
3	$\emptyset$	Ω_2	$\emptyset$	$\emptyset$	$\emptyset$
4	$\Omega_\emptyset$	$\emptyset$	$\Omega_\emptyset$	$\Omega_\emptyset$	$\Omega_\emptyset$
5	$\Omega_\emptyset$	$\Omega_\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
6	$\Omega_\emptyset$	Ω_2	$\emptyset$	$\emptyset$	$\emptyset$
7	Ω_1	$\emptyset$	Ω_1	Ω_1	Ω_1
8	Ω_1	$\Omega_\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
9	Ω_1	Ω_1	$\emptyset$	$\emptyset$	$\emptyset$
10	Ω_1	Ω_2	$\Omega_1 \setminus \Omega_2$	$\Omega_1 \setminus \Omega_2$	$\Omega_1 \setminus \Omega_2$
11	Ω_1	Ω_3	$\emptyset$	$\emptyset$	$\emptyset$

Table 2. Evaluation of the difference operator. Assume that $\Omega_\emptyset = \{\mu_0\}$ (join identity), and $\Omega_1, \Omega_2, \Omega_3$ are sets of mappings distinct of $\Omega_\emptyset$ and satisfying that $\text{dom}(\Omega_1) \cap \text{dom}(\Omega_2) \neq \emptyset$ and $\text{dom}(\Omega_1) \cap \text{dom}(\Omega_3) = \emptyset$. Note that, the difference operator is consistent with classical properties of negation (association and double negation).

Proposition 1. *Let P_1, P_2 be graph patterns. We have that $(P_1 \,\text{DIFF}\, P_2)$ is equivalent to*

$$(((P_1 \,\text{OPT}\, P_2) \,\text{AND}(g \,\text{GRAPH}\ \ (?X \,{:p}\ {:o}))) \,\text{FILTER} \neg \,\text{bound}(?X)) \qquad (3)$$

where g is a named graph containing the single triple (:s :p :o) and $?X$ is a free variable.

Proof. In order to prove that the above equivalence holds, we have analyzed the corner cases for $(P_1 \,\text{DIFF}\, P_2)$. In Table 1, we compare the evaluation of the graph patterns presented in Proposition 1, considering all the possible solutions for $[\![P_1]\!]$ and $[\![P_2]\!]$, and including the special case when the default graph is the empty graph. We showed that the evaluation of both graph patterns are equivalent for all the cases. Additionally, Table 1 includes the results for the graph pattern presented in Equation 2, and shows that it fails in the case (5) when the default graph is the empty graph as described before.

In table 2 we show the evaluation cases for the difference operator and compare with the results of desirable set negation properties mentioned in the introduction. We can see that this negation behaves well in all cases (including the corner cases which made noise).

3 Negation in SPARQL 1.1

SPARQL 1.1 [3] presents two ways of thinking about negation: the MINUS and the NOT EXISTS constructors. In this section we study both features.

	$[\![P_1]\!]$	$[\![P_2]\!]$	$[\![P_1]\!] - [\![P_2]\!]$	$[\![P_1]\!] - ([\![P_1]\!] \bowtie [\![P_2]\!])$	$[\![P_1]\!] - ([\![P_1]\!] - ([\![P_1]\!] - [\![P_2]\!]))$
1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
2	$\emptyset$	$\Omega_\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
3	$\emptyset$	Ω_2	$\emptyset$	$\emptyset$	$\emptyset$
4	$\Omega_\emptyset$	$\emptyset$	$\Omega_\emptyset$	$\Omega_\emptyset$	$\Omega_\emptyset$
5	$\Omega_\emptyset$	$\Omega_\emptyset$	$\Omega_\emptyset$	$\Omega_\emptyset$	$\Omega_\emptyset$
6	$\Omega_\emptyset$	Ω_2	$\Omega_\emptyset$	$\Omega_\emptyset$	$\Omega_\emptyset$
7	Ω_1	$\emptyset$	Ω_1	Ω_1	Ω_1
8	Ω_1	$\Omega_\emptyset$	Ω_1	$\emptyset$	Ω_1
9	Ω_1	Ω_1	$\emptyset$	$\emptyset$	$\emptyset$
10	Ω_1	Ω_2	$\Omega_1 \setminus \Omega_2$	$\Omega_1 \setminus \Omega_2$	$\Omega_1 \setminus \Omega_2$
11	Ω_1	Ω_3	Ω_1	$\emptyset$	Ω_1

Table 3. Evaluation of the subtraction operator (i.e. semantics of MINUS). Assume that $\Omega_\emptyset = \{\mu_0\}$ (join identity), and $\Omega_1, \Omega_2, \Omega_3$ are sets of mappings distinct of $\Omega_\emptyset$ and satisfying that $\mathrm{dom}(\Omega_1) \cap \mathrm{dom}(\Omega_2) \neq \emptyset$ and $\mathrm{dom}(\Omega_1) \cap \mathrm{dom}(\Omega_3) = \emptyset$.

Negation using MINUS. The definition of the MINUS operator is based on the subtraction of multisets of solution mappings. The *subtraction of two multisets of solution mappings*, Ω_1 and Ω_2, is defined as

$$\Omega_1 - \Omega_2 = \{\mu_1 \in \Omega_1 \mid \forall \mu_2 \in \Omega_2, \text{ either } \mu_1 \text{ and } \mu_2 \text{ are not compatible or } \mathrm{dom}(\mu_1) \text{ and } \mathrm{dom}(\mu_2) \text{ are disjoint}\}$$

where $\mathrm{card}_{\Omega_1 \setminus \Omega_2}(\mu_1) = \mathrm{card}_{\Omega_1}(\mu_1)$ (note that this definition also follows set semantics). The *subtraction* of two graph patterns P_1, P_2 is defined as

$$[\![(P_1 \,\mathrm{MINUS}\, P_2)]\!] = [\![P_1]\!] - [\![P_2]\!].$$

In table 3 we show the evaluation cases for the subtraction operator (i.e. the MINUS operator). Note the inconsistencies in (8) and (11) for the expression $A - (A \cap B)$ which differ from $A - B$. In case (8), the problem is given by the join identity property of the resultset $\Omega_\emptyset$. In case (11), the inconsistency is produced by the join of graph patterns having no variables in common.

Negation using NOT EXISTS. Given two graph patterns P_1 and P_2, the expression $(P_1 \,\mathrm{FILTER}\, \mathrm{EXISTS}(P_2))$ returns the mappings of P_1 such that P_2 evaluates to *true*, given the values of variables in P_1 (i.e. the evaluation of P_2 depends on the variables correlated with P_1).

The expression $(P_1 \,\mathrm{FILTER}\, \mathrm{NOT}\, \mathrm{EXISTS}(P_2))$ is equivalent to the graph pattern $(P_1 \,\mathrm{FILTER}\, \neg\, \mathrm{EXISTS}(P_2))$. Hence, we have that

$$[\![(P_1 \,\mathrm{FILTER}\, \mathrm{NOT}\, \mathrm{EXISTS}(P_2))]\!] = \{\mu_1 \in [\![P_1]\!] \mid \mu_1 \text{ is not compatible with every } \mu_2 \in [\![\mu_1(P_2)]\!]\}$$

where $\mu_1(P_2)$ is the graph pattern formed by replacing every occurrence of a variable $?V$ in P_2 by $\mu_1(?V)$, for each $?V \in \mathrm{dom}(\mu_1)$.

Note that, NOT EXISTS is a filter operation, therefore the solutions keep the cardinality of the solutions (of the graph pattern) being filtered.

It is straightforward to show that $(P_1 \,\text{FILTER}\, \text{NOT}\, \text{EXISTS}(P_2))$ is equivalent to $(P_1 \,\text{DIFF}\, P_2)$ when the variables of P_1 do not occur inside filter constraints of P_2, i.e. the graph patterns of P_2 are filter safe. The occurrence of correlated variables inside filter constraints can introduce new features in the language which are not discussed in this paper.

4 Negation under bag semantics

We have seen that all current specifications of SPARQL do not really include negation, but just forms of filtering out some mappings. It can be shown that indeed, SPARQL 1.1 cannot simulate difference of bags.

Theorem 1. *SPARQL 1.1 does not implement and cannot simulate the difference of bags.*

Proof. The proof is a long, but straightforward, case by case analysis. We sketch here the two core bolts to which one can reduce such a checking. First, note that the algebra operators never performs difference of bags nor allow for it. The semantics of bags is indicated in the documentation by their cardinality. It is not difficult to check in the SPARQL specification (see [3], Sec 18.5) that no operator performs negative cardinality operations.

Second, an alternative possibility to get difference of bags (arithmetic difference) would be using in smart ways the $-$ sign of multiplicative expressions (see [3], line 116 in the grammar), and use it to create ex-post the corresponding copies of a solution mapping. But this is not possible as there is no way of creating mappings from a numerical value in SPARQL 1.1. (essentially check the new constructors BIND and VALUES do not do the desired work).

References

1. R. Angles and C. Gutierrez. The Expressive Power of SPARQL. In *Proc. of the International Semantic Web Conference (ISWC)*, 2008.
2. M. Arenas and J. Pérez. Querying Semantic Web Data with SPARQL. In *30th ACM Symposium on Principles of Database Systems (PODS)*, 2011.
3. S. Harris and A. Seaborne. SPARQL 1.1 Query Language - W3C Recommendation. http://www.w3.org/TR/2013/REC-sparql11-query-20130321/, March 21 2013.
4. L. Libkin. Expressive power of SQL. *Theoretical Computer Science*, 296(3):379–404, 2003.
5. E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Recommendation. http://www.w3.org/TR/2008/REC-115-sparql-query-20080115/, January 15 2008.