Semantics of SPARQL

Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez

Abstract. This paper presents a formal semantics of SPARQL based on [1]. Extensions to [1] are discussed, including support for blank nodes in graph patterns, and bag/multisets semantics for solutions. The motivation for this paper is to make the definitions of [1] closer to the last official SPARQL release (4th October 2006), and to serve as feedback to the *RDF Data Access Working Group* .

1 Preliminary Notions

Definition 1.1 (RDF Terms, Triples, and Variables) *Assume there are pairwise disjoint infinite sets I, B, and L (IRIs, Blank nodes, and literals). A tuple $(s, p, o) \in (I \cup B) \times I \times (I \cup B \cup L)$ is called an* RDF triple*. In this tuple, s is the* subject*, p the* predicate *and o the* object*. We denote the union $I \cup B \cup L$ by T (RDF terms). Assume additionally the existence of an infinite set V of variables disjoint from the above sets.*

Definition 1.2 (RDF Graph) *An* RDF graph *is a set of RDF triples. If G is an RDF graph,* $\mathrm{term}(G)$ *is the set of elements of T appearing in the triples of G, and* $\mathrm{blank}(G)$ *is the set of blank nodes appearing in G, i.e.* $\mathrm{blank}(G) = \mathrm{term}(G) \cap B$.

Definition 1.3 (RDF Dataset) *An* RDF dataset *[2] is a set*

$$\mathcal{D} = \{G_0, \langle u_1, G_1\rangle, \ldots, \langle u_n, G_n\rangle\}$$

where $G_0, \ldots, G_n$ are RDF graphs, $u_1, \ldots, u_n$ are IRIs, and $n \geq 0$. In the dataset, G_0 is the default graph*, and the pairs $\langle u_i, G_i\rangle$ are* named graphs*, with u_i the name of G_i. Every dataset $\mathcal{D}$ is equipped with a function $d_{\mathcal{D}}$ such that $d_{\mathcal{D}}(u) = G$ if $\langle u, G\rangle \in \mathcal{D}$ and $d_{\mathcal{D}}(u) = \emptyset$ otherwise. Additionally,* $\mathrm{name}(\mathcal{D})$ *stands for the set of IRIs that are names of graphs in $\mathcal{D}$, and* $\mathrm{term}(\mathcal{D})$ *and* $\mathrm{blank}(\mathcal{D})$ *stand for the set of terms and blank nodes appearing in the graphs of $\mathcal{D}$, respectively. For the sake of simplicity, we assume that the graphs in a dataset have disjoint sets of blank nodes, i.e. for $i \neq j$,* $\mathrm{blank}(G_i) \cap \mathrm{blank}(G_j) = \emptyset$.

Definition 1.4 (Mapping) *A* mapping *μ from V to T is a partial function $\mu : V \to T$. The domain of μ,* $\mathrm{dom}(\mu)$*, is the subset of V where μ is defined. The empty mapping $\mu_\emptyset$ is a mapping such that* $\mathrm{dom}(\mu_\emptyset) = \emptyset$ *(i.e. $\mu_\emptyset = \emptyset$).*

2 Syntax and Semantics of Basic Graph Patterns

Definition 2.1 (Triple and Basic Graph Pattern) *A tuple* $t \in (I \cup L \cup V) \times (I \cup V) \times (I \cup L \cup V)$ *is a* triple pattern. *A* Basic Graph Pattern *is a finite set of triple patterns. Given a triple pattern* t, $\mathrm{var}(t)$ *is the set of variables occurring in* t. *Similarly, given a basic graph pattern* P, $\mathrm{var}(P) = \bigcup_{t \in P} \mathrm{var}(t)$, *i.e.* $\mathrm{var}(P)$ *is the set of variables occurring in* P.

Note 2.2 In our definitions, triple and basic graph patterns do not contain blank nodes. See Section 5.1 for extensions in this respect.

Definition 2.3 (Basic Graph Patterns and Mappings) *Given a triple pattern* t *and a mapping* μ *such that* $\mathrm{var}(t) \subseteq \mathrm{dom}(\mu)$, $\mu(t)$ *is the triple obtained by replacing the variables in* t *according to* μ. *Given a basic graph pattern* P *and a mapping* μ *such that* $\mathrm{var}(P) \subseteq \mathrm{dom}(\mu)$, *we have that* $\mu(P) = \bigcup_{t \in P} \{\mu(t)\}$, *i.e.* $\mu(P)$ *is the set of triples obtained by replacing the variables in the triples of* P *according to* μ.

Definition 2.4 (Subgraph Matching) *Let* G *be an RDF graph over* T, *and* P *a basic graph pattern. The* evaluation *of* P *over* G, *denoted by* $[\![P]\!]_G$ *is defined as the set of mappings*

$$[\![P]\!]_G = \{\mu : V \to T \mid \mathrm{dom}(\mu) = \mathrm{var}(P) \text{ and } \mu(P) \subseteq G\}.$$

If $\mu \in [\![P]\!]_G$, *we say that* μ *is a* solution for P in G.

Note 2.5 For every RDF graph G, $[\![\,\emptyset\,]\!]_G = \{\mu_\emptyset\}$, i.e. the evaluation of an empty basic graph pattern against any graph always results in the set containing only the empty mapping. For every basic graph pattern $P \neq \emptyset$, $[\![P]\!]_\emptyset = \emptyset$.

3 Syntax and Semantics of General Graph Patterns

3.1 Syntax

Definition 3.1 (Value Constraint) *A* SPARQL value constraint *is defined recursively as follows:*

(1) If $?X, ?Y \in V$ *and* $u \in I \cup L$, *then* $?X = u$, $?X = ?Y$, $\mathrm{bound}(?X)$, $\mathrm{isIRI}(?X)$, $\mathrm{isLiteral}(?X)$, *and* $\mathrm{isBlank}(?X)$ *are* atomic *value constraints.*[1]
(2) If R_1 *and* R_2 *are value constraints then* $\neg R_1$, $R_1 \wedge R_2$, *and* $R_1 \vee R_2$ *are value constraints.*

Note 3.2 In our definitions, value constraints do not contain blank nodes.

Definition 3.3 *A* SPARQL *graph pattern is defined recursively as follows:*

(1) A basic graph pattern is a graph pattern.

[1] For a complete list of atomic value constraints see [2].

(2) If P_1 and P_2 are graph patterns then (P_1 AND P_2), (P_1 UNION P_2), and (P_1 OPT P_2) are graph patterns.
(3) If P is a graph pattern and $X \in I \cup V$ then (X GRAPH P) is a graph pattern.
(4) If P is a graph pattern and R is a value constraint, then (P FILTER R) is a graph pattern.

3.2 Semantics

Definition 3.4 *Given a mapping μ and a value constraint R, we define a notion of satisfaction of R by μ, denoted by $\mu \models R$, from a notion of* evaluation *in a three valued logic with values* $\{\texttt{true}, \texttt{false}, \texttt{error}\}$. *For every atomic value constraint R, excluding* $\text{bound}(\cdot)$, *if* $\text{var}(R) \not\subseteq \text{dom}(\mu)$ *the evaluation of R results in* `error`*; else, the evaluation results in* `true` *if:*

- *R is $?X = c$ and $\mu(?X) = c$,*
- *R is $?X = ?Y$ and $\mu(?X) = \mu(?Y)$,*
- *R is* $\text{isIRI}(?X)$ *and $\mu(?X) \in I$,*
- *R is* $\text{isLiteral}(?X)$ *and $\mu(?X) \in L$,*
- *R is* $\text{isBlank}(?X)$ *and $\mu(?X) \in B$,*

and results in `false` *otherwise. For the case of* $\text{bound}(?X)$, *the evaluation results in* `true` *iff* $?X \in \text{dom}(\mu)$, *else it results in* `false`. *For non-atomic constraints, the evaluation is defined as usual in a three valued logic:*

R_1	R_2	$R_1 \wedge R_2$	$R_1 \vee R_2$
true	true	true	true
true	error	error	true
true	false	false	true
error	true	error	true
error	error	error	error
error	false	false	error
false	true	false	true
false	error	false	error
false	false	false	false

R_1	$\neg R_1$
true	false
error	error
false	true

Finally, $\mu \models R$ iff the evaluation of R against μ results in `true`.

Definition 3.5 (Compatible Mappings) *Two mappings $\mu_1 : V \to T$ and $\mu_2 : V \to T$ are* compatibles *if for every $?X \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2)$ it is the case that $\mu_1(?X) = \mu_2(?X)$, i.e. when $\mu_1 \cup \mu_2$ is also a mapping.*

Note 3.6 Two mappings with disjoint domains are always compatible, and the empty mapping $\mu_\emptyset$ is compatible with any other mapping. Intuitively, μ_1 and μ_2 are compatibles if μ_1 *can be extended* with μ_2 to obtain a new mapping, and vice versa.

Definition 3.7 (Set of Mappings and Operations) *Let Ω_1 and Ω_2 be sets of mappings. We define the join of, the union of, and the difference between Ω_1 and Ω_2 as:*

$$\Omega_1 \bowtie \Omega_2 = \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2 \text{ are compatible mappings}\},$$
$$\Omega_1 \cup \Omega_2 = \{\mu \mid \mu \in \Omega_1 \text{ or } \mu \in \Omega_2\},$$
$$\Omega_1 \setminus \Omega_2 = \{\mu \in \Omega_1 \mid \text{ for all } \mu' \in \Omega_2,\ \mu \text{ and } \mu' \text{ are not compatible}\}.$$

Based on the previous operators, we define the left outer-join as:

$$\Omega_1 ⟕ \Omega_2 = (\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 \setminus \Omega_2).$$

Note 3.8 Intuitively, $\Omega_1 \bowtie \Omega_2$ is the set of mappings that result from extending mappings in Ω_1 with their compatible mappings in Ω_2, and $\Omega_1 \setminus \Omega_2$ is the set of mappings in Ω_1 that cannot be extended with any mapping in Ω_2. The operation $\Omega_1 \cup \Omega_2$ is the usual set theoretical union. A mapping μ is in $\Omega_1 ⟕ \Omega_2$ if it is the extension of a mapping of Ω_1 with a compatible mapping of Ω_2, or if it belongs to Ω_1 and cannot be extended with any mapping of Ω_2.

Definition 3.9 *Let $\mathcal{D}$ be an RDF dataset and G an RDF graph in $\mathcal{D}$. The* evaluation *of a graph pattern over G in the dataset $\mathcal{D}$, denoted by $[\![\cdot]\!]_G^{\mathcal{D}}$, is defined recursively as follows:*

(1) If P is a basic graph pattern, then $[\![P]\!]_G^{\mathcal{D}} = [\![P]\!]_G$.

(2)
- $[\![(P_1 \text{ AND } P_2)]\!]_G^{\mathcal{D}} = [\![P_1]\!]_G^{\mathcal{D}} \bowtie [\![P_2]\!]_G^{\mathcal{D}}$,
- $[\![(P_1 \text{ UNION } P_2)]\!]_G^{\mathcal{D}} = [\![P_1]\!]_G^{\mathcal{D}} \cup [\![P_2]\!]_G^{\mathcal{D}}$,
- $[\![(P_1 \text{ OPT } P_2)]\!]_G^{\mathcal{D}} = [\![P_1]\!]_G^{\mathcal{D}} ⟕ [\![P_2]\!]_G^{\mathcal{D}}$.

(3)
- *If $u \in I$, then $[\![(u \text{ GRAPH } P)]\!]_G^{\mathcal{D}} = [\![P]\!]_{d_{\mathcal{D}}(u)}^{\mathcal{D}}$,*
- *if $?X \in V$, then*

$$[\![(?X \text{ GRAPH } P)]\!]_G^{\mathcal{D}} = \bigcup_{v \in \text{name}(\mathcal{D})} \left([\![P]\!]_{d_{\mathcal{D}}(v)}^{\mathcal{D}} \bowtie \{\mu_{?X \to v}\} \right).$$

where $\mu_{?X \to v}$ is a mapping such that $\text{dom}(\mu) = \{?X\}$ and $\mu(?X) = v$.

(4) $[\![(P \text{ FILTER } R)]\!]_G^{\mathcal{D}} = \{\mu \in [\![P]\!]_G^{\mathcal{D}} \mid \mu \models R\}$.

Moreover, the evaluation of a graph pattern P over the RDF dataset $\mathcal{D}$, denoted by $[\![P]\!]^{\mathcal{D}}$, is simply $[\![P]\!]_{G_0}^{\mathcal{D}}$ where G_0 is the default graph in $\mathcal{D}$.

Note 3.10 The GRAPH operator in [2] transforms the dataset into a new one with a different default graph, and the queries are always evaluated using the default graph. In our definition, the dataset is a parameter of the evaluation as well as the graph that is being used. These two alternative definitions are equivalent, that is, the same set of solutions is obtained by using these two alternative semantics.

Notice that since the graphs in a dataset do note share blank nodes (see Definition 1.3), the mappings that result from distinct graphs in the evaluation of ($?X$ GRAPH P) cannot share blank nodes.

Note 3.11 The semantics of SPARQL in [2] is defined in terms of *solutions* for graph patterns rather than operations among sets of mappings. In our definitions, the solutions for a pattern P in a dataset $\mathcal{D}$ are all the mappings in $[\![P]\!]^{\mathcal{D}}$. See Definition 6.2 for a comparison.

Note 3.12 Several algebraic properties of graph patterns are proved in [1], most notably that AND and UNION are associative and commutative. This permits us to avoid parenthesis when writing sequences of AND operators or UNION operators. This is consistent with the definitions of Group Graph Pattern and Union Graph Pattern in [2] as being just sets of graph patterns.

Proposition 3.13 *Let $\{t_1, t_2, \ldots, t_n\}$ be a basic graph pattern, where $n \geq 1$ and every t_i is a triple pattern $(1 \leq i \leq n)$. Then for every dataset $\mathcal{D}$:*

$$[\![\{t_1, t_2, \ldots, t_n\}]\!]^{\mathcal{D}} = [\![(\{t_1\} \text{ AND } \{t_2\} \text{ AND } \cdots \text{ AND } \{t_n\})]\!]^{\mathcal{D}}.$$

Note 3.14 The above proposition implies that it is equivalent to consider basic graph patterns or triple patterns as the base case when defining SPARQL general graph patterns. Indeed triple patterns are used in [1] as the base case. When blank nodes are considered, basic graph patterns must be used as the base case, because the scope of blank nodes are basic graph patterns according to [2]. See Section 5.1 for details.

4 Semantics of Query Result Forms

4.1 SELECT Result Form

Definition 4.1 *Given a mapping $\mu : V \to T$ and a set of variables $W \subseteq V$, the* restriction *of μ to W, denoted by $\mu_{|W}$, is a mapping such that* $\mathrm{dom}(\mu_{|W}) = \mathrm{dom}(\mu) \cap W$ *and $\mu_{|W}(?X) = \mu(?X)$ for every $?X \in \mathrm{dom}(\mu) \cap W$.*

Definition 4.2 *A SELECT query is a tuple (W, P), where $W \subseteq V$ is a finite set of variables and P is a graph pattern. The* answer *of (W, P) in a dataset $\mathcal{D}$ is the set of mappings:*

$$\{\mu_{|W} \mid \mu \in [\![P]\!]^{\mathcal{D}}\}.$$

4.2 CONSTRUCT Result Form

Definition 4.3 (Templates) *A* template *is a finite subset of $(T \cup V) \times (I \cup V) \times (T \cup V)$. Given a mapping μ and a template H, we have that $\mu(H)$ is the result of replacing in H every variable $x \in \mathrm{dom}(\mu)$ by $\mu(x)$, and leaving unchanged the remaining variables.*

Moreover, $\mathrm{blank}(H)$ *is defined to be the set of blank nodes appearing in H.*

Note 4.4 The definition of *template* is similar to basic graph patterns, with the addition that templates can have blank nodes in their triples. Furthermore, for a template H we do not impose the restriction $\mathrm{var}(H) \subseteq \mathrm{dom}(\mu)$ in order to define $\mu(H)$, as we did for basic graph patterns in Definition 2.3.

Definition 4.5 (Renaming Function) *Let H be a template, P a graph pattern, and $\mathcal{D}$ a dataset. Then we say that a set of functions:*

$$\{f_\mu \mid \mu \in [\![P]\!]^{\mathcal{D}}\},$$

is a set of renaming functions *for H and $[\![P]\!]^{\mathcal{D}}$ if: (1) the domain of every function f_μ is* $\mathrm{blank}(H)$ *and its range is a subset of* $(B \setminus \mathrm{blank}(\mathcal{D}))$*, that is, it does not include any blank from* $\mathrm{blank}(\mathcal{D})$*, (2) every function f_μ is one-to-one, and (3) for every pair of distinct mappings $\mu, \nu \in [\![P]\!]^{\mathcal{D}}$, f_μ and f_ν have disjoint ranges (codomains).*

Moreover, we denote by $f_\mu(H)$ the template resulting from replacing the blank nodes in H according to f_μ.

Definition 4.6 *A CONSTRUCT query is a tuple $q = (H, P)$ where H is a template and P is a graph pattern. Let $\mathcal{D}$ be a dataset such that $\mathcal{D}$ does not share blank nodes with H, and consider a fixed set of renaming functions $F = \{f_\mu \mid \mu \in [\![P]\!]^{\mathcal{D}}\}$. Then the answer of q in the dataset $\mathcal{D}$ (with renaming functions in F) is the RDF graph:*

$$\bigcup_{\mu \in [\![P]\!]^{\mathcal{D}}} \Big(\mu(f_\mu(H)) \ \cap \ ((I \cup B) \times I \times (I \cup L \cup B)) \Big).$$

Note 4.7 The answer of q in a dataset $\mathcal{D}$ is defined as the union over $\mu \in [\![P]\!]^{\mathcal{D}}$ of the set of triples $\mu(f_\mu(H))$ intersected with the set of well-formed RDF triples, to ensure that the resulting set is a valid RDF graph.

The introduction of the renaming functions f_μ, for every $\mu \in [\![P]\!]^{\mathcal{D}}$, is to ensure that there are fresh blank nodes in the template before the application of each μ.

5 Extensions to the Subgraph Matching Approach

5.1 Graph Patterns with Blank Nodes

Definition 5.1 *We extend the definitions of triple patterns to be tuples in the set $(T \cup V) \times (I \cup V) \times (T \cup V)$. The difference with Definition 2.1 is that now triple patterns are allowed to have blank nodes as components. Equivalently, we extend the definition of basic graph patterns. Also for a triple pattern t and a basic graph pattern P, we define* $\mathrm{blank}(t)$ *and* $\mathrm{blank}(P)$ *as the sets of blank nodes appearing in t and P, respectively.*

Definition 5.2 *Let G be an RDF graph, and P a basic graph pattern with blank nodes. We say that a mapping μ is a solution for P in G if:*

- $\mathrm{dom}(\mu) = \mathrm{var}(P)$
- *there exists a substitution* $\theta : \mathrm{blank}(P) \to \mathrm{term}(G)$ *such that* $\mu(\theta(P)) \subseteq G$,

where $\theta(P)$ is the basic graph pattern that results from replacing the blank nodes of P according to θ. Now, the evaluation *of P over G, denoted by $[\![P]\!]_G$, is defined as the set of all solutions for P in G.*

Note 5.3 This definition extends the definition of the semantics of graph pattern expressions without blanks nodes, as by using the substitution $\theta : \emptyset \to \mathrm{term}(G)$, we obtain the same set of solutions as in Definition 2.4 for a graph pattern expression P without blank nodes (since $\theta(P) = P$).

Definition 5.4 *Given a dataset $\mathcal{D}$ and a general graph pattern P constructed from basic graph patterns possibly with blank nodes, we define the evaluation of P in $\mathcal{D}$, denoted by $[\![P]\!]^{\mathcal{D}}$, by applying Definition 3.9 with the base case as in Definition 5.2.*

5.2 Bag/Multisets Semantics and Cardinality

Definition 5.5 (Bags of Mappings and Operations) *A* bag *of mappings is a set of mappings in which every mapping is* annotated *with a positive integer that represents the* cardinality *of that mapping in the bag. We will denote the cardinality of the mapping μ in the bag M by $\mathrm{card}_M(\mu)$ (or simply $\mathrm{card}(\mu)$ when M is understood from the context). If $\mu \notin M$ then $\mathrm{card}_M(\mu) = 0$.*

In Definition 3.7, we consider operations between set of mappings. Those operations can be extended to bags, roughly speaking, making the operations not to discard duplicates. Formally, if Ω_1, Ω_2 are bags of mappings, then

$$\begin{aligned}
&\textit{for } \mu \in \Omega_1 \bowtie \Omega_2,\ \mathrm{card}_{\Omega_1 \bowtie \Omega_2}(\mu) = \sum_{\mu=\mu_1\cup\mu_2} \mathrm{card}_{\Omega_1}(\mu_1) \cdot \mathrm{card}_{\Omega_2}(\mu_2),\\
&\textit{for } \mu \in \Omega_1 \cup \Omega_2,\ \mathrm{card}_{\Omega_1 \cup \Omega_2}(\mu) = \mathrm{card}_{\Omega_1}(\mu) + \mathrm{card}_{\Omega_2}(\mu),\\
&\textit{for } \mu \in \Omega_1 \setminus \Omega_2,\ \mathrm{card}_{\Omega_1 \setminus \Omega_2}(\mu) = \mathrm{card}_{\Omega_1}(\mu).
\end{aligned}$$

Definition 5.6 (Cardinality of Basic Graph Pattern Solutions) *Consider a basic graph pattern P (possibly with blank nodes) and an RDF graph G. The cardinality of the mapping $\mu \in [\![P]\!]_G$ is defined as the number of distinct substitutions $\theta : \mathrm{blank}(P) \to \mathrm{term}(G)$ such that $\mu(\theta(P)) \subseteq G$, i.e.*

$$\mathrm{card}_{[\![P]\!]_G}(\mu) = |\{\theta : \mathrm{blank}(P) \to \mathrm{term}(G) \mid \mu(\theta(P)) \subseteq G\}|.$$

Note 5.7 For a basic graph pattern P without blank nodes, every solution $\mu \in [\![P]\!]_G$ has cardinality 1, as in this case the only possible substitution is $\theta : \emptyset \to \mathrm{term}(G)$. Notice that this is consistent with the fact that an RDF graph is a set (without duplicates).

Definition 5.8 *Given a dataset $\mathcal{D}$ and a general graph pattern P composed from basic graph patterns possibly with blank nodes, we define the evaluation of P in $\mathcal{D}$ using a bag/multiset semantics, simply as in Definition 3.9 but applying bag operators (as in Definition 5.5) and with the base case as in Definition 5.6.*

Moreover, the cardinality of a mapping $\mu \in [\![P]\!]^{\mathcal{D}}$ is denoted by $\mathrm{card}_{[\![P]\!]^{\mathcal{D}}}(\mu)$.

Proposition 5.9 *Let P be a graph pattern without blank nodes and composed only by* AND, FILTER *and* OPT *operators, and let $\mathcal{D}$ be an RDF dataset. Then every solution $\mu \in [\![P]\!]^{\mathcal{D}}$ has cardinality* 1.

Note 5.10 The above proposition implies that in absence of blank nodes in graph patterns, duplicated solutions could be generated only by the use of UNION and GRAPH operators.

Definition 5.11 (Cardinality in SELECT Result Form) *Informally, for bag/multiset semantics, in a SELECT query $q = (W, P)$ we simply take the projection of the solutions for P over variables W without discarding duplicates. Formally, given a SELECT query $q = (W, P)$ and a mapping μ in the answer of q in dataset $\mathcal{D}$, we define the cardinality of μ as*

$$\sum_{\nu_{|W}=\mu} \mathrm{card}_{[\![P]\!]^{\mathcal{D}}}(\nu).$$

Note 5.12 For a CONSTRUCT query (H, P), every possible duplicate generated in the evaluation of P is discarded when taking the (set) union of the mappings applied to H. Indeed, CONSTRUCT generates an RDF graph which by definition is a set.

6 Final Remarks

Remark 6.1 (Solutions versus Set of Mappings and Operators) The semantics of SPARQL is defined in [2] in terms of *solutions* for graph patterns rather than operations among sets of mappings. Note that there is a close correspondence between the notions of *mappings* and *solutions* as follows: a mapping μ is a solution for graph pattern P in an RDF graph G of a dataset $\mathcal{D}$ iff $\mu \in [\![P]\!]^{\mathcal{D}}_{G}$.

Definition 6.2 *The following are alternative (informal) definitions of the semantics of graph patterns in term of solutions. We concentrate here in the algebra.*

(2)
- *μ is a solution for (P_1* AND *P_2) if $\mu = \mu_1 \cup \mu_2$, where μ_1 is a solution for P_1 and μ_2 is a solution for P_2.*
- *μ is a solution for (P_1* UNION *P_2) if μ is a solution for P_1 or μ is a solution for P_2.*
- *μ is a solution for (P_1* OPT *P_2) if either (i) $\mu = \mu_1 \cup \mu_2$, where μ_1 is a solution for P_1 and μ_2 is a solution for P_2, or (ii) μ is a solution for P_1 and there is no mapping μ' that is a solution for P_2 and is compatible with μ.*

(3)
- *μ is a solution for (u* GRAPH *P) (in the dataset $\mathcal{D}$) if μ is a solution for P in the graph with name u (in the dataset $\mathcal{D}$).*
- *μ is a solution for ($?X$* GRAPH *P) (in the dataset $\mathcal{D}$) if $\mu = \mu' \cup \{?X \rightarrow v\}$, where μ' is a solution for P in the graph with name v (in the dataset $\mathcal{D}$).*

(4) μ is a solution for (P FILTER R) *if μ is a solution for P that satisfies R.*

Remark 6.3 (General Framework versus Subgraph Matching) In [2], in defining the semantics of SPARQL a notion of entailment is introduced with the idea of making the definition generic enough to support notions more general than simple entailment (e.g. RDFS entailment, OWL entailment, etc.). Current developments of the *Data Access Working Group* not settled yet this issue. What is clear consensus is that in the case of simple RDF any definition should coincide with subgraph matching, which is the approach followed in this paper as well as in [1]. It is important to notice that a final decision on this point may affect only the definition of the semantics of basic graph pattern expressions; the remaining definitions (for the algebra) will not be affected.

Remark 6.4 (Query Languages and Entailment) We would like to add a small methodological remark on the issue of incorporating a notion of entailment (at least the notions present in RDF) into a query language. It can be proved that it necessarily brings strong compromises in terms of computational complexity, as the following result shows:

Proposition [Franconi–Gutierrez]: Let KB a knowledge base, and let $\equiv$ be the notion of logical equivalence among elements in KB. Assume further that the query language has the following two natural properties: for every pair of knowledge bases D and D' and for every query q, (1) If $D \equiv D'$ then $q(D) = q(D')$, and (2) The language contains the identity query. Then, the cost of evaluating a query (worst case) has at least the complexity of evaluating $\equiv$.

Condition (2) is an *almost unavoidable* requirement in any reasonable query language. Condition (1) is what strongly differentiates a Database approach to query languages from a KB-approach. In databases one only need the weaker condition: (1') If $D = D'$ then $q(D) = q(D')$. Note that current version of SPARQL treats simple RDF graphs as databases (following the subgraph matching approach), indeed the semantics does not satisfy (1), but satisfies (1') and (2). In fact, given graphs $G_1 = \{(a, c, d), (a, c, B)\}$ and $G_2 = \{(a, c, d)\}$, clearly equivalent under simple RDF, the query $q = (a, c, ?X)$ gives two different answers. The approach in this paper is also a database oriented approach to the semantics of SPARQL in the simple RDF case.

References

1. J. Pérez, M. Arenas, C. Gutierrez. *Semantics and Complexity of SPARQL.* 5th International Semantic Web Conference (ISWC-06), November 2006.
2. E. Prud'hommeaux, A. Seaborne. *SPARQL Query Language for RDF.* W3C WD 4 October 2006. `http://www.w3.org/TR/2006/WD-rdf-sparql-query-20061004/`.

A Proofs

Proof of Proposition 3.13 The proof is by induction on n. The base case $n = 1$ trivially holds. Now, by definition and using the induction hypothesis we have

$$[\![(\{t_1\} \text{ AND } \cdots \text{ AND } \{t_{n+1}\})]\!]^{\mathcal{D}} = [\![\{t_1, \ldots, t_n\}]\!]^{\mathcal{D}} \bowtie [\![\{t_{n+1}\}]\!]^{\mathcal{D}}$$

and then we must prove that

$$[\![\{t_1, \ldots, t_{n+1}\}]\!]_{G_0} = [\![\{t_1, \ldots, t_n\}]\!]_{G_0} \bowtie [\![\{t_{n+1}\}]\!]_{G_0}$$

with G_0 the default graph of $\mathcal{D}$. We first show that

$$[\![\{t_1, \ldots, t_{n+1}\}]\!]_{G_0} \subseteq [\![\{t_1, \ldots, t_n\}]\!]_{G_0} \bowtie [\![\{t_{n+1}\}]\!]_{G_0}.$$

Let μ be a mapping in $[\![\{t_1, \ldots, t_{n+1}\}]\!]_{G_0}$ then $\mu(\{t_1, \ldots, t_{n+1}\}) \subseteq G_0$. Let μ_1 be the restriction of μ to domain $\text{var}(\{t_1, \ldots, t_n\})$, and μ_2 be the restriction of μ to domain $\text{var}(t_{n+1})$. We have that $\mu(\{t_1, \ldots, t_n\} \cup \{t_{n+1}\}) \subseteq G_0$ then $\mu_1(\{t_1, \ldots, t_n\}) \subseteq G_0$ and $\mu_2(\{t_{n+1}\}) \subseteq G_0$, and then $\mu_1 \in [\![\{t_1, \ldots, t_n\}]\!]_{G_0}$ and $\mu_2 \in [\![\{t_{n+1}\}]\!]_{G_0}$. Note also that μ_1 and μ_2 are compatibles and then $\mu \in [\![\{t_1, \ldots, t_n\}]\!]_{G_0} \bowtie [\![\{t_{n+1}\}]\!]_{G_0}$ because $\mu = \mu_1 \cup \mu_2$. Now we show that

$$[\![\{t_1, \ldots, t_n\}]\!]_{G_0} \bowtie [\![\{t_{n+1}\}]\!]_{G_0} \subseteq [\![\{t_1, \ldots, t_{n+1}\}]\!]_{G_0}.$$

Let $\mu \in [\![\{t_1, \ldots, t_n\}]\!]_{G_0} \bowtie [\![\{t_{n+1}\}]\!]_{G_0}$ then $\mu = \mu_1 \cup \mu_2$ for two compatibles mappings $\mu_1 \in [\![\{t_1, \ldots, t_n\}]\!]_{G_0}$ and $\mu_2 \in [\![\{t_{n+1}\}]\!]_{G_0}$. Note first that $\text{dom}(\mu) = \text{var}(\{t_1, \ldots, t_{n+1}\})$. Now consider $\mu(\{t_1, \ldots, t_{n+1}\})$ by definition we have

$$\mu(\{t_1, \ldots, t_{n+1}\}) = \bigcup_{i=1}^{n+1} \{\mu(t_i)\}$$

then

$$\mu(\{t_1, \ldots, t_{n+1}\}) = \bigcup_{i=1}^{n} \{\mu_1(t_i)\} \cup \{\mu_2(t_{n+1})\} = \mu_1(\{t_1, \ldots, t_n\}) \cup \mu_2(\{t_{n+1}\}) \subseteq G_0$$

and then $\mu \in [\![\{t_1, \ldots, t_n\}]\!]_{G_0}$.

Proof of Proposition 5.9 In the extended version of [1] in Lemma 2 of Appendix A, the following property is proved: If P is composed only by AND, OPT, and FILTER, then for every pair of mappings $\mu_1, \mu_2 \in [\![P]\!]_G$ if μ_1 and μ_2 are compatibles then $\mu_1 = \mu_2$. We will use Lemma 2 in this proof.

Let $\mathcal{D}$ be a dataset with G_0 as default graph. The proof follows by induction in the construction of P. If P is a basic graph pattern then by definition, every $\mu \in [\![P]\!]^{\mathcal{D}} = [\![P]\!]_{G_0}$ has cardinality 1. Suppose first that $P = (P' \text{ FILTER } (R))$ and $\mu \in [\![P]\!]^{\mathcal{D}}$, then $\mu \in [\![P']\!]^{\mathcal{D}}$ because $[\![P]\!]^{\mathcal{D}} \subseteq [\![P']\!]^{\mathcal{D}}$ and then applying the induction hypothesis we have that μ has cardinality 1. Suppose now that

$P = (P_1 \text{ AND } P_2)$ and $\mu \in [\![P]\!]^{\mathcal{D}}$, then $\mu = \mu_1 \cup \mu_2$ for $\mu_1 \in [\![P_1]\!]^{\mathcal{D}}$ and $\mu_2 \in [\![P_2]\!]^{\mathcal{D}}$. By the induction hypothesis the cardinality of μ_1 and of μ_2 is 1, and if $\mu = \mu_1' \cup \mu_2'$ with $\mu_1' \in [\![P_1]\!]^{\mathcal{D}}$, $\mu_2' \in [\![P_2]\!]^{\mathcal{D}}$ we have that μ_1 and μ_1' are compatibles and then (by Lemma 2) $\mu_1 = \mu_1'$, similarly $\mu_2 = \mu_2'$, and then the cardinality of μ is 1. Suppose now that $P = (P_1 \text{ OPT } P_2)$ and $\mu \in [\![P]\!]^{\mathcal{D}}$, then by definition $\mu \in [\![(P_1 \text{ AND } P_2)]\!]^{\mathcal{D}}$ or $\mu \in [\![P_1]\!]^{\mathcal{D}} \smallsetminus [\![P_2]\!]^{\mathcal{D}}$, but μ is not simultaneously in both sets. If $\mu \in [\![(P_1 \text{ AND } P_2)]\!]^{\mathcal{D}}$ we already showed that the cardinality of μ is 1. If $\mu \in [\![P_1]\!]^{\mathcal{D}} \smallsetminus [\![P_2]\!]^{\mathcal{D}}$ then by definition the cardinality of μ with respect to $[\![P]\!]^{\mathcal{D}}$ is the same as the cardinality of μ in $[\![P_1]\!]^{\mathcal{D}}$ which is 1 by the induction hypothesis.