A core SPARQL fragment

Daniel Hernandez

About this document. This is an extract of a thesis I submitted to the University of Chile, in pursuance of a PhD in December 2019. This document summarizes the core algebraic fragment of SPARQL I study throughout the thesis. I made this document available to facilitate the reuse of this fragment in other works.

The *Resource Description Framework* (RDF) [17, 11, 12] was promoted by the W3C as the main data model to share data on the Web. The initial idea was to publish RDF documents similarly as previously was done with HTML. RDF documents are essentially a set of claims about resources. Resources are named with Web identifiers (URLs). Data about resources can be retrieved with the Web protocol (HTTP) by sending a request on the resource identifier. With this design, the Web infrastructure for documents serves also to build the Web of Data.

After the standardization of the RDF data model, several query languages were proposed [3] for the RDF data model, including RQL [5], SquishQL [15], NautiLOD [8], and SPARQL [20, 10], among others. SPARQL became a W3C Recommendation in 2008, and it is still considered the standard language for querying RDF data.

This chapter reviews the RDF data model (Section 1) and the SPARQL query language (Section 2). To simplify the study of SPARQL, we use the algebraic formalization by Perez et al. [18]. Their formalization has some differences with the standard SPARQL algebra, but as Angles and Gutierrez [1], and Kaminski et al. [14] showed, the differences do not imply a different expressive power.

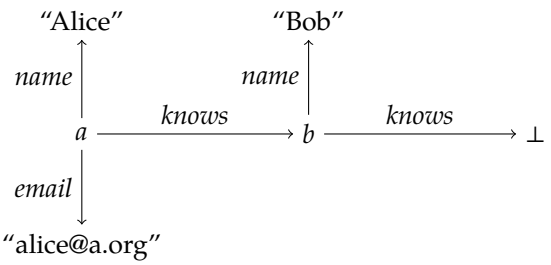
1 The RDF data model

RDF is based on a special type of labeled digraph called RDF graph. The structure of RDF graphs is stated by the following definition.

Definition 1 (RDF Syntax)**.** *We assume three disjoints sets* $\mathbf{I}$, $\mathbf{B}$ *and* $\mathbf{L}$ *of identifiers, blank nodes and literals, respectively. An RDF graph G is a set of triples* (s, p, o) *in the universe* $(\mathbf{I} \cup \mathbf{B}) \times \mathbf{I} \times (\mathbf{I} \cup \mathbf{B} \cup \mathbf{L})$ *where s is called the subject, p is called the predicate, and o is called the object.*

Notation. We denote identifiers as words with one or more letters (e.g., *a*, *b*, *c*, *name*, *email*, and *knows*). We denote literals with quotation marks (e.g., "Alice" and "Bob"), except for literals codifying numbers where we omit the quotation marks (e.g., 1, 2, and 3). We denote blank nodes using the symbol $\perp$ with a numeric subscript when there is more than one blank node (e.g., $\perp_1$, $\perp_2$, and $\perp_3$).

Example 1. *The following figure depicts an RDF graph:*



Each triple (s, p, o) *of the RDF graph is depicted as an arc from* s *to* o *with label* p*. In this graph, elements* "Alice", "Bob" *and "alice@a.org" are literals; elements* a, b, *name, knows, and email are identifiers; and* $\perp$ *is a blank node.*

Since RDF graphs are sets of RDF triples, we use the operators $\cup$, $\cap$, and $\setminus$ between RDF graphs, with their standard meaning for sets.

Definition 2 (RDF vocabulary)**.** *An RDF vocabulary (or simply a vocabulary) is a finite subset of* $\mathbf{I} \cup \mathbf{L}$*. Given an RDF graph* G*, the vocabulary of* G*, denoted* $\text{voc}(G)$*, is the set of all identifiers and literals occurring in* G*.*

Example 2. *If* G *is the graph of Example 1 then* $\text{voc}(G)$ *is composed of all labels of nodes and arcs occurring in* G*, except the blank node* $\perp$*.*

We next present the semantics of RDF graphs as defined by Hogan et al. [13] based in the definition by Hayes [11]. We will omit datatype interpretations, and the use of RDF vocabularies with predefined semantics (e.g., RDFS [4] and OWL [16]) as they are not directly concerned with the subject of this thesis.

Definition 3 (Semantics of ground triples [13])**.** *An interpretation* $\mathcal{A}$ *over a vocabulary* V *is a tuple* (R, P, E, I) *such that* R *is a non-empty set, called the domain, or the universe or the resources of* $\mathcal{A}$*;* P *is a set of (not necessarily disjoint from or a subset of* R*) called the properties of* $\mathcal{A}$*;* $E : P \to 2^{R \times R}$ *is a mapping that assigns an extension, denoted* $p^{\mathcal{A}}$*, to each property* $p \in P$*; and* $I : V \to R \cup P$ *is the interpretation mapping that assigns a resource or a property to each element of* V *such that* I *is the identity for literals.*

Given a vocabulary V*, an interpretation* $\mathcal{A} = (R, P, E, I)$ *over* V*, and a triple* $(s, p, o) \in V^3$*, we say that* $\mathcal{A} \models (s, p, o)$ *if and only if* $I(p) \in P$ *and* $(s, o) \in E(p)$*.*

Observe that Definition 3 gives a special status to literals. Since I is the identity function for literals, then R includes all literals occurring in V. On the other hand, the definition imposes no restriction to interpretations of an RDF graph regarding the resource associated to an IRI. This restriction comes from the RDF 1.0 specification [11, §1.4] that states that *'if* E *is a plain literal* `"aaa"` *in* V *then* $I(E) =$ `aaa`*.'* This restriction is not present in the RDF 1.1 specification [12, §], where there is a different treatment for literals. These nuances are not relevant for this thesis.

Definition 3 provides a semantics for RDF triples without blank nodes. In order to define the semantics of RDF graphs with blank nodes we have to consider a function that maps blank nodes to elements in $\mathbf{I} \cup \mathbf{L}$.

Definition 4 (Semantics of an RDF graph)**.** *Let* G *be an RDF graph,* V *be a vocabulary such that* $V \supseteq \text{voc}(G)$*,* $\mathcal{A} = (R, P, E, I)$ *be an interpretation over* V*,* $v : \mathbf{B} \to R$ *be a function, and* I_v *denote an amended version of* I *that includes* $\mathbf{B}$ *as part of its domain such that* $I_v(b) = v(b)$ *for* $b \in \mathbf{B}$ *and* $I_v(x) = I(x)$ *for* $x \in \mathbf{I} \cup \mathbf{L}$*. We say that* $\mathcal{A}$ *is a model of* G*, denoted* $\mathcal{A} \models G$*, if there exists a function* v *such that for each* $(s, p, o) \in G$*, it holds that* $\mathcal{A} \models (I_v(s), I_v(p), I_v(o))$*.*

Example 3. *Let* G *be the following RDF graph:*

$$a \xrightarrow{\quad knows \quad} b \xrightarrow{\quad knows \quad} \perp$$

Then, interpretations $\mathcal{A}_1$ to $\mathcal{A}_4$ below are models of G.

$$\mathcal{A}_1 = (\ \{1,2,3\},\quad \{p\},\quad \{p \mapsto \{(1,2),(2,3)\}\},\quad \{a \mapsto 1, b \mapsto 2, \perp \mapsto 3, knows \mapsto p\}\ )$$
$$\mathcal{A}_2 = (\ \{1\},\quad \{p\},\quad \{p \mapsto \{(1,1)\}\},\quad \{a \mapsto 1, b \mapsto 1, \perp \mapsto 1, knows \mapsto p\}\ )$$
$$\mathcal{A}_3 = (\ \{1,2,3\},\quad \{p\},\quad \{p \mapsto \{(1,2),(2,3),(3,1)\}\},\quad \{a \mapsto 1, b \mapsto 2, \perp \mapsto 3, knows \mapsto p\}\ )$$
$$\mathcal{A}_4 = (\ \{1,2\},\quad \{p,q\},\quad \{p \mapsto \{(1,1)\}, q \mapsto \{(1,2)\}\},\quad \{a \mapsto 1, b \mapsto 1, \perp \mapsto 1, knows \mapsto p\}\ )$$

The semantics of an RDF graph G is defined in terms of the interpretations of G. This approach to define the semantics of a data-model in terms of finite structures or models is called model-theoretic. In this thesis we assume familiarity with the model-theoretic and the proof-theoretic semantics of relational databases. An introduction of both approaches is given by Reiter [23]. RDF graphs have the following differences with the usual model-theoretic and proof-theoretic semantics given to relational databases:

1. Interpretations of RDF graphs allow predicates to occur as elements in relations. The main reason of this design is that RDF aims to describe not only resources (i.e., the elements of the relations), but also notions as predicates and classes used to describe the resources.

2. Interpretations of an RDF graph are not bounded to a specific "relational" vocabulary as the models of a relation do. In fact, the interpretations of an RDF graph G may contain relations that are not referred by the predicates in G. For instance, in Example 3, predicate q of interpretation $\mathcal{A}_4$ does not represent a predicate in the graph G. The RDF model is designed for the Web. In this context we cannot assume that people will get a consensus over a fixed set of predicates to be used to describe the world.

The notions of entailment and equivalence between RDF graphs are defined as usual.

Definition 5 (Simple entailment)**.** *An RDF graph G_1 entails an RDF graph G_2, denoted $G_1 \models G_2$, if and only if every interpretation over the vocabulary of $G_1 \cup G_2$ which satisfies G_1 also satisfies G_2. We say that two RDF graphs G_1 and G_2 are equivalent, denoted $G_1 \equiv G_2$, if and only if $G_1 \models G_2$ and $G_2 \models G_1$.*

A notion that is closely related with entailment and the equivalence of RDF graphs is the notion of map between graphs.

Definition 6 (Map between RDF graphs)**.** *A map $h : \mathbf{I} \cup \mathbf{B} \cup \mathbf{L} \to \mathbf{I} \cup \mathbf{B} \cup \mathbf{L}$ is a function preserving elements in $\mathbf{I} \cup \mathbf{L}$, i.e., $h(x) = x$ if $x \in \mathbf{I} \cup \mathbf{L}$. Given a graph G and a map h, $h(G)$ denotes the RDF graph $\{(h(s), h(p), h(o)) \mid (s,p,o) \in G\}$. Two RDF graphs G_1 and G_2 are said isomorphic if and only if there are maps h_1 and h_2 such that $h_1(G_1) = G_2$ and $h_2(G_2) = G_1$.*

We overloaded the meaning of map to speak of a map $h : G_1 \to G_2$ if and only if G_1 and G_2 are RDF graphs, h is a map, and $h(G_1) \subseteq G_2$.

So far, we have reviewed the standard notions of entailment and map among RDF graphs. It is well-known that these notions are equivalent (see [6, 12, 9]). This equivalence is formalized by the following theorem:

Theorem 1 ([6, 12, 9])**.** *Let G_1 and G_2 be two RDF graphs. Then, $G_1 \models G_2$ if and only if there is a map $h : G_2 \to G_1$.*

In this thesis we assumed familiarity with the Reiter [21] *proof-theoretic* semantics of the relational model extended with null values, whose theory codifies three assumptions over a relational database, namely the *closed-domain*, the *unique-name*, and the *closed-world* assumptions. We next discus the differences among the relational model and the RDF model regarding these assumptions.

1. *No Unique Name Assumption:* As is exemplified by the model $\mathcal{A}_2$ of Example 3, different identifiers and blank nodes may refer to the same resource (in this case a, b, and $\perp$ refer to resource 1). The lack of the unique name assumption is motivated by the distributed nature of the Web. Data publishers in different parts of the world may use different identifiers to refer to the same resource.

2. *No Closed World Assumption:* Let G be an RDF graph and (s, p, o) be a triple without blank nodes that is not in G. Then, there exists an interpretation $\mathcal{A}$ of G such that $\mathcal{A} \models (s, p, o)$. To see this, let G_1 and G_2 be two RDF graphs where $G_2 = G_1 \cup \{(s, p, o)\}$ and $(s, p, o) \notin G_1$. Since there exists a map $h : G_1 \to G_2$, by Theorem 1, it holds that $G_2 \models G_1$. Under the semantics provided by Definition 4, all RDF graphs G are satisfiable because a triple in G cannot express information that contradicts other triples in G. Since RDF graphs are satisfiable, there exists an interpretation $\mathcal{A}$ such that $\mathcal{A} \models G_2$. Since $G_2 \models G_1$, by definition, $\mathcal{A} \models G_1$. Since $(s, p, o) \in G_2$, it holds that $\mathcal{A} \models (s, p, o)$. Thus, for every graph G and triple (s, p, o) there exists an interpretation $\mathcal{A}$ of G such that $\mathcal{A} \models (s, p, o)$. Hence, no negative information can be inferred from RDF graphs, and thus RDF graphs do not follow the closed world assumption in the sense of Reiter [22].

2 The SPARQL query language

SPARQL is a query language designed to query RDF graphs. In this section we present the core fragment of the SPARQL language for consideration in this thesis. This fragment is essentially the fragment studied by Perez et al. [18]. We do not include the clauses FROM and GRAPH as they do because those features are not related to the focus of this thesis, but we add the clauses VALUES, BIND and EXISTS.

In order to define the syntax and semantics of SPARQL, we need to introduce some notions. We assume a countable infinite set $\mathbf{V}$, called the set of variables, and a set $\mathbf{F}$, called the set of functions, that consists in functions of the form $f : (\mathbf{I} \cup \mathbf{B} \cup \mathbf{L} \cup \{\varnothing\})^n \to \mathbf{I} \cup \mathbf{B} \cup \mathbf{L} \cup \{\varnothing\}$, where sets $\mathbf{I}$, $\mathbf{B}$, and $\mathbf{L}$ are the same sets that in Definition 1, sets $\mathbf{V}$, $\mathbf{F}$, $\mathbf{I}$, $\mathbf{B}$, and $\mathbf{L}$ are pairwise disjoint, and $\varnothing$, called the unbound value, is an element that is not in the set $\mathbf{I} \cup \mathbf{B} \cup \mathbf{L}$. The prefix "?" is used to denote variables (e.g., ?x). A *SPARQL mapping* (or simply a *mapping* when no confusion arises) is a partial function $\mu : \mathbf{V} \to \mathbf{I} \cup \mathbf{B} \cup \mathbf{L}$. Two mappings μ_1 and μ_2 are said to be compatible, denoted $\mu_1 \sim \mu_2$ if and only if for every common variable X holds $\mu_1(X) = \mu_2(X)$. Given two compatible mappings μ_1 and μ_2, the join of μ_1 and μ_2, denoted $\mu_1 \smile \mu_2$, is the mapping with domain $\mathrm{dom}(\mu_1) \cup \mathrm{dom}(\mu_2)$ that is compatible with μ_1 and μ_2. Given a mapping μ and a finite set $\mathcal{X}$ of variables, we write $\mu_{|\mathcal{X}}$ to denote the mapping μ restricted to the domain $\mathcal{X}$, that is, $\mu_{|\mathcal{X}} \sim \mu$ and $\mathrm{dom}(\mu_{|\mathcal{X}}) = \mathrm{dom}(\mu) \cap \mathcal{X}$.

Throughout this thesis we sometimes use the translation of sets of mappings to relations used by Cyganiak [7] and Polleres [19], among others. This consists in viewing mappings as tuples under the named perspective. However, as mappings in a set Ω of mappings may have different domains, we need to choose one for the relation associated to Ω. The set of variables $\mathcal{X}$ of the chosen domain must be big enough to contain the domains of all mappings in Ω. Then, for each variable $?x \in \mathcal{X}$, we extend ("fill" in) all mappings μ where $?x \notin \mathrm{dom}(\mu)$ defining $\mu(?x) = \varnothing$. The result of this procedure is a relation in the domain $\mathbf{I} \cup \mathbf{B} \cup \mathbf{L} \cup \{\varnothing\}$. This procedure is formalized as follows:

Definition 7 (Filled mapping)**.** *Given a SPARQL mapping μ with* $\mathrm{dom}(\mu) = \mathcal{X}$ *and a set of variables $\mathcal{Y}$ such that $\mathcal{X} \subseteq \mathcal{Y}$, the filling of μ over $\mathcal{Y}$, denoted* $\mathrm{fill}(\mu, \mathcal{Y})$*, is the mapping such that* $\mathrm{dom}(\mathrm{fill}(\mu, \mathcal{Y})) = \mathcal{Y}$ *and*

$$\mathrm{fill}(\mu, \mathcal{Y})(?\mathrm{x}) = \begin{cases} \mu(?\mathrm{x}) & \text{if } ?\mathrm{x} \in \mathrm{dom}(\mu), \\ \varnothing & \text{otherwise.} \end{cases}$$

Now we are ready to define the SPARQL syntax.

Definition 8 (SPARQL syntax)**.** *The set of SPARQL queries (or simply queries) is defined recursively as follows:*

- *An element of* $(\mathbf{I} \cup \mathbf{B} \cup \mathbf{V}) \times (\mathbf{I} \cup \mathbf{V}) \times (\mathbf{I} \cup \mathbf{B} \cup \mathbf{L} \cup \mathbf{V})$ *is a triple pattern. A set of triple patterns is a query—called a basic graph pattern.*
- *If Q_1, Q_2 are queries, then:*
  - $(Q_1$ UNION $Q_2)$ *is a query—called a UNION query.*
  - $(Q_1$ AND $Q_2)$ *is a query—called an AND query.*
  - $(Q_1$ OPT $Q_2)$ *is a query—called an OPT query.*
  - $(Q_1$ MINUS $Q_2)$ *is a query—called a MINUS query.*
- *If Q is a query and $\mathcal{X} \subset \mathbf{V}$ is a finite set of variables, then* (SELECT $\mathcal{X}$ WHERE Q) *is a query—called a SELECT query.*
- *If Q is a query and φ is a SPARQL buit-in condition (see below), then* (Q FILTER φ) *is a query—called a FILTER query.*
- *If $\mathcal{X} \subset \mathcal{X}$ is a finite set of variables, and Ω is a set of SPARQL mappings such that* $\mathrm{dom}(\mu) \subseteq \mathcal{X}$ *for each $\mu \in \Omega$, then* (VALUES $\mathcal{X}$ Ω) *is a query—called a VALUES query.*
- *If Q is a query, $f : (\mathbf{I} \cup \mathbf{B} \cup \mathbf{L} \cup \{\varnothing\})^n \to (\mathbf{I} \cup \mathbf{B} \cup \mathbf{L} \cup \{\varnothing\})$ is a function in $\mathbf{F}$, and $?\mathrm{y}, ?\mathrm{x}_1, \ldots, ?\mathrm{x}_n$ are variables such that $?\mathrm{y}$ does not occur in Q nor in $\{?\mathrm{x}_1, \ldots, ?\mathrm{x}_n\}$, then* ($Q$ BIND $f(?\mathrm{x}_1, \ldots, ?\mathrm{x}_n)$ AS $?\mathrm{y}$) *is a query—called a BIND query.*
- *A SPARQL built-in condition (or symply a filter-condition) is defined recursively as follows:*
  - *An equality $t_1 = t_2$, where t_1, t_2 are elements of $\mathbf{I} \cup \mathbf{B} \cup \mathbf{L} \cup \mathbf{V}$, is a filter-condition.*
  - *If t is an element of $\mathbf{I} \cup \mathbf{B} \cup \mathbf{L} \cup \mathbf{V}$ then* isBlank(t) *is a filter-condition.*
  - *If* $?\mathrm{x}$ *is a variable then* bound($?\mathrm{x}$) *is a filter-condition.*
  - *A Boolean combination of filter-conditions (with operators $\wedge$, $\vee$, and $\neg$) is a filter-condition.*

*We will call this SPARQL fragment AMUSFOVB capturing the initial letters of the operations: **A**ND, **M**INUS, **U**NION, **S**ELECT, **F**ILTER, **O**PT, **V**ALUES, and **B**IND, applied over basic graph patterns. We will name subsets of this fragment by removing the corresponding initials. For instance, the fragment considering only operators **A**ND, **S**ELECT, and **O**PT will be denoted ASO.*

Historically, SPARQL *queries* and *graph patterns* were two different concepts. Polleres [19] defines a SPARQL query as a quadruple (V, P, DS, SM) where V is the result form, P is a graph pattern, DS is a dataset, and SM is a set of solution modifiers. In this thesis we ignore datasets and solution modifiers, so

according to Polleres notation, a query is simple a pair (V, P). Furthermore, in this thesis we restrict result forms to the SELECT clause. Thus, we only study SELECT queries.

Note: Since in SPARQL 1.1 graph patterns admit SELECT queries as graph patterns, there is no need to consider SPARQL queries and graph patterns as different concepts. Thus, throughout this thesis we will use both terms with the same meaning.

Before presenting the semantics of SPARQL we have to discuss *set* and *multiset semantics*. A multiset is a modification of the concept of set, that unlike sets, allows repeated elements. In SPARQL 1.0 a SELECT query has either the form (SELECT $\mathcal{X}$ WHERE P) or (SELECT DISTINCT $\mathcal{X}$ WHERE P) where P is a graph pattern without another SELECT query inside. In the former case the query returns multisets, and in the latter case returns sets. Thus, there were two alternative semantics for graph patterns, namely set and multiset semantics. Since SPARQL 1.1 admits SELECT queries as graph patterns, queries can be evaluated as a combination of both semantics. The theory of multisets is complex (see the work of Angles and Gutierrez [2]). For the sake of the simplicity, in this thesis we will focus on the *set semantics* of SPARQL (where repeated elements are removed) as in done by Perez et al. [18].

According to Perez et al. [18], the semantics of SPARQL is defined using operations over sets of mappings. These operators are presented in the following definition:

Definition 9 (Algebra of mappings [18])**.** *Let Ω_1 and Ω_2 be two sets of mappings. Then, the operators $\bowtie$, $\cup$, $-$, and $\ltimes$ are defined over sets of mappings as follows:*

$$\Omega_1 \bowtie \Omega_2 = \{\mu_1 \backsim \mu_2 \mid \mu_1 \in \Omega_1, \mu_2 \in \Omega_2, \mu_1 \sim \mu_2\},$$
$$\Omega_1 \cup \Omega_2 = \{\mu \mid \mu \in \Omega_1 \text{ or } \mu \in \Omega_2\},$$
$$\Omega_1 - \Omega_2 = \{\mu_1 \mid \mu_1 \in \Omega_1, \text{ and there does not exist } \mu_2 \in \Omega_2, \mu_1 \sim \mu_2\},$$
$$\Omega_1 \ltimes \Omega_2 = (\Omega_1 \bowtie \Omega_2) \cup (\Omega_1 - \Omega_2).$$

Given a query Q we write $\text{var}(Q)$ to denote the set of variables occurring in Q. We use this notation also for filter-conditions, i.e., $\text{var}(\varphi)$ are the variables occurring in the filter-condition φ.

Now we are ready to present the semantics of SPARQL.

Definition 10 (SPARQL semantics)**.** *Let G be a RDF graph, and Q be a SPARQL query. Then, the evaluation of Q over G, denoted $[\![Q]\!]_G$, is the set of mappings recursively defined as follows:*

- *If Q is a basic graph pattern, then $[\![Q]\!]_G$ is the set of all mappings $\mu|_{\text{var}(Q)}$ such that $\mu(Q') \subseteq G$, where Q' is the result of replacing all blank nodes in Q by fresh variables, and μ is a mapping with $\text{dom}(\mu) = \text{var}(Q')$.*
- *If Q is (Q_1 AND Q_2) then $[\![Q]\!]_G = [\![Q_1]\!]_G \bowtie [\![Q_2]\!]_G$.*
- *If Q is (Q_1 UNION Q_2) then $[\![Q]\!]_G = [\![Q_1]\!]_G \cup [\![Q_2]\!]_G$.*
- *If Q is (Q_1 MINUS Q_2) then $[\![Q]\!]_G = [\![Q_1]\!]_G - [\![Q_2]\!]_G$.*
- *If Q is (Q_1 OPT Q_2) then $[\![Q]\!]_G = [\![Q_1]\!]_G \ltimes [\![Q_2]\!]_G$.*
- *If Q is (SELECT $\mathcal{X}$ WHERE Q_1) then $[\![Q]\!]_G$ is the set of mappings $\mu|_{\mathcal{X}}$ such that $\mu \in [\![Q_1]\!]_G$.*
- *If Q is (VALUES $\mathcal{X}$ Ω) then $[\![Q]\!]_G$ is the set of mappings Ω.*
- *If Q is (Q_1 FILTER φ) then $[\![Q]\!]_G$ is the set of mappings $\mu \in [\![Q_1]\!]_G$ such that $\mu(\varphi)$ is true (see Definition 11).*

- *If Q is* $(Q_1 \text{ BIND } f(?x_1, \ldots, ?x_n) \text{ AS } ?y)$ *then let μ' be the mapping* $\text{fill}(\mu, \{?x_1, \ldots, ?x_n\})$*, and y_μ be the value $f(\mu'(?x_1), \ldots, \mu'(?x_n))$. Then, $[\![Q]\!]_G$ is the set of SPARQL mappings*

$$\{\mu \in [\![Q_1]\!]_G \mid y_\mu = \varnothing\} \cup \{\mu \smile \{?y \mapsto y_\mu\} \mid \mu \in [\![Q_1]\!]_G \text{ and } y_\mu \neq \varnothing\}.$$

A filter-condition φ drops all mappings μ where $\mu(\varphi)$ is not true. The following definition defines the truth value of $\mu(\varphi)$ for filter-conditions.

Definition 11 (Semantics of SPARQL filter-conditions)**.** *Let μ be a finite mapping from variables to elements in $\mathbf{I} \cup \mathbf{B} \cup \mathbf{L}$, φ be a SPARQL filter-condition without occurrences of EXISTS clauses, and t_1, t_2 be elements in $\mathbf{V} \cup \mathbf{I} \cup \mathbf{B} \cup \mathbf{L}$. Let $\nu : \mathbf{V} \cup \mathbf{I} \cup \mathbf{B} \cup \mathbf{L} \to \mathbf{V} \cup \mathbf{I} \cup \mathbf{B} \cup \mathbf{L}$ be the function defined as*

$$\nu(t) = \begin{cases} \mu(t) & \text{if } t \in \text{dom}(\mu), \\ \varnothing & \text{if } t \in \mathbf{V} \setminus \text{dom}(\mu), \\ t & \text{if } t \notin \mathbf{V}. \end{cases}$$

The truth value of $\mu(\varphi)$ is defined recursively as follows:

- *If φ is an equality $t_1 = t_2$ then:*
  1. *$\mu(\varphi)$ is error if $\nu(t_1) = \varnothing$ or $\nu(t_1) = \varnothing$.*
  2. *$\mu(\varphi)$ is true if $\nu(t_1) \neq \varnothing$, $\nu(t_1) \neq \varnothing$, and $\nu(t_1) = \nu(t_2)$.*
  3. *$\mu(\varphi)$ is false if $\nu(t_1) \neq \varnothing$, $\nu(t_1) \neq \varnothing$, and $\nu(t_1) \neq \nu(t_2)$.*
- *If φ is an expression of the form* $\text{bound}(?x)$ *then $\mu(\varphi)$ is true if $?x \in \text{dom}(\mu)$. Otherwise, $\mu(\varphi)$ is false.*
- *If φ is an expression of the form* $\text{isBlank}(t_1)$ *then:*
  1. *$\mu(\varphi)$ is error if $\nu(t) = \varnothing$.*
  2. *$\mu(\varphi)$ is true if $\nu(t) \neq \varnothing$ and $\nu(t) \in \mathbf{B}$.*
  3. *$\mu(\varphi)$ is false if $\nu(t) \neq \varnothing$ and $\nu(t) \notin \mathbf{B}$.*
- *If φ is a Boolean combination of conditions using operators $\wedge$, $\vee$ and $\neg$, then the truth value of $\mu(\varphi)$ is the usual for 3-valued logic (where error is interpreted as unknown).*

We write $\mu \models \varphi$ to mean that φ is true over μ.

The following definition of $\text{dom}(Q)$ corresponds to the set of variables that in the specification [10, §18.2.1] are called the *in-scope* variables of a query Q. Despite the fact that the output of a SPARQL is not a relation because its mappings can have different domains (e.g., when Q is a union of queries with different variables), most engines use $\text{dom}(Q)$ as the attributes of the "relation" that result of evaluating Q.

Definition 12 (Domain of a SPARQL query)**.** *Let (s, p, o) be a triple pattern, Q_1, Q_2 be SPARQL queries, φ be a filter condition, and $\mathcal{X}$ be a set of variables. The domain of a SPARQL query Q, denoted* $\text{dom}(Q)$*, is defined recursively*

as follows:

$$\begin{aligned}
\mathrm{dom}((s,p,o)) &= \mathrm{var}((s,p,o)),\\
\mathrm{dom}(Q_1 \text{ AND } Q_2) &= \mathrm{dom}(Q_1) \cup \mathrm{dom}(Q_2),\\
\mathrm{dom}(Q_1 \text{ UNION } Q_2) &= \mathrm{dom}(Q_1) \cup \mathrm{dom}(Q_2),\\
\mathrm{dom}(Q_1 \text{ MINUS } Q_2) &= \mathrm{dom}(Q_1),\\
\mathrm{dom}(Q_1 \text{ FILTER } \varphi) &= \mathrm{dom}(Q_1),\\
\mathrm{dom}(Q_1 \text{ OPT } Q_2) &= \mathrm{dom}(Q_1) \cup \mathrm{dom}(Q_2),\\
\mathrm{dom}(\text{SELECT } \mathcal{X}\ Q_1) &= \mathcal{X},\\
\mathrm{dom}(\text{VALUES } \mathcal{X}\ \Omega) &= \mathcal{X},\\
\mathrm{dom}(Q_1 \text{ BIND}(f(?x_1,\ldots,?x_n) \text{ AS } ?y)) &= \mathrm{dom}(Q_1) \cup \{?y\}.
\end{aligned}$$

References

[1] Renzo Angles and Claudio Gutiérrez. "Negation in SPARQL". In: *AMW*. Vol. 1644. CEUR Workshop Proceedings. CEUR-WS.org, 2016.

[2] Renzo Angles and Claudio Gutiérrez. "The Multiset Semantics of SPARQL Patterns". In: *International Semantic Web Conference (1)*. Vol. 9981. LNCS. 2016, pp. 20–36.

[3] James Bailey et al. "Web and Semantic Web Query Languages: A Survey". In: *Reasoning Web, First International Summer School 2005, Msida, Malta, July 25-29, 2005, Tutorial Lectures*. Vol. 3564. LNCS. Springer, 2005, pp. 35–133.

[4] Dan Brickley and R.V. Guha. *RDF Schema 1.1*. W3C Recommendation. Feb. 2014.

[5] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. "Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema". In: *The Semantic Web - ISWC 2002, First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002, Proceedings*. Vol. 2342. LNCS. Springer, 2002, pp. 54–68.

[6] Ashok K. Chandra and Philip M. Merlin. "Optimal Implementation of Conjunctive Queries in Relational Data Bases". In: *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*. ACM, 1977, pp. 77–90.

[7] Richard Cyganiak. "A relational algebra for SPARQL". In: *Digital Media Systems Laboratory HP Laboratories Bristol. HPL-2005-170* (2005), p. 35.

[8] Valeria Fionda, Giuseppe Pirrò, and Claudio Gutiérrez. "NautiLOD: A Formal Language for the Web of Data Graph". In: *TWEB* 9.1 (2015), 5:1–5:43.

[9] Claudio Gutierrez et al. "Foundations of Semantic Web databases". In: *J. Comput. Syst. Sci.* 77.3 (2011), pp. 520–541.

[10] Steve Harris, Andy Seaborne, and Eric Prud'hommeaux. *SPARQL 1.1 Query Language*. W3C Recommendation. 2013.

[11] Patrick Hayes. *RDF Semantics*. W3C Recommendation. Feb. 2004.

[12] Patrick Hayes and Peter F. Patel-Schneider. *RDF 1.1 Semantics*. W3C Recommendation. Feb. 2014.

[13] Aidan Hogan et al. “Everything you always wanted to know about blank nodes”. In: *J. Web Sem.* 27 (2014), pp. 42–69.

[14] Mark Kaminski, Egor V. Kostylev, and Bernardo Cuenca Grau. “Query Nesting, Assignment, and Aggregation in SPARQL 1.1”. In: *ACM Trans. Database Syst.* 42.3 (2017), 17:1–17:46.

[15] Libby Miller, Andy Seaborne, and Alberto Reggiori. “Three Implementations of SquishQL, a Simple RDF Query Language”. In: *The Semantic Web - ISWC 2002, First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002, Proceedings*. Ed. by Ian Horrocks and James A. Hendler. Vol. 2342. LNCS. Springer, 2002, pp. 423–435.

[16] Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition)*. W3C Recommendation. Dec. 2012.

[17] Ralph R. Swick Ora Lassila. *Resource Description Framework (RDF) Model and Syntax Specification*. W3C Recommendation. Feb. 1999.

[18] Jorge Pérez, Marcelo Arenas, and Claudio Gutiérrez. “Semantics and complexity of SPARQL”. In: *ACM Trans. Database Syst.* 34.3 (2009), 16:1–16:45.

[19] Axel Polleres. “From SPARQL to rules (and back)”. In: *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*. Ed. by Carey L. Williamson et al. ACM, 2007, pp. 787–796.

[20] Eric Prud’hommeaux and Andy Seaborne. *PARQL Query Language for RDF*. W3C Recommendation. 2008.

[21] Raymond Reiter. “A Sound and Sometimes Complete Query Evaluation Algorithm for Relational Databases with Null Values”. In: *J. ACM* 33.2 (Apr. 1986), pp. 349–370. ISSN: 0004-5411.

[22] Raymond Reiter. “On Closed World Data Bases”. In: *Logic and Data Bases*. Advances in Data Base Theory. New York: Plemum Press, 1977, pp. 55–76.

[23] Raymond Reiter. “Towards a Logical Reconstruction of Relational Database Theory”. In: *On Conceptual Modelling (Intervale)*. 1982, pp. 191–233.