Algebraic Structures for Capturing the Provenance of SPARQL Queries*

Floris Geerts
University of Antwerp
floris.geerts@ua.ac.be

Grigoris Karvounarakis
LogicBlox & ICS-FORTH
grigoris.karvounarakis@logicblox.com

Vassilis Christophides
ICS-FORTH & University of Crete Heraklion
christop@ics.forth.gr

Irini Fundulaki
ICS-FORTH
fundul@ics.forth.gr

ABSTRACT

We show that the evaluation of SPARQL algebra queries on various notions of annotated RDF graphs can be seen as particular cases of the evaluation of these queries on RDF graphs annotated with elements of so-called *spm-semirings*. Spm-semirings extend semirings, used for positive relational algebra queries on annotated relational data, with a new operator to capture the semantics of the non-monotone SPARQL operator `OPTIONAL`. Furthermore, spm-semiring-based annotations ensure that desired SPARQL query equivalences hold when querying annotated RDF. In addition to introducing spm-semirings, we study their properties and provide an alternative characterization of these structures in terms of semirings with an embedded boolean algebra (or seba-structure for short). This characterization allows to construct spm-semirings and to identify a universal object in the class of spm-semirings. Finally, we show that this universal object provides a concise provenance representation and can be used to evaluate SPARQL queries on arbitrary spm-semiring-annotated RDF graphs.

Categories and Subject Descriptors: H.2.1 [**Database Management**]: Logical Design – *Data models*.

General Terms: Theory, Languages.

Keywords: RDF, annotations, provenance models, query languages.

1. INTRODUCTION

In recent years, the W3C Linked Open Data (LOD) Initiative (linkeddata.org) has boosted the publication and interlinkage of massive amounts of scientific, corporate and government data sets on the emerging *Data Web* for open access, in the form of RDF data [21] queried with the SPARQL query language [24]. In such settings, where RDF data is freely *exchanged*, *integrated*, and *materialized* in distributed repositories, it is crucial to be able assess the *quality* of replicated and, possibly, incomplete or uncertain data.

Towards this end, several models for annotated RDF data have been proposed [18, 22, 19, 8], for representing various dimensions of *data quality* such as *trust*, *truth of imprecise information* or *probability* of its validity. In all these cases, when annotated data is transformed through SPARQL queries, one needs to compute appropriate annotations for the query results. For instance, in the case of *trust assessment* [4, 18] the trustworthiness of query results is determined based on the trustworthiness of source datasets from which they were derived. Similarly, for *uncertain* and *fuzzy* data sets, the probabilities of query results are derived based on the probabilities associated with the original data [22, 19].

If source annotations were static and common for all users, this computation could be done during query evaluation [18, 22, 19]. However, in general, different users may have different beliefs about e.g., the *trustworthiness* of specific RDF source triples, and their beliefs may change over time, even when the relationship of query results with the source data remains unchanged. For this reason, *abstract provenance models* [16] are used to capture this relationship along with the query operators that combined source data to derive query results.

In the relational setting, provenance models that are capable of abstracting the query evaluation on annotated relational data, have been put forward for the positive fragment of the relational algebra. In particular, the modeling of annotations by means of *semirings* has shown great promise [16], both as a platform for theoretical study and as a representation employed in systems that record in the hosting repository when the data is imported [15], and use it to compute appropriate annotations for different applications and users at a later time [20].

For annotated RDF data and positive SPARQL queries one can verify, similarly to the relational case, that semirings suffice as the annotation structure [26]. However, when the *non-monotone* SPARQL operator `OPTIONAL` is brought into the picture, it can be easily verified that extensions of semirings have to be considered. Indeed, the SPARQL 1.0 algebra semantics [23] of `OPTIONAL` is defined in terms of a left-outer join that involves a non-monotone `DIFFERENCE` operator.[1] More specifically, for any two RDF data sets G_1 and G_2, $(G_1 \texttt{ OPTIONAL } G_2)$ can be written as $(G_1 \texttt{ AND } G_2)$ `UNION` $(G_1 \texttt{ DIFFERENCE } G_2)$ [23]. Thus, although `DIFFERENCE` is — strictly speaking — not part of the SPARQL 1.0 specification, we add it for convenience.

However, as will be explained in Section 2, the semantics of `DIFFERENCE` and `OPTIONAL` is not the same as that of relational difference and (left-)outer join, respectively. Thus, extensions to the semiring framework intended to cope with the latter [12, 1, 2, 13] cannot be applied directly to capture the provenance

*This work was partially supported by the EU FP7 project IP-601043

EDBT/ICDT '13, March 18 - 22 2013, Genoa, Italy

[1]The algebra described in the SPARQL 1.1 specification [17] also contains a similar operator called Diff.

of SPARQL queries. Recent work [9] suggests that it may be possible to express the SPARQL difference in terms of relational operators, in order to leverage the structure of m-semirings, an extension of semirings for relational queries involving difference [12]. However, while a provenance model based on the so-called universal m-semiring exists, this model does not allow for a concise and simple representation of its expressions [12] and, thus, its practical usability as a provenance model is rather limited.

For these reasons, in this paper we propose a new algebraic structure for capturing the semantics of the SPARQL `DIFFERENCE` (and thus also `OPTIONAL`). More specifically, we identify a set of SPARQL query equivalences that involve `DIFFERENCE` and that hold under both the bag and set semantics, and show that these equivalences also hold when evaluating SPARQL queries on a wide variety of annotated RDF data. Then, we define so-called *spm-semirings*, an extension of semirings with a new operation $\ominus$, based on identities derived from the aforementioned SPARQL equivalences. Furthermore, we show that spm-semirings do have a universal structure that provides a concise representation of the provenance of RDF data and SPARQL queries involved.

The underlying techniques rely on a characterization of spm-semirings in terms of semirings with an embedded boolean algebra, of *seba-structure* for short. This characterization is non-trivial and interesting in its own right. Furthermore, the spm-semiring-based provenance expressions can indeed be used to compute appropriate annotations in a wide variety of application domains. We thus provide a complete picture of SPARQL query evaluation on annotated RDF and propose an abstract provenance model that incorporates non-monotone SPARQL operators, something that is still open for queries with difference in the relational case.

In summary, we make the following contributions:

1. We illustrate that the semantics of SPARQL on various notions of annotated RDF have a great commonality (Section 2). Based on this, we generalize the semantics of SPARQL algebra expressions to RDF data annotated with values from some arbitrary annotation domain K, or K-annotated RDF for short (Section 3). For this purpose, K is equipped with binary operations $\oplus$, $\otimes$ and $\ominus$, and constants 0 and 1, that accommodate all SPARQL algebra operators.

2. We identify a set of SPARQL algebra equivalences (some involving `DIFFERENCE`) that are desirable to hold on K-annotated RDF (Section 4). We show that for these equivalences to hold, $(K, \oplus, \otimes, \ominus, 0, 1)$ must be an spm-semiring, and vice versa. A minimal set of identities defining spm-semirings is provided.

3. An alternative characterization of spm-semirings is given in Section 5, based on semirings with an embedded boolean algebra, or seba-structures for short. We prove the correctness of this characterization and show how it can be used to construct spm-semirings based on semirings commonly used in practice.

4. We identify a universal object in the class of spm-semirings (Section 6), leveraging the characterization in terms of seba-structures, and show that the evaluation of SPARQL queries on spm-semiring annotated RDF factors through the evaluation of RDF annotated with elements in the universal object. The universal object is therefore proposed in Section 7 as provenance model for annotated RDF and SPARQL.

Finally, we compare spm-semirings with related work in the relational and Semantic Web context in Section 8, and conclude with directions for future work in Section 9.

2. QUERYING ANNOTATED RDF

In this section, we provide examples of the evaluation of SPARQL queries on annotated RDF data. We first recall RDF [21] and SPARQL [24], with the standard bag semantics in which we represent multiplicities as annotations. We then observe that, similar to the relational case, the semantics of SPARQL on various forms of annotated RDF have a great commonality.

RDF and SPARQL in a nutshell. RDF is the standard model for representing Semantic Web data as sets of *triples* of the form (*subject*, *predicate*, *object*). Intuitively, each triple contains some information about the resource in its subject. For instance, Table (a) of Fig. 1 shows an example of an RDF triple set, denoted by G, where the columns stand for the corresponding components of each triple. SPARQL is the standard language used to query RDF data. We present the SPARQL semantics based on the algebra of Perez et al. [23]. The operators of this algebra manipulate bags of mappings and include a) unary operators σ and π that correspond to the SPARQL constructs `FILTER` and `SELECT`, respectively, and b) binary operators $\cup$, $\bowtie$, and $\ltimes$ for the SPARQL constructs `UNION`, `AND`, and `OPTIONAL`, respectively. The operator $\ltimes$ can be defined in terms of $\cup$, $\bowtie$ and $\setminus$, the algebraic counterpart of `DIFFERENCE` [23]. The following example illustrates the standard bag semantics of SPARQL queries.

EXAMPLE 1 (BAGS). The evaluation of a triple pattern on a set of triples is a *bag of mappings*. Figure 1 depicts the following:

Table (b) shows the mapping bag Ω derived from the evaluation of the SPARQL pattern $(?x, ?y, ?z)$ over G. Here, $?x$, $?y$ and $?z$ denote variables that are bound to constants in triple patterns. The pattern $(?x, ?y, ?z)$ selects all triples from the RDF graph and the result of its evaluation on G is denoted by $[\![(?x, ?y, ?z)]\!]_G$.

To simplify the presentation, we will use the tabular representation of the mapping bags shown in Table (b), where the first three columns correspond to variables in the mappings and the fourth column (**#**) represents the multiplicity of the mapping. The last two columns (**tr**(**ust**) and **fu**(**zzy**)) can be ignored for now, as can be the gray shaded entries in Tables (h) and (i). We further employ symbols μ_i to identify individual mappings.

Tables (c–e) illustrate the evaluation of the operators σ and π. The output mapping bags are denoted by Ω_1, Ω_2 and Ω_3. For instance, mapping μ_{11} of Ω_3, has two derivations originating from mappings in Ω, namely one by projecting μ_4 and another one by projecting μ_5. The multiplicity of μ_{11} is obtained by *adding* the multiplicities of μ_4 and μ_5.

Table (f) shows the result of $\Omega_4 = \Omega_2 \cup \Omega_3$, where Ω_2 and Ω_3 are shown in Tables (d) and (e), respectively. Note that unlike the relational union, the SPARQL union can be applied on bags of mappings defined on different variables. In Ω_4, mapping μ_{12} has two derivations, both of them originating from Ω_3, while the two derivations of μ_{13} originate from Ω_2 (μ_8) and Ω_3 (μ_{10}). The multiplicity of μ_{13} is obtained by *adding* the multiplicities of μ_8 and μ_{10}.

Table (g) depicts the result of $\Omega_4 \bowtie \Omega_1$. For instance, mapping μ_{16} is derived by joining $\mu_{13} \in \Omega_4$ and $\mu_6 \in \Omega_1$. Mappings can be joined only if they are *compatible* [23]. In our example, μ_6 and μ_{13} are compatible because they agree on their common variable, i.e., they both bind $?y$ to b. The multiplicity of μ_{16} is computed as the *product* of the multiplicities of the two input mappings ($1 \times 2 = 2$).

Table (h) illustrates an example of the difference operator, i.e., $\Omega_4 \setminus \Omega_1$. Note that neither μ_{12} nor μ_{13} in Ω_4 are in the result – recall that we ignore the gray shaded entries for now – because Ω_1 contains a mapping (μ_6) that is compatible with the mappings μ_{12} and μ_{13} in Ω_4. On the other hand, there is no mapping in Ω_1 that is compatible with μ_{14}. As a consequence, μ_{14} appears in the result as μ_{19} in Table (h). Similarly to the union, the difference operator can be applied on bags of mappings defined on different variables.

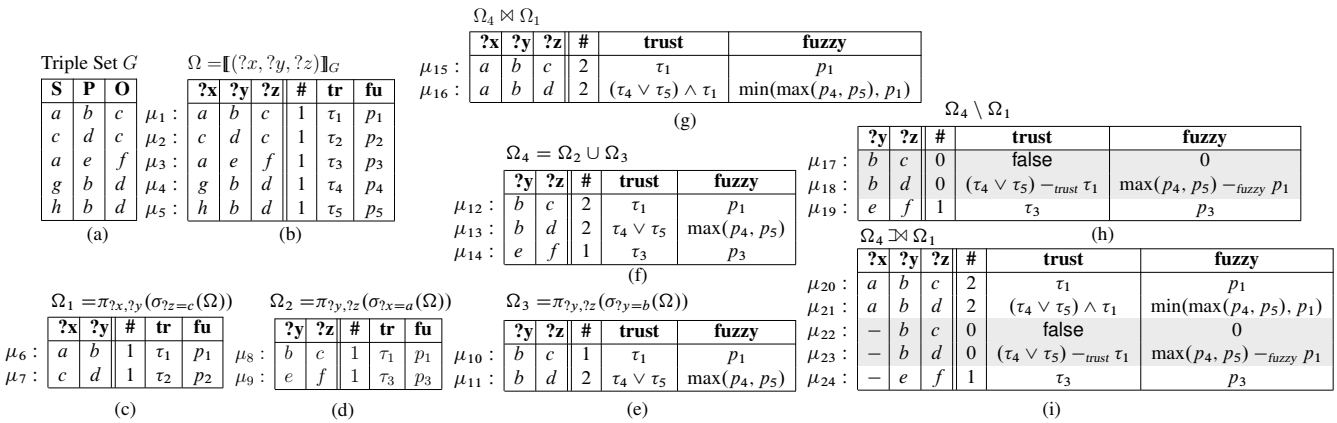

Figure 1: Example of RDF graph and evaluation of SPARQL algebra operators (bag/trust/fuzzy semantics).

Finally, Table (i) depicts the result of $\Omega_4 ⟕ \Omega_1$, which contains all mappings from Tables (g) and (h). Indeed, $\Omega_4 ⟕ \Omega_1 = (\Omega_4 \bowtie \Omega_1) \cup (\Omega_4 \setminus \Omega_1)$. The symbol "$-$" denotes that $?x$ is not bound to a constant in the mappings μ_{22}, μ_{23} and μ_{24}. □

The previous example shows that the multiplicities of mappings in the result of σ, π, $\cup$ and $\bowtie$ SPARQL algebra operators are computed in a similar way as when evaluating the corresponding operators of the relational algebra under bag semantics [16]. In particular, in the case of alternative derivations (e.g., for π or $\cup$) of a mapping, its multiplicity equals the sum of the multiplicities of the different derivations. In the case of $\bowtie$, the multiplicity of the result mapping equals the product of the multiplicities of the two input mappings that were combined.

However, the multiplicities of the result mappings of the $\setminus$ operator are computed differently from the corresponding case of the difference operator ($\setminus_{ra}$) in the relational algebra under bag semantics. Indeed, let R and S be two relations and denote by $R(t)$ and $S(t)$ the multiplicity of a tuple t in R and S, respectively. Then, $(R \setminus_{ra} S)(t) := \max\big(0, R(t) - S(t)\big)$ in the bag semantics of the relational algebra [16]. In contrast, the SPARQL difference ($\setminus$) is defined in terms of compatibility, not equality. Indeed, Table (h) shows that when considering the SPARQL difference $\Omega_4 \setminus \Omega_1$, a tuple (mapping) t in Ω_4 is in the output as long as there is no mapping in Ω_1 that is compatible with it. That is, $(\Omega_4 \setminus \Omega_1)(t) = \Omega_4(t) -_{bag} \Sigma_{t' \sim t} \Omega_1(t')$, where for any two natural number n and m, $n -_{bag} m = n$ in case that $m = 0$, and $n -_{bag} m = 0$ otherwise. Thus, SPARQL follows the standard relational bag semantics for its positive operators but uses a different semantics for $\setminus$. Similarly, relational left-outer join which is defined as a union between a join and a relational difference, also uses a different semantics than SPARQL `OPTIONAL`.

EXAMPLE 2 (BAGS CONT'D). Consider Tables (c), (f) and (h) of Fig. 1. Observe that although μ_{12} and μ_{13} both have multiplicity 2 in Ω_4, and the compatible mappings μ_6 and μ_7 in Ω_1 have multiplicity 1, neither μ_{12} nor μ_{13} is present in $\Omega_4 \setminus \Omega_1$. In contrast, in the relational bag semantics for $\setminus_{ra}$, both μ_{12} and μ_{13} would be in the result with multiplicity 1, as given by $\max(0, 2-1)$. □

SPARQL on annotated RDF. We next consider the semantics of SPARQL when RDF is adorned with trust information [18]. In this setting, for a given SPARQL query, the goal is to find which result mappings are trusted based on the trustworthiness of the input mappings. More specifically, in case of mappings with multiple derivations, a single trusted derivation suffices to infer that the result mapping is trusted. When two mappings are combined in a derivation, both of them should be trusted in order for the result mapping to be trusted. Based on this semantics, which has also been studied in the relational context [15, 16], one can compute the trusted result mappings by answering the query over the subset of the input consisting of trusted triples only. This semantics also coincides with the set semantics of SPARQL [23] in which a trusted mapping belongs to the output mapping set and an untrusted mapping does not.

EXAMPLE 3 (TRUST, SET). Consider Fig. 1 where instead of the **#** column, we now focus on the annotations in the **tr(ust)** column. For example, each triple in Table (b) comes with a boolean trust value (τ_i) that is `true` if the triple is trusted, and is `false` otherwise. It is readily verified that the desired trust semantics is obtained for the example SPARQL queries in Fig. 1 by combining trust values through *disjunction* ($\vee$) and *conjunction* ($\wedge$), instead of addition and multiplication, respectively, used to compute multiplicities in Example 1. For example, μ_{11} in Table (e) is trusted only if one of the mappings μ_4 or μ_5 is trusted. Similarly, μ_{16} in Table (g) is trusted if μ_1 is trusted *and* either μ_4 or μ_5 is trusted. To deal with $\setminus$, we consider boolean negation (denoted by $\bar{\tau}$ for a trust variable τ). Indeed, consider the gray shaded entry μ_{18} in Table (h). This mapping is trusted only if μ_1 is *untrusted* and either μ_4 or μ_5 is trusted. This is expressed by $(\tau_4 \vee \tau_5) -_{trust} \bar{\tau}_1$. Hence, the gray shaded mappings can be part of the result depending on the trust information of the source mappings. In general, if φ and ψ are two propositional formulas over boolean variables, then their difference is defined as $\varphi -_{trust} \psi := \varphi \wedge \bar{\psi}$. Note that this is similar to the notion of difference given in the bag semantics (cf. Example 1): $\varphi \wedge \bar{\psi}$ equals φ in case that ψ is `false`, and equals `false` otherwise. □

We conclude this section by considering SPARQL on fuzzy RDF data [22]. In this setting, every mapping is annotated with a real number in the range $[0, 1]$, where the annotation denotes the probability that the mapping exists in the particular mapping set. Mappings annotated with 0 certainly do not exist in the mapping set, while those annotated with 1 certainly exist. In the case of mappings with alternative derivations, the probability of the mapping is that of the derivation with the highest probability, while the probability of a composite derivation equals the minimum of the probabilities over the two input mappings.

EXAMPLE 4 (FUZZY). Consider Fig. 1 where instead of the **#** column, we now focus on the **fu(zzy)** column. For example, each triple in Table (b) comes with a probability (p_i) that is a real number in $[0, 1]$. It is readily verified that the desired fuzzy semantics is obtained for the example SPARQL queries in Fig. 1 if we take the *maximum* ($\max$) and *minimum* ($\min$) instead of addition and multiplication, respectively, as used for bag semantics (cf. Example 1).

To deal with $\setminus$, we need to consider an additional operator $-_{fuzzy}$ on probabilities that is defined as $p -_{fuzzy} q := 0$ in case that $q \neq 0$, and $p -_{fuzzy} q := p$ otherwise. For example, the gray shaded entry μ_{18} in Table (h) appears with probability $\max(p_4, p_5) -_{fuzzy} p_1$. That is, if μ_1 has non-zero probability, then μ_{18} will not appear in the query result. Similarly, if both p_4 and p_5 have zero probability, then μ_{18} is not part of the output. In all other cases, μ_{18} is a result mapping with probability $\max(p_4, p_5)$. Note again the similarity between $-_{fuzzy}$, $-_{bag}$ and $-_{trust}$. □

Summary and lookahead. The previous examples suggest a commonality between the different semantics of SPARQL on annotated RDF. Similar to the semiring-based approach for annotated relational data we *unify* the semantics of SPARQL for a wide range of annotations as follows: First, we extend mappings to annotated mappings that take values in some abstract set K of annotations; Second, we enrich K with operations for capturing the semantics of the query language operators. More specifically, we enrich K with the following three binary operations:

- A binary operator $\oplus$ for modeling $+$, $\vee$ and $\max$, among others, for the operators π and $\cup$;
- A binary operator $\otimes$ for modeling $\times$, $\wedge$ and $\min$, among others, for the operators $\bowtie$ and $\mathbin{⨝}$; and
- A binary operator $\ominus$ for modeling $-_{bag}$, $-_{trust}$ and $-_{fuzzy}$, among others, for the operators $\setminus$ and $\mathbin{▷}$.

Finally, based on SPARQL query equivalences that are known to hold in the bag, trust (set) and fuzzy setting, among others, we identify a set of additional properties that the algebraic structure consisting of K, $\oplus$, $\otimes$ and $\ominus$ must have, and provide a characterization of these structures. We provide additional examples of annotated RDF, commonly used in practice, in Section 5 and show that these are all unified by our approach.

3. SEMANTICS OF SPARQL ON ANNOTATED RDF

We next formalize the semantics of SPARQL on annotated RDF. First we extend RDF with annotations and then extend the semantics of SPARQL correspondingly.

Let I, B and L be pairwise disjoint infinite sets of Internationalized Resource Identifiers (IRIs), blank nodes, and literals, respectively. A triple $(s,p,o) \in (I \cup B) \times I \times (I \cup B \cup L)$ is called an *RDF triple*. As mentioned before, s is the subject, p the predicate and o the object. An *RDF graph* is a finite set of RDF triples.

DEFINITION 1. Let K be a set of annotations, disjoint from I, B and L. A *K-annotated RDF triple* is of the form $(s,p,o) \mapsto k$ where (s,p,o) is an RDF triple and $k \in K$. A finite set of annotated RDF triples is called a *K-annotated RDF graph* if every triple (s,p,o) has a single annotation in K. □

A SPARQL *graph pattern expression* is defined inductively as follows. Let V be a set of variables, disjoint from I, B and L.

1. A triple from $(I \cup V) \times (I \cup V) \times (I \cup L \cup V)$ is a graph pattern (hereby called *triple pattern*).
2. If P_1 and P_2 are graph patterns, then $(P_1 \texttt{ UNION } P_2)$, $(P_1 \texttt{ AND } P_2)$, $(P_1 \texttt{ DIFFERENCE } P_2)$, and $(P_1 \texttt{ OPTIONAL } P_2)$ are also graph patterns.
3. If P is a graph pattern and R is a SPARQL built-in condition, then $(P \texttt{ FILTER } R)$ is a graph pattern. We refer to [23] for an overview of built-in conditions supported by SPARQL.
4. If P is a graph pattern and $S \subseteq V$, then $\texttt{SELECT}_S(P)$ is a graph pattern.

We next extend the semantics of SPARQL from RDF graphs to K-annotated RDF graphs, hereby closely following the SPARQL semantics given in [23, 3] Let $var(t)$ denote the set of variables from V that appear in a triple pattern t and denote $I \cup B \cup L$ by T. A *mapping* μ from V to T is a partial function $\mu : V \to T$. The domain of μ, denoted by $dom(\mu)$, is the subset of V on which μ is defined. We say that two mappings μ_1 and μ_2 are *compatible* when for all $v \in dom(\mu_1) \cap dom(\mu_2)$, it is the case that $\mu_1(v) = \mu_2(v)$. We denote this by $\mu_1 \sim \mu_2$. It is readily verified that if $\mu_1 \sim \mu_2$ then $\mu_1 \cup \mu_2$ is also a mapping from $V \to T$. We denote by $\mathcal{M}$ the set of all mappings from V to T.

DEFINITION 2. Let K be a set of annotations and let 0 denote a distinguished element from K. A *K-annotated mapping set on $\mathcal{M}$* is a total function $\mathbf{k} : \mathcal{M} \to K$ such that its support $\{\mu \mid \mu \in \mathcal{M}, \mathbf{k}(\mu) \neq 0\}$ is finite. □

Intuitively, we interpret a mapping with its annotation set to 0 as not being part of the mapping set. The finite support requirement thus ensures that only a finite number of mappings are considered.

We already mentioned that the semantics of SPARQL graph patterns on RDF graphs can be defined in terms of algebra operations on mapping sets. We now generalize the semantics of the algebra operators σ, π, $\cup$, $\bowtie$, $\mathbin{⨝}$ and $\setminus$ to K-annotated mapping sets. As indicated at the end of the previous section, we assume the presence of binary operations $\oplus$, $\otimes$ and $\ominus$, and distinguished constant elements 0 and 1 in the set K of annotations. At this point $(K, \oplus, \otimes, \ominus, 0, 1)$ is simply regarded as an algebra on its carrier set K and we do not yet impose any conditions on its operations. These will be specified in the next section. The definitions below are inspired by the bag, trust (set) and fuzzy semantics of SPARQL as illustrated in Section 2. The algebra operations on mapping sets are defined as follows. Let $\mathbf{k}$ be a K-annotated mapping set on $\mathcal{M}$ and let $S \subseteq V$. Then for all $\mu \in \mathcal{M}$ and $\nu \in \mathcal{M}$ such that $dom(\nu) \subseteq S$:

$$(\sigma_R(\mathbf{k}))(\mu) := \mathbf{k}(\mu) \otimes F_R(\mu)$$

$$(\pi_S(\mathbf{k}))(\nu) := \bigoplus_{\substack{\mu \in \mathcal{M}, \exists \mu' \in \mathcal{M} \\ \mu = \nu \cup \mu' \,\&\, dom(\mu') \cap S = \emptyset}} \mathbf{k}(\mu),$$

where for all $\mu \in \mathcal{M}$, $F_R(\mu) = 1$ if μ satisfies the filter and $F_R(\mu) = 0$ otherwise. We refer to [23] for the definition of when a mapping satisfies a filter. Let $\mathbf{k}_1$ and $\mathbf{k}_2$ be two K-annotated mapping sets. Then, for all $\mu \in \mathcal{M}$:

$$(\mathbf{k}_1 \cup \mathbf{k}_2)(\mu) := \mathbf{k}_1(\mu) \oplus \mathbf{k}_2(\mu)$$

$$(\mathbf{k}_1 \bowtie \mathbf{k}_2)(\mu) := \bigoplus_{\substack{\mu_1, \mu_2 \in \mathcal{M} \\ \mu = \mu_1 \cup \mu_2}} \mathbf{k}_1(\mu_1) \otimes \mathbf{k}_2(\mu_2)$$

$$(\mathbf{k}_1 \setminus \mathbf{k}_2)(\mu) := \mathbf{k}_1(\mu) \ominus \Big(\bigoplus_{\mu' \in \mathcal{M}, \mu \sim \mu'} \mathbf{k}_2(\mu') \Big)$$

$$(\mathbf{k}_1 \mathbin{⨝} \mathbf{k}_2)(\mu) := ((\mathbf{k}_1 \bowtie \mathbf{k}_2) \cup (\mathbf{k}_1 \setminus \mathbf{k}_2))(\mu).$$

We refer to the above as the *SPARQL K-annotated algebra*. We are now ready to define the semantics of SPARQL on annotated RDF graphs. The semantics is defined inductively, following the definition of graph patterns described above. That is, we first define the evaluation of a triple pattern t on a K-annotated RDF graph G_a as the K-annotated mapping set

$$\forall \mu \in \mathcal{M}: \quad [\![t]\!]_{G_a}(\mu) := \begin{cases} k & \text{if } \mu(t) \mapsto k \in G_a \\ 0 & \text{otherwise.} \end{cases}$$

Note that this indeed a K-annotated mapping: (i) it defines a function because RDF triples in G_a have a unique annotation; and (ii) it

has finite support since RDF graphs are finite objects. The semantics of the remaining SPARQL graph patterns is defined as follows:

$$[\![P_1 \texttt{ FILTER } R]\!]_{G_a} := \sigma_R\big([\![P_1]\!]_{G_a}\big)$$
$$[\![\texttt{SELECT}_S P_1]\!]_{G_a} = \pi_S\big([\![P_1]\!]_{G_a}\big)$$
$$[\![P_1 \texttt{ UNION } P_2]\!]_{G_a} := [\![P_1]\!]_{G_a} \cup [\![P_2]\!]_{G_a}$$
$$[\![P_1 \texttt{ AND } P_2]\!]_{G_a} := [\![P_1]\!]_{G_a} \bowtie [\![P_2]\!]_{G_a}$$
$$[\![P_1 \texttt{ DIFFERENCE } P_2]\!]_{G_a} := [\![P_1]\!]_{G_a} \setminus [\![P_2]\!]_{G_a}$$
$$[\![P_1 \texttt{ OPTIONAL } P_2]\!]_{G_a} := [\![P_1]\!]_{G_a} \mathbin{⟕} [\![P_2]\!]_{G_a}$$

So far, we have not imposed any conditions on the set K and its operations. We need to impose conditions, however, to ensure that the semantics is well-defined. For instance, suppose that $\oplus$ is chosen such that $0 \oplus 0 = 1$. Then, the union of two empty RDF graphs results in a total function on $\mathcal{M}$ whose support is infinite and thus the above semantics is not well-defined. Indeed, recall that a K-annotated mapping set has a finite support but the choice of $\oplus$ causes infinitely many result triples. We identify additional conditions on the operations in the next section. It can already be verified, however, that the above semantics does make sense for the following choices of K: For $(\mathbb{N}, +, \times, -_{bag}, 0, 1)$, $(\{\mathsf{true}, \mathsf{false}\}, \vee, \wedge, -_{trust}, \mathsf{false}, \mathsf{true})$ and $([0,1], \max, \min, -_{fuzzy}, 0, 1)$, the above semantics coincides with the bag, trust (set) and fuzzy semantics of SPARQL, respectively (cf. Section 2).

4. SPARQL ANNOTATION STRUCTURE

We next identify a set of SPARQL equivalences that are expected to hold in the general K-annotated setting and show that these equivalences enforce a specific structure on the underlying set K of annotations.

SPARQL equivalences and identities on $(K, \oplus, \otimes, \ominus, 0, 1)$. Similar to the relational case, SPARQL optimization techniques rely on SPARQL algebra *equivalences*. For the bag semantics of SPARQL, an extensive list of such equivalences is identified by Schmidt et al. [25]. For example, the equivalence $P_1 \cup P_2 \equiv P_2 \cup P_1$ holds for all SPARQL algebra expressions P_1 and P_2. That is, for any RDF graph G, $[\![P_1]\!]_G \cup [\![P_2]\!]_G = [\![P_2]\!]_G \cup [\![P_1]\!]_G$. On the other hand, *identities* are commonly used to express conditions on algebraic structures such as $(K, \oplus, \otimes, \ominus, 0, 1)$. For example, the identity $k_1 \oplus k_2 = k_2 \oplus k_1$ expresses that for all values k_1, k_2 in K, $\oplus$ is commutative. In this case, K is said to satisfy the identity. Algebraic structures that satisfy a set of identities are commonly known as equational varieties [7]. Equivalences and identities are always assumed to be universally quantified; we omit this quantification.

We next establish a connection between SPARQL equivalences on K-annotated RDF graphs and identities on the set K of annotations. Denote by m_i a variable that is interpreted as a K-annotated mapping set and let $P_i(m_1, \ldots, m_n)$, for $i = 1, 2$, be two SPARQL algebra expressions over these variables. Then $P_1 \equiv P_2$ indicates that for all K-annotated mapping sets $\mathbf{k}_1, \ldots, \mathbf{k}_n$,

$$P_1(\mathbf{k}_1, \ldots, \mathbf{k}_n) = P_2(\mathbf{k}_1, \ldots, \mathbf{k}_n).$$

We turn such an equivalence into an identity on $(K, \oplus, \otimes, \ominus, 0, 1)$ as follows. We make the following assumptions on expressions P in an equivalence: If P contains a selection then it must be of the form $\sigma_{?x=?x}$, indicating that all triples are selected; P is not allowed to contain projections, as these do not have a direct counterpart in $(K, \oplus, \otimes, \ominus, 0, 1)$; and $\emptyset$ can be used in P, where $\emptyset$ denotes the empty mapping set. Although it is possible to consider arbitrary expressions P, the restricted set of expressions defined above suffices for our purposes. Given such an expression P, we define $\mathsf{ann}(P)$ inductively as follows: (i) if $P = m_i$, then $\mathsf{ann}(m_i) = k_i$ where k_i is a variable to be interpreted as a value in K; (ii) if $P = Q \cup R$, then $\mathsf{ann}(P) = \mathsf{ann}(Q) \oplus \mathsf{ann}(R)$; if $P = Q \bowtie R$, then $\mathsf{ann}(P) = \mathsf{ann}(Q) \otimes \mathsf{ann}(R)$; (iii) if $P = Q \setminus R$, then $\mathsf{ann}(P) = \mathsf{ann}(Q) \ominus \mathsf{ann}(R)$; (iv) if $P = Q ⟕ R$, then $\mathsf{ann}(P) = (\mathsf{ann}(Q) \otimes \mathsf{ann}(R)) \oplus (\mathsf{ann}(Q) \ominus \mathsf{ann}(R))$; (v) if $P = \sigma_{?x=?x}(Q)$ then $\mathsf{ann}(P) = \mathsf{ann}(Q) \otimes 1$; and finally, (vi) if $P = \emptyset$, then $\mathsf{ann}(P) = 0$. Similarly, given an expression e using variables $k_1, \ldots, k_n$, operations $\oplus$, $\otimes$, $\ominus$, and constants 0 and 1, we can define a SPARQL expression $\mathsf{exp}(e)$ along the same lines.

The following lemma is readily verified.

LEMMA 1. *If $P_1 \equiv P_2$ is a SPARQL K-annotated algebra equivalence, then $\mathsf{ann}(P_1) = \mathsf{ann}(P_2)$ is an identity that holds on $(K, \oplus, \otimes, \ominus, 0, 1)$. Conversely, if $e_1 = e_2$ is an identity that holds on $(K, \oplus, \otimes, \ominus, 0, 1)$, then $\mathsf{exp}(e_1) \equiv \mathsf{exp}(e_2)$ is a SPARQL K-annotated algebra equivalence.* □

In other words, one can blur the distinction between SPARQL equivalences on K-annotated RDF graphs and identities on the underlying annotation structure $(K, \oplus, \otimes, \ominus, 0, 1)$. The lemma also implies that the choice of annotation structure *entirely* depends on the SPARQL equivalences that one would like to be satisfied when evaluating SPARQL on K-annotated RDF graphs. We next propose a set of equivalences that are desired to hold when evaluating SPARQL on K-annotated RDF graphs.

SPARQL equivalences for the positive fragment. Consider first the positive fragment of the SPARQL algebra, i.e., the fragment that does not include $\setminus$ and $⟕$. It has been noted [25] that the following SPARQL algebra equivalences hold in the case of set and bag semantics, and thus are natural to impose in the general K-annotated setting: For any expression P_1, P_2 and P_3, we have that

$$\mathcal{E}_1 = \left\{ \begin{array}{lll} \sigma_{?x=?x}(P_1) \equiv P_1 & P_1 \bowtie \emptyset \equiv \emptyset & P_1 \cup \emptyset \equiv P_1 \\ P_1 \cup P_2 \equiv P_2 \cup P_1 & P_1 \bowtie P_2 \equiv P_2 \bowtie P_1 & \\ P_1 \cup (P_2 \cup P_3) \equiv (P_1 \cup P_2) \cup P_3 & & \\ P_1 \bowtie (P_2 \bowtie P_3) \equiv (P_1 \bowtie P_2) \bowtie P_3 & & \\ P_1 \bowtie (P_2 \cup P_3) \equiv (P_1 \bowtie P_2) \cup (P_1 \bowtie P_3) & & \end{array} \right\}.$$

The following Proposition is a direct consequence of Lemma 1. Indeed, the SPARQL equivalences in $\mathcal{E}_1$ precisely correspond to the set of semiring identities, shown as id_1–id_8 in Fig. 2.

PROPOSITION 1. *The positive fragment of the SPARQL K-annotated algebra satisfies the equivalences in $\mathcal{E}_1$ if and only if $(K, \oplus, \otimes, 0, 1)$ is a semiring. Furthermore, for any positive SPARQL algebra expression P and K-annotated RDF graph G_a, $[\![P]\!]_{G_a}$ has a finite support.*

PROOF. The correspondence between equivalences in $\mathcal{E}_1$ and the semiring identities id_1–id_8 given in Fig. 2 follows from Lemma 1. One can show that these identities ensure finite support of K-annotated mapping sets by induction on the structure of SPARQL expressions. □

id_1:	$k \otimes 1 = k$	id_2:	$k \otimes 0 = 0$	id_3:	$k \oplus 0 = k$
id_4:	$k_1 \oplus k_2 = k_2 \oplus k_1$			id_5:	$k_1 \otimes k_2 = k_2 \otimes k_1$
id_6:	$k_1 \oplus (k_2 \oplus k_3) = (k_1 \oplus k_2) \oplus k_3$			id_9:	$k \ominus k = 0$
id_7:	$k_1 \otimes (k_2 \otimes k_3) = (k_1 \otimes k_2) \otimes k_3$				
id_8:	$k_1 \otimes (k_2 \oplus k_3) = (k_1 \otimes k_2) \oplus (k_1 \otimes k_3)$				
id_{10}:	$k_1 \ominus (k_2 \oplus k_3) = (k_1 \ominus k_2) \ominus k_3$				
id_{11}:	$k_1 \otimes (k_2 \ominus k_3) = (k_1 \otimes k_2) \ominus k_3$				
id_{12}:	$(k_1 \ominus (k_1 \ominus k_2)) \oplus (k_1 \ominus k_2) = k_1$				

Figure 2: Axiomatization $\mathcal{A}$ of spm-semirings.

SPARQL equivalences involving difference. We next turn our attention to the difference operator. The examples given in

Section 2 suggest that the operation $\ominus$ should satisfy $k_1 \ominus k_2 = k_1$ in case that $k_2 = 0$ and $k_1 \ominus k_2 = 0$ otherwise.

We first observe that the non-monotone operation $\ominus$ cannot be derived from the operations $\oplus$ and $\otimes$ in a semiring. Indeed, this can be shown in a similar way as the proof that aggregate functions (which are also non-monotone) cannot be captured using the semiring operations only [2]. It is thus necessary to extend semirings with an additional operator $\ominus$. However, the result below tells us that there is no set of identities to define $\ominus$ so that its semantics coincides as outlined above.

PROPOSITION 2. *There exists no equational variety* $(K, \oplus, \otimes, \ominus, 0, 1)$ *such that for any* $k_1,\ k_2 \in K,\ k_1 \ominus k_2 = k_1$ *in case that* $k_2 = 0$*, and* $k_1 \ominus k_2 = 0$ *otherwise.*

PROOF. This is an immediate consequence of Birkhoff's Theorem which implies that any equational variety should be closed under taking homomorphisms, subalgebras and products (see e.g., [7]). Suppose, for the sake of contradiction, that identities exist that define the desired $\ominus$ and denote by $\mathcal{V}$ the corresponding variety. Let $(K, \oplus_K, \otimes_K, \ominus_K, 0_K, 1_K)$ and $(L, \oplus_L, \otimes_L, \ominus_L, 0_L, 1_L)$ be two algebraic structures in $\mathcal{V}$ for which, in addition, $0_K \neq 1_K$ and $0_L \neq 1_L$ hold. Consider the canonical $\ominus_{K,L}$ on the product $K \times L$ defined as $(k_1, \ell_1) \ominus_{K,L} (k_2, \ell_2) = (k_1 \ominus_K k_2, \ell_1 \ominus_L \ell_2)$. The operations $\oplus_{K,L}$, $\otimes_{K,L}$ and constants $0_{K,L}$ and $1_{K,L}$ are similarly defined. Then, by assumption, $(K \times L, \oplus_{K,L}, \otimes_{K,L}, \ominus_{K,L}, 0_{K,L}, 1_{K,L})$ is a structure in $\mathcal{V}$ and thus $(1_K, 1_L) \ominus_{K,L} (0_K, 1_L)$ should be equal to $0_{K,L}$. However, observe that $(1_K, 1_L) \ominus_{K,L} (0_K, 1_L) = (1_K, 0_L) \neq (0_K, 0_L) = 0_{K,L}$, by the definition of $\ominus_{K,L}$. This shows that $\mathcal{V}$ is not closed under taking taking products, a contradiction. Hence, no set of identities exists that defines the desired $\ominus$. □

Not all is lost, however. Schmidt et al. [25] identified a number of equivalences of SPARQL expressions that involve $\setminus$ and that hold under the set and bag semantics. More specifically, the following equivalences were shown to hold for any SPARQL expressions P_1, P_2 and P_3:

$$\mathcal{E}_2 = \left\{ \begin{array}{ll} P_1 \setminus P_1 \equiv \emptyset & P_1 \setminus (P_2 \cup P_3) \equiv (P_1 \setminus P_2) \setminus P_3 \\ & P_1 \bowtie (P_2 \setminus P_3) \equiv (P_1 \bowtie P_2) \setminus P_3 \end{array} \right\}.$$

These equivalences are readily shown to hold in the trust and fuzzy setting as well. We further identify the following equivalence: for any SPARQL expressions P_1 and P_2, we have that

$$\mathcal{E}_3 = \{ \quad (P_1 \setminus (P_1 \setminus P_2)) \cup (P_1 \setminus P_2) \equiv P_1 \quad \}.$$

This equivalence also holds in all settings we have considered so far. By imposing $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$ on the SPARQL K-annotated algebra, we obtain a generalization of Proposition 1 that accommodates for the difference ($\setminus$) and optional operator ($\bowtie$). The proof again relies on Lemma 1.

PROPOSITION 3. *The SPARQL* K*-annotated algebra satisfies the equivalences in* $\mathcal{E}_1$*,* $\mathcal{E}_2$ *and* $\mathcal{E}_3$ *if and only if* $(K, \oplus, \otimes, \ominus, 0, 1)$ *is an algebraic structure satisfying the identities shown in Figure 2. Furthermore, for any SPARQL expression* P *and* K*-annotated RDF graph* G_a*,* $[\![P]\!]_{G_a}$ *has a finite support.* □

DEFINITION 3. An *spm-semiring* is an algebraic structure $(K, \oplus, \otimes, \ominus, 0, 1)$ that satisfies the identities id_1–id_{12} given in Figure 2. Here, "spm" stands for "SPARQL minus". □

Proposition 3 thus implies that spm-semirings are a good candidate annotation structure when considering SPARQL on annotated RDF. It can be readily verified that $(\mathbb{N}, +, \times, -_{bag}, 0, 1)$, $(\{\mathsf{true}, \mathsf{false}\}, \vee, \wedge, -_{trust}, \mathsf{false}, \mathsf{true})$ and $([0,1], \max, \min, -_{fuzzy}, 0, 1)$ are spm-semirings. We will see more examples of spm-semirings in the next section.

Minimality. We next address the minimality of the set of identities that define spm-semirings. Let $\mathcal{I}$ be a set of identities and K be an algebraic structure. We say that K *satisfies* $\mathcal{I}$, denoted by $K \models \mathcal{I}$, if K satisfies all identities in $\mathcal{I}$. Let e be an identity. Then e is *implied* by $\mathcal{I}$, denoted by $\mathcal{I} \models e$, if for all structures K such that $K \models \mathcal{I}$ it holds that $K \models e$. A set $\mathcal{I}$ of identities is said to be *minimal* if for all $e \in \mathcal{I}$, $(\mathcal{I} \setminus e) \not\models e$. Let $\mathcal{A}$ be the set of identities shown in Figure 2.

PROPOSITION 4. *The set of identities* $\mathcal{A} \setminus \{\mathsf{id}_2, \mathsf{id}_3\}$ *is minimal.*

PROOF. Let $\mathcal{A}' = \mathcal{A} \setminus \{\mathsf{id}_2, \mathsf{id}_3\}$. It suffices to show the following: (i) for any identity $e \in \mathcal{A}'$, there exists a structure K_e such that $K_e \models (\mathcal{A}' \setminus e)$ but $K_e \not\models e$; and (ii) for any K such that $K \models \mathcal{A}'$, we have that $K \models \mathsf{id}_2$ and $K \models \mathsf{id}_3$. □

5. CHARACTERIZATION OF SPM-SEMIRINGS

We next address the question of how to construct spm-semirings. More specifically, we characterize the class of spm-semirings in terms of a combination of semirings and boolean algebras, two standard algebraic structures. Not only does this characterization allow for showing that certain structures are spm-semirings, it also opens the way for identifying a universal object in the class of spm-semirings, as will be shown in the next section.

As observed in Section 2, SPARQL adheres to relational bag semantics for most of its operations, except for difference, for which a different semantics is used. More specifically, when taking a difference $P_1 \setminus P_2$, the bag semantics is used for P_1 whereas the multiplicities in P_2 are ignored and only the existence of compatible mappings in P_2 matters. In the more general K-annotated RDF setting, this can be captured through the use of semirings for most operations, and a combination of semirings and boolean algebras for difference. For this purpose we next define a new algebraic structure, called *seba*, that consists of a semiring K together with an embedded boolean algebra B.

DEFINITION 4. A *seba-structure* is of the form $(K, B, d, \imath)$ where K is a commutative semiring $(K, \oplus, \otimes, 0, 1)$, B a boolean algebra $(B, \vee, \wedge, \bar{}, \bot, \top)$, $d : K \to B$ and $\imath : B \to K$ are mappings such that

$$\begin{array}{rr} d(0) = \bot \text{ and } d(1) = \top & \imath(\bot) = 0 \text{ and } \imath(\top) = 1; \\ d(k_1 \oplus k_2) = d(k_1) \vee d(k_2) & \imath(b_1 \vee b_2) = \imath(b_1) \oplus \big(\imath(\bar{b_1}) \otimes \imath(b_2)\big); \\ d(k \otimes \imath(b)) = d(k) \wedge b & \imath(b_1 \wedge b_2) = \imath(b_1) \otimes \imath(b_2), \end{array}$$

and finally, $d \circ \imath = \mathsf{Id} : B \to B$ and $k \otimes \imath(\overline{d(k)}) = 0$. □

Intuitively, a seba-structure consists of a semiring K in which a boolean algebra B is embedded (by means of the mapping $\imath$), and in addition, in which every element in K maps to an element in B (by means of the mapping d). The mapping $d : K \to B$ is to reflect that $d(k) = \bot$ in case that $k = 0$ and $d(k) \neq \bot$ otherwise.

DEFINITION 5. We say that an algebraic structure $(L, \oplus, \otimes, \ominus, 0, 1)$ is *derived* from a seba-structure $(K, B, d, \imath)$ if $(L, \oplus, \otimes, 0, 1)$ and $(K, \oplus, \otimes, 0, 1)$ coincide and for any $k, \ell \in L$, $k \ominus \ell = k \otimes \imath(\overline{d(\ell)})$. □

The main result of this section is that spm-semirings and seba-structures are closely related.

THEOREM 1. *Every spm-semiring is derived from some seba-structure, and vice versa, every structure derived from a seba-structure is an spm-semiring.*

PROOF. We first show that every spm-semiring $(K,\oplus,\otimes,\ominus,0,1)$ is derived from some seba-structure. More specifically, such a seba-structure can be constructed as follows. Define (i) the semiring $(K,\oplus,\otimes,0,1)$; (ii) the boolean algebra $(B,\vee,\wedge,\bot,\top)$, where $B:=\{b\in K\mid \exists \ell\in K, b=1\ominus(1\ominus\ell)\}$ and $b_1\vee b_2:=1\ominus\big(1\ominus(k_1\oplus k_2)\big)$, $\overline{b}:=1\ominus k$, and $b_1\wedge b_2:=\overline{\overline{b_1}\vee\overline{b_2}}$. Here, k, k_1 and k_2 are such that $b=1\ominus(1\ominus k)$, $b_1=1\ominus(1\ominus k_1)$ and $b_2=1\ominus(1\ominus k_2)$; (iii) the mapping $d:K\to B$ such that for any $k\in K$, $d(k)=1\ominus(1\ominus k)$; and (iv) the identity mapping $\imath:B\to K$. Leveraging the fact that $(K,\oplus,\otimes,\ominus,0,1)$ is an spm-semiring and thus satisfies the identities given in Fig. 2, we can verify that the operations $\vee$, $\wedge$ and $\overline{\ }$ are well-defined and that together with B they constitute a boolean algebra. Furthermore, the conditions on d and $\imath$ in the definition of seba-structure are readily shown to hold. Finally, we show that for any $k,\ell\in K$, $k\ominus\ell=k\otimes\imath(\overline{d(\ell)})$. Conversely, we show that derived structures satisfy the identities of spm-semirings. □

We next illustrate sebas and their derived spm-semirings in the following examples. Table 1 summarizes these spm-semirings and their application domains.

EXAMPLE 5 (TRUST, SET). Consider the relational trust semiring $T=(\{\mathsf{true},\mathsf{false}\},\vee,\wedge,\mathsf{false},\mathsf{true})$ [16]. The semiring T is readily extended to a boolean algebra T_b by incorporating complementation $\overline{\ }$. Consider T_b together with the mappings $d:T\to T_b$ and $\imath:T_b\to T$, both of which are the identity mappings. It is readily verified that $(T,T_b,d,\imath)$ is a seba-structure. The derived spm-semiring consists of T together with the additional operator $b_1\ominus b_2:=b_1\wedge\overline{b_2}$. Note that $\ominus$ coincides with $-_{trust}$ (cf. Example 3). □

EXAMPLE 6 (BOOLEAN ALGEBRAS). The previous example generalizes to any semiring K that can be extended to a boolean algebra K_b. As in the previous example, $(K,K_b,d,\imath)$ with d and $\imath$ the identity mapping, forms a seba-structure. The derived spm-semiring is then given by K extended with $k_1\ominus k_2=k_1\wedge\overline{k_2}$. In other words, $\ominus$ coincides with the standard notion of difference in boolean algebras. For example, consider the relational probability semiring [16] $(\mathcal{P}(E),\cup,\cap,\emptyset,E)$ for a set E of events. Clearly, this semiring can be equipped with complementation, and is thus part of a boolean algebra. As a consequence, the minus in the derived spm-semiring is given by $E_1\cap E_2^c$, where, E^c denotes the complement of E. □

The previous example shows that $\ominus$ in spm-semirings does not always satisfy that $k_1\ominus k_2=k_1$ in case that $k_2=0$ and $k_1\ominus k_2=0$ otherwise. Indeed, consider the sets $A=\{a,b\}$ and $B=\{c,d\}$. Then $A\cap B^c=A$ despite the fact that $B\neq\emptyset$. This is not unexpected, however, in view of Proposition 2. We will see below, however, that the probability semiring can be extended to an spm-semiring in another way such that $\ominus$ is as intended.

EXAMPLE 7 (BAGS, FUZZY). Let $B_2=(\{0,1\},\vee,\wedge,\overline{\ },0,1)$ be the two-element boolean algebra and let $(\mathbb{N},+,\times,0,1)$ be the semiring of natural numbers. Define $d:\mathbb{N}\to\{0,1\}$ as $d(x)=0$ if $x=0$ and $d(x)=1$ if $x\neq 0$. Let $\imath:\{0,1\}\to\mathbb{N}$ be the identity mapping. Then clearly, $d(0)=0$, $d(1)=1$, $\imath(0)=0$ and $\imath(1)=1$. Furthermore, it is readily verified that for any $x,y\in\mathbb{N}$ and $b,b_1,b_2\in\{0,1\}$, $d(x+y)=d(x)\vee d(y)$, $d(x\times\imath(b))=d(x)\wedge b$, $\imath(b_1\vee b_2)=\imath(b_1)+\imath(\overline{b_1})\times\imath(b_2)=b_1+(\overline{b_1}\times b_2)$, and $\imath(b_1\wedge b_2)=\imath(b_1)\times\imath(b_2)=b_1\times b_2$. We also have that $x\times\imath(\overline{d(x)})=0$ and that $d\circ\imath$ is the identity mapping. Hence, $(\mathbb{N},B_2,d,\imath)$ is a seba-structure. We can thus extend $\mathbb{N}$ with $\ominus$ derived from $(\mathbb{N},B_2,d,\imath)$ by letting $x\ominus y=x\times\overline{d(y)}$, for any $x,y\in\mathbb{N}$. That is, $x\ominus y=0$ if $y\neq 0$ and $x\ominus y=x$ otherwise. Note that $\ominus$ coincides $-_{bag}$ when SPARQL is evaluated on RDF under—the default—bag semantics (cf. Example 1). Along the same lines, the fuzzy semiring $F=([0,1],\max,\min,0,1)$ can be extended with $\ominus$ derived from $(F,B_2,d,\imath)$ with d and $\imath$ as above. Note that, in this case, $p\ominus q=\min\{p,\overline{d(q)}\}$ is equal to 0 if $q\neq 0$, and is p otherwise, i.e., $\ominus$ coincides with $-_{fuzzy}$, as desired (cf. Example 4). □

Spm-semiring	Application
$(\{\mathsf{true},\mathsf{false}\},\vee,\wedge,-_{trust},\mathsf{false},\mathsf{true})$	Trust/Set Semantics
$(N^\infty,\min,+,-_{rtrust},\infty,0)$	Ranked Trust
$(\mathcal{C},\min,\max,-_{acc},0,P)$	Access control
$(\mathcal{P}(E),\cup,\cap,-_{prob},\emptyset,E)$	Event Tables
$(\mathbb{N},+,\times,-_{bag},0,1)$	Bags
$([0,1],\max,\min,-_{fuzzy},0,1)$	Fuzzy

Table 1: Spm-semirings and their applications.

EXAMPLE 8 (ZERO-SUM FREE SEMIRINGS). The previous example can be generalized to arbitrary zero-sum free semirings. Recall that a semiring K is zero-sum free if for any $k,\ell\in K$ we have that $k\oplus\ell=0$ implies that both k and ℓ are 0. It is readily verified that for such K, $(K,B_2,d,\imath)$ is a seba-structure, where $d(k)=0$ if $k=0$ and $d(k)=1$ if $k\neq 0$, and $\imath(0)=0_K$ and $\imath(1)=1_K$. The minus in the derived structure is consequently defined as $k\ominus\ell=k\otimes\imath(\overline{d(\ell)})$. □

The restriction to zero-sum free semirings in the previous example is necessary. Indeed, let $(K,B,d,\imath)$ be a seba-structure and assume that $k\oplus\ell=0$. Then also $d(k\oplus\ell)=\bot=d(k)\vee d(\ell)$ and hence thus both $d(k)=\bot$ and $d(\ell)=\bot$. We claim that $d(k)=\bot$ iff $k=0$. Suppose, for the sake of contradiction, that $d(k)=\bot$ for $k\neq 0$. Then, from $0\neq k=k\otimes\imath(\overline{d(k)})=0$ we obtain a contradiction. Similarly, for $d(\ell)=\bot$.

EXAMPLE 9 (PROBABILITY). Let us reconsider the probability semiring [16] $(\mathcal{P}(E),\cup,\cap,\emptyset,E)$. Clearly this semiring is zero-sum free and the previous example tells us that a seba-structure can be obtained as follows: for any event $E_i\subseteq E$, $d(E_i)=1$ if $E_i\neq\emptyset$ and $d(E_i)=0$ otherwise, and $\imath(0)=\emptyset$ and $\imath(E)=1$. It is readily verified that the minus in the derived spm-semiring satisfies $E_i\ominus E_j=E_i$ when $E_j=\emptyset$ and $E_i\ominus E_j=\emptyset$ otherwise. We also denote $\ominus$ by $-_{prob}$. □

EXAMPLE 10 (RANKED TRUST, ACCESS POLICY). Consider the tropical semiring $(\mathbb{N}^\infty,\min,+,\infty,0)$ [16], where $\mathbb{N}^\infty$ is short for $\mathbb{N}\cup\{\infty\}$. This semiring has been used to model ranked trust [20] for relational queries, and can be regarded as a generalization of the boolean trust semiring. In the case of RDF data, triples that are completely untrusted, and should be disregarded, have ∞ as their rank, whereas completely trusted triples have rank 0. Clearly, this is a zero-sum free semiring and, similarly to the previous examples, one can extend it to an spm-semiring such that $m-_{rtrust}n=\infty$ in case that $n\neq\infty$ and $m-_{rtrust}n=m$ otherwise. Along the same lines, one can extend the semiring $(\mathcal{C},\min,\max,0,P)$ [11], with $\mathcal{C}=\{P(\text{ublic}),C(\text{onfidential}),S(\text{ecret}),T(\text{op Secret}),0\}$, used in the context of confidentiality policies, to an spm-semiring. Here, the order between the levels of access is specified as $P<C<S<T<0$ and the operators $\min$ and $\max$ are defined relative to this order. It is readily verified that in the resulting spm-semiring, $v-_{acc}w=v$ if w is 0 and can thus be ignored, and $v-_{acc}w=0$ otherwise. □

6. UNIVERSAL SPM-SEMIRING

The importance of the universal "most general" object in a class of algebraic structures has already been emphasized in the relational context [16]. Indeed, in that setting, semirings are the appropriate annotation structure and the semiring of polynomials $(\mathbb{N}[X],+,\times,0,1)$ is known to be universal. It has been shown that the evaluation of positive relational algebra expressions on semiring annotated relations *factors through* the evaluation on relations that take their annotations from $(\mathbb{N}[X],+,\times,0,1)$ [16]. This implies, among other things, that it suffices to extend relational algebra evaluation algorithms to relations that are annotated with values in the universal object, from which the query results on specific annotation structures can then be easily obtained. In this section, we first identify a universal object in the class of seba-structures for which we then show that the derived structure is a universal spm-semiring.

Universal seba-structure. We first define the notion of universal seba-structure. Let $(K_1,B_1,d_1,\imath_1)$ and $(K_2,B_2,d_2,\imath_2)$ be two seba-structures. We say that a mapping (h_s,h_b) between the two seba-structures is a *seba-homomorphism* if the following conditions are satisfied:

- h_s is a semiring homomorphism from K_1 to K_2;
- $h_b = d_2 \circ h_s \circ \imath_1$ is a boolean algebra morphism from B_1 to B_2;
- $d_2 \circ h_s = h_b \circ d_1$ and $h_s \circ \imath_1 = \imath_2 \circ h_b$.

A seba-structure $(K,B,d,\imath)$ is *universal* in the class of all seba-structures, relative to a set of generators $X=\{x_1,\ldots,x_n\}$, if for any seba-structure $(K',L',d',\imath')$ and any valuation $\nu: X \to K'$, we can uniquely extend ν to a seba-homomorphism (h_s,h_b) from $(K,B,d,\imath)$ to $(K',B',d',\imath')$ such that h_s coincides with ν on X.

In the following, we construct a universal seba-structure by extending polynomials with boolean variables. More specifically, we first define a semiring of extended polynomials in which two polynomials are considered equivalent based on the semantics of the boolean variables. Second, we show that this semiring has a boolean algebra embedded in it. The elements of this algebra are polynomials consisting of boolean variables only. Finally, we define the mappings d and $\imath$ between the semiring and boolean algebra and show that, all combined, these form a seba-structure.

① **Semiring:** Let $X=\{x_1,\ldots,x_n\}$ and $B=\{b_1,\ldots,b_n,\bar{b}_1,\ldots,\bar{b}_n\}$ be two disjoint sets of variables. Here, the elements in B are to be interpreted as booleans such that b_i is true (resp. $\bar{b}_i$ is false) if x_i is different from zero and b_i is false (resp. $\bar{b}_i$ is true) if x_i is zero. An *extended polynomial* is an element of $\mathbb{N}[B,X]$ in which the variables in B only appear with exponent 0 or 1. More formally, let $\kappa \in \mathbb{N}^n$ be an integer-valued multi-index and $\beta \in \{0,1\}^{2n}$ be a binary multi-index. For $i \in [1,n]$ and $j \in [1,2n]$, we denote by $\kappa(i)$ and $\beta(j)$ the ith and jth entry in κ and β, respectively. Given (β,κ) and a natural number $a \in \mathbb{N}$, the corresponding *extended monomial* is the expression

$$a \cdot b_1^{\beta(1)} \cdots b_n^{\beta(n)} \cdot (\bar{b}_1)^{\beta(n+1)} \cdots (\bar{b}_n)^{\beta(2n)} \cdot x_1^{\kappa(1)} \cdots x_n^{\kappa(n)},$$

which is succinctly denoted by $a \cdot \mathbf{b}^{\beta} \cdot \mathbf{x}^{\kappa}$. Consequently, an *extended polynomial with integer coefficients* is an expression of the form

$$\sum_{(\beta,\kappa)\in I} a_{\beta,\kappa} \cdot \mathbf{b}^{\beta} \cdot \mathbf{x}^{\kappa},$$

where $a_{\beta,\kappa} \in \mathbb{N}$ and I is an index set consisting of pairs of binary and integer-valued multi-indices. The set of all extended polynomials over X and B is denoted by $\mathbb{N}_e[B,X]$. That is, $\mathbb{N}_e[B,X]$ is the subset of $\mathbb{N}[B,X]$ in which the exponents of boolean variables are restricted to 0 or 1.

EXAMPLE 11. Let $X=\{x_1,x_2,x_3\}$, $B=\{b_1,b_2,b_3,\bar{b}_1,\bar{b}_2,\bar{b}_3\}$. Consider the indices (β_i,κ_i), for $i=1,2,3,4$, with $\beta_1=(1,0,0,0,1,0)$, $\beta_2=(0,0,0,0,1,0)$, $\beta_3=(1,0,0,0,0,0)$, $\beta_4=(1,0,0,1,0,0)$, and $\kappa_1=(3,0,0)$, $\kappa_2=(1,2,0)$, $\kappa_3=(0,0,4)$ and $\kappa_4=(3,0,0)$. Then given the coefficients $a_{\beta_1,\kappa_1}=1$, $a_{\beta_2,\kappa_2}=2$, $a_{\beta_3,\kappa_3}=4$ and $a_{\beta_4,\kappa_4}=6$, the corresponding extended polynomial is given by $b_1\bar{b}_2x_1^3+2\bar{b}_2x_1x_2^2+4b_1x_3^4+6b_1\bar{b}_1x_1^3$. □

To ensure that the variables in B in the extended polynomials are treated as boolean, we define an equivalence relation on $\mathbb{N}_e[B,X]$ that indicates when two polynomials are equivalent. More specifically, let $\asymp$ be the smallest equivalence relation on extended polynomials such that for any two polynomials $p[B,X]$ and $q[B,X]$ in $\mathbb{N}_e[B,X]$ we impose the following conditions.

- *To impose that b_i and $\bar{b}_i$ cannot be true at the same time:* $p[B,X] \asymp q[B,X]$ if $p[B,X]$ can be obtained from $q[B,X]$ by removing a monomial in $q[B,X]$ that contains both b_i and $\bar{b}_i$. In other words, $b_i \cdot \bar{b}_i = 0$;
- *To impose that the presence of $\bar{b}_i$ implies that x_i is zero:* $p[B,X] \asymp q[B,X]$ if $p[B,X]$ can be obtained from $q[B,X]$ by removing a monomial in $q[B,X]$ that contains both $\bar{b}_i$ and x_i^n for $n>0$. In other words, $\bar{b}_i \cdot x_i^n = 0$ for $n>0$;
- *To impose that the presence of x_i implies that b_i is true:* $p[B,X] \asymp q[B,X]$ if $p[B,X]$ can be obtained from $q[B,X]$ by replacing a monomial $m[B,X]$ in $q[B,X]$ that contains x_i^n but does not contain b_i, by the monomial $b_i \cdot m[B,X]$. In other words, $b_i \cdot x_i^n = x_i^n$; and
- *To impose that b_i and $\bar{b}_i$ are complements:* $p[B,X] \asymp q[B,X]$ if $p[B,X]$ can be obtained from $q[B,X]$ by replacing a monomial $m[B,X]$ in $q[B,X]$ that does not contain x_i^m for $m>0$ by $b_i \cdot m[B,X] + \bar{b}_i \cdot m[B,X]$. In other words, $b_i + \bar{b}_i = 1$.

Let ***BIM*** be the set of all binary multi-indices in $\{0,1\}^{2n}$ such that for each $i \in [1,n]$ either $\beta(i)=1$ and $\beta(n+i)=0$, or $\beta(i)=0$ and $\beta(n+i)=1$. In other words, for an index $\beta \in$ ***BIM***, $\mathbf{b}^{\beta}$ contains either b_i or $\bar{b}_i$, but not both. Proposition 5 below implies that any extended polynomial is equivalent to an extended polynomial in normal form. We first define this normal form:

DEFINITION 6. An extended polynomial is said to be in *normal form* if it is of the form

$$\sum_{\beta \in BIM} \mathbf{b}^{\beta} \cdot p_{\beta}[X],$$

where $p_{\beta}[X] \in \mathbb{N}[X]$ such that if x_i^m occurs in $p_{\beta}[X]$ then $\beta(i)=1$ (and thus $\beta(n+i)=0$), and if $\beta(i)=0$ then $\beta(n+i)=1$ and x_i^m does not occur in $p_{\beta}[X]$, for some $m>0$. The *support* of an extended polynomial in normal form is the set of of indices in ***BIM*** such that $p_{\beta}[X] \neq 0$. □

PROPOSITION 5. *Let $p[B,X]$ be a polynomial in $\mathbb{N}_e[B,X]$. Then $p[B,X]$ is equivalent to an extended polynomial in normal form. Furthermore, if $p[B,X] \asymp q[B,X]$ then $p[B,X]$ and $q[B,X]$ are equivalent to the same normal form.* □

EXAMPLE 12. Consider the extended polynomial $b_1\bar{b}_2x_1^3+2\bar{b}_2x_1x_2^2+4b_1x_3^4+6b_1\bar{b}_1x_1^3$ from Example 11. It is readily verified that the equivalent normal form is given by $4b_1b_2b_3x_3^4+0b_1b_2\bar{b}_3+b_1\bar{b}_2b_3(x_1^3+4x_3^4)+0\bar{b}_1b_2b_3+b_1\bar{b}_2\bar{b}_3$ $x_1^3+0\bar{b}_1b_2\bar{b}_3+0\bar{b}_1\bar{b}_2b_3+0\bar{b}_1\bar{b}_2\bar{b}_3$. The support of this polynomial is $\{(1,1,1,0,0,0),(1,0,1,0,1,0),(1,0,0,0,1,1)\}$. □

Consider the quotient structure $\mathbb{N}_e[B,X]/\asymp$ and denote by $p[B,X]/\asymp$ the equivalence class in $\mathbb{N}_e[B,X]/_\asymp$ corresponding to the extended polynomial $p[B,X]$. Define the following operations: $(p[B,X]/\asymp)+_e(q[B,X]/\asymp):=(p[B,X]+q[B,X])/\asymp$ and $(p[B,X]/\asymp)\times_e(q[B,X]/\asymp):=(p[B,X]\cdot q[B,X])/\asymp$, and define the following constants: $0_e=0/\asymp$ and $1_e=1/\asymp$.

LEMMA 2. *The structure* $(\mathbb{N}_e[B,X]/\asymp,+_e,\times_e,0_e,1_e)$ *is a semiring.*

PROOF. It suffices to show that $\asymp$ is a *congruence relation* on $\mathbb{N}_e[B,X]$. Indeed, it is known that for any semiring K and any congruence relation $\asymp$, the quotient structure $K/\asymp$ is a semiring. In $K/\asymp$, the operations and constants are canonically defined as above. The lemma then follows since $(\mathbb{N}_e[B,X],+,\cdot,0,1)$ is a semiring. To show that $\asymp$ is a congruence relation, we use Proposition 5. □

② **Boolean algebra:** We next identify a boolean algebra that resides within $\mathbb{N}_e[B,X]/\asymp$. Let

$$\mathbb{B}=\{f_S=(\sum_{\beta\in S}\mathbf{b}^\beta)/\asymp \mid S\subseteq \mathit{BIM}\}$$

and define for f_S and f_T in $\mathbb{B}$: disjunction $f_S\vee_b f_T:=(\sum_{\beta\in S\cup T}\mathbf{b}^\beta)/\asymp$, conjunction $f_S\wedge_b f_T:=(\sum_{\beta\in S\cap T}\mathbf{b}^\beta)/\asymp$, and $c_b(f_S):=(\sum_{\beta\in \mathit{BIM}\setminus S}\mathbf{b}^\beta)/\asymp$, and define $\bot_b$ and $\top_b$ as $f_\emptyset$ and f_{BIM}, respectively.

LEMMA 3. *The structure* $(\mathbb{B},\vee_b,\wedge_b,c_b,\bot_b,\top_b)$ *is a boolean algebra.*

PROOF. This follows from the fact that $(2^{\mathit{BIM}},\cup,\cap,\setminus,\emptyset,\mathit{BIM})$ is a boolean algebra. □

③ **Mappings** d_e **and** $\imath_e$**:** The mappings $d:\mathbb{N}_e[B,X]/\asymp\to\mathbb{B}$ and $\imath:\mathbb{B}\to\mathbb{N}_e[B,X]/\asymp$ are defined as follows. Let $p[B,X]\in\mathbb{N}_e[B,X]$ and let its equivalent normal form be $p_{nf}[B,X]=\sum_{\beta\in\mathit{BIM}}\mathbf{b}^\beta\cdot p_\beta[X]$. We define $d_e(p[B,X]/\asymp):=f_S$, where S is the support of $p_{nf}[B,X]$. Proposition 5 implies that the mapping d_e is well-defined. Furthermore, we define $\imath_e:\mathbb{B}\to\mathbb{N}_e[B,X]/\asymp$ as the identity function that maps f_S to f_S, for some $S\subseteq\mathit{BIM}$. Observe that given these definitions the variables in X and B are closely related:

$$d_e((x_i)/\asymp)=(\sum_{\beta\in\mathit{BIM}\mid\beta(i)=1}\mathbf{b}^\beta)/\asymp=(b_i)/\asymp.$$

That is, $(b_i)/\asymp=\imath_e(d_e((x_i)/\asymp))$ for $i\in[1,n]$ and similarly, $(\bar{b}_i)/\asymp=\imath_e(d_e(c_b(x_i/\asymp)))$ for $i\in[1,n]$.

EXAMPLE 13. Consider the extended polynomial $p[B,X]=b_1\bar{b}_2x_1^3+2\bar{b}_2x_1x_2^2+4b_1x_3^4+6b_1\bar{b}_1x_1^3$ from Example 11. Then $d_e(p[B,X]/\asymp)=b_1b_2b_3+b_1\bar{b}_2b_3+b_1\bar{b}_2\bar{b}_3$, given the normal form of $p[B,X]$ and its support from Example 12. □

PROPOSITION 6. *The structure consisting of the semiring* $((\mathbb{N}_e[B,X])/\asymp,+_e,\times_e,0_e,1_e)$*, boolean algebra* $(\mathbb{B},\vee_b,\wedge_b,c_b,\bot_b,\top_b)$ *and mappings* d *and* $\imath$*, is a seba-structure.*

PROOF. Lemmas 2 and 3 imply that $\mathbb{N}_e[B,X])/\asymp$ is a semiring and that $\mathbb{B}$ is a boolean algebra, respectively. We show the proposition by verifying that the mappings d_e and $\imath_e$ satisfy the conditions stated in the definition of a seba-structure (Definition 4). □

THEOREM 2. *The seba-structure* $(\mathbb{N}_e[B,X]/\asymp,\mathbb{B},d_e,\imath_e)$ *is universal in the class of all seba-structures, relative to the generator set* X.

PROOF. Let $(K,B,d,\imath)$ be a seba-structure consisting of a semiring $(K,\oplus,\otimes,0,1)$, boolean algebra $(B,\vee,\wedge,\bot,\top)$ and mappings $d:K\to B$ and $\imath:B\to K$. Let $X=\{x_1,\ldots,x_n\}$ be a set of variables. Let $X/\asymp=\{x_1/\asymp,\ldots,x_n/\asymp\}$ be the corresponding elements in $\mathbb{N}_e[B,X]/\asymp$. Furthermore, let $\mathbb{N}/\asymp=\{n/\asymp\mid n\in\mathbb{N}\}$ be the embedding of $\mathbb{N}$ in $\mathbb{N}_e[B,X]/\asymp$. Note that $0/\asymp=0_e$ and $1/\asymp=1_e$. Consider the standard embedding $\mu:\mathbb{N}/\asymp\to K$ defined by $\mu(0_e)=0$, $\mu(1_e)=1$ and $\mu(k/\asymp)=k\otimes\mu(1_e)$. Let $\nu:X/\asymp\to K$ be a valuation. We need to show that there exists a unique seba-morphism (h_s,h_b) from $(\mathbb{N}_e[B,X]/\asymp,\mathbb{B},d_e,\imath_e)$ to $(K,B,d,\imath)$. This is verified by showing that μ and ν completely specify the value of this morphism on every other element in the structure $(\mathbb{N}_e[B,X]/\asymp,\mathbb{B},d_e,\imath_e)$. □

Universal spm-semiring. Having a universal seba-structure at our disposal, we next describe the corresponding universal spm-semiring. The following lemma tells that we can indeed move from seba-structures to spm-semirings with regards to universal structures.

Let $(K,\oplus_K,\otimes_K,\ominus_K,0_K,1_K)$ and $(L,\oplus_L,\otimes_L,\ominus_L,0_L,1_L)$ be two spm-semirings. A *homomorphism* between these structures is a mapping $h:K\to L$ such that $h(0_K)=0_L$ and $h(1_K)=1_L$, $h(x\oplus_K y)=h(x)\oplus_L h(y)$, $h(x\otimes_K y)=h(x)\otimes_L h(y)$ and $h(x\ominus_K y)=h(x)\ominus_L h(y)$.

LEMMA 4. *A seba-homomorphism between two seba-structures induces a homomorphism between their derived spm-semirings, and vice versa.* □

We are now finally ready to define a universal spm-semiring. An spm-semiring $(K,\oplus_K,\otimes_K,\ominus_K,0_K,1_K)$ is *universal* in the class of all spm-semirings, relative to a set of generators $X=\{x_1,\ldots,x_n\}$, if for any spm-semiring $(L,\oplus_L,\otimes_L,\ominus_L,0_L,1_L)$ and any valuation $\nu:X\to L$ that assigns a value from L to variables in X, we can *uniquely* extend ν to an spm-semiring homomorphism $h:K\to L$ such that h coincides with ν on X.

DEFINITION 7. Consider the spm-semiring $(\mathbb{N}_e[B,X]/\asymp,+_e,\times_e,-_e,0_e,1_e)$ where

$$(p[B,X]/\asymp)-_e(q[B,X]/\asymp):=(p[B,X]/\asymp\times_e f_S,$$

where S is the *complement* of the support of the normal form of $q[B,X]$. In other words, this is the spm-semiring derived from the universal seba-structure $(\mathbb{N}_e[B,X]/\asymp,\mathbb{B},d_e,\imath_e)$. □

PROPOSITION 7. *The spm-semiring* $(\mathbb{N}_e[B,X]/\asymp,+_e,\times_e,-_e,0_e,1_e)$ *is universal in the class of spm-semirings.*

PROOF. This readily follows from Theorem 2 and Lemma 4 by leveraging the universe seba-structure from which the spm-semiring is derived. □

7. PROVENANCE

We next show how the universal spm-semiring (and seba-structure) can be used in practice. More specifically, we first show that—in analogy to the relational case— the evaluation of SPARQL queries on annotated RDF factors through the universal spm-semiring. We then show how to perform computations in the universal spm-semiring and illustrate that the use of spm-semirings as annotation structure indeed generalizes the different semantics of SPARQL given in Section 2. We conclude by observing that the universal spm-semiring can be used for recording the *how-provenance* [16], among other things, of SPARQL query results.

Factorization of SPARQL query evaluation. Let K and L be two spm-semirings. A homomorphism $h: K \to L$ is said to *commute* with a SPARQL expression P if for any K-annotated RDF graph G_a:

$$[\![P]\!]_{h(G_a)} = h\big([\![P]\!]_{G_a}\big),$$

where $h(G_a)$ is the L-annotated RDF graph obtained from G_a by replacing $(s,p,o) \mapsto k$ by $(s,p,o) \mapsto h(k)$, for each K-annotated RDF triple $(s,p,o) \mapsto k$ in G_a. The following can be readily shown by induction on the structure of SPARQL expressions.

PROPOSITION 8. *Let h be a mapping between two spm-semirings $(K, \oplus_K, \otimes_K, \ominus_K, 0_K, 1_K)$ and $(L, \oplus_L, \otimes_L, \ominus_L, 0_L, 1_L)$. The transformation given by h from K-annotated to L-annotated RDF graphs commutes with all SPARQL expressions if and only if h is a homomorphism between spm-semirings.* □

DEFINITION 8. Let G_a be a K-annotated RDF graph and let $X = \{x_1, \ldots, x_n\}$ be a set of variables, one for each triple in G_a. The *abstractly tagged* version of G_a, denoted by G_a/X, is the X-annotated RDF graph in which each triple (s,p,o) in G is annotated with its corresponding variable in X. Figure 3(b) shows the abstractly tagged version of the RDF graph given in Figure 1(b). □

Recall the universal spm-semiring $(\mathbb{N}_e[B,X]/\asymp, +_e, \times_e, -_e, 0_e, 1_e)$ from the previous section. We regard the abstractly tagged version G_a/X of G_a as an RDF graph annotated by elements in $\mathbb{N}_e[B,X]/\asymp$ by interpreting x_i as its equivalence class $x_i/\asymp$ in $X/\asymp$. Let K be an spm-semiring and consider a valuation $\nu: X/\asymp \to K$. Denote by $Eval_\nu$ the unique homomorphism from $(\mathbb{N}_e[B,X]/\asymp, +_e, \times_e, -_e, 0_e, 1_e)$ to K that coincides with ν on $X/\asymp$. Proposition 8 implies that the evaluation of SPARQL queries factors through the universal spm-semiring:

PROPOSITION 9. *Let K be an spm-semiring and P be a SPARQL expression. For any K-annotated RDF graph G_a we have that*

$$[\![P]\!]_{G_a} = Eval_\nu\big([\![P]\!]_{G_a/X)}\big),$$

where G_a/X is the abstractly tagged version of G_a and $\nu: X/\asymp \to K$ is the function that associates with each equivalence class $x_i/\asymp$ the unique annotation of the triple in G_a tagged with x_i. □

Computing with the universal spm-semiring. We next show how to compute the annotations in $\text{Eval}_\nu\big([\![P]\!]_{G_a/X)}\big)$. Although the elements in $\mathbb{N}_e[B,X]/\asymp$ are equivalence classes, we can simply work with a single representative in each of these classes. Let $p[B,X]$ and $q[B,X]$ be two polynomials in $\mathbb{N}_e[B,X]$. Clearly, $p[B,X]$ belongs to $p[B,X]/\asymp$ and $q[B,X]$ belongs to $q[B,X]/\asymp$. Algorithm 1 shows how to find a representative in the class corresponding to the addition, multiplication and difference of $p[B,X]/\asymp$ and $q[B,X]/\asymp$. In a nutshell, the algorithm first computes the normal forms of $p[B,X]$ and $q[B,X]$[2] and applies the definition $+_e$, $\times_e$ and $-_e$ on these normal forms. A simplification procedure SIMPLIFY is consecutively applied. This procedure only extracts the terms in the support of the extended polynomial and in addition eliminates "redundant" boolean variables. Here, a boolean variable b_i is redundant if the normal form contains $\mathbf{b}^\beta p[X] + \mathbf{b}^{\beta'} p[X]$ for some polynomial $p[X]$ in which x_i does not appear and such that $\beta(j) = \beta'(n+j)$ for all $j \in [1,n]$, $i \neq j$, and $\beta(i) = 0$ and $\beta'(n+i) = 1$.

EXAMPLE 14. Let $p[B,X] = 2x_1^2 + b_1 x_2$ and $q[B,X] = 3b_2 x_1 + x_2^3$. Then $p_{nf}[B,X] = b_1 b_2 (2x_1^2 + x_2) + b_1 \bar{b}_2 (2x_1^2)$ and $q_{nf}[B,X] = b_1 b_2 (3x_1 + x_2^3) + \bar{b}_1 b_2 (x_2^3)$, where we only write the terms in the supports of the normal forms. Then, a representative in $(p[B,X]/\asymp) +_e (q[B,X]/\asymp)$ is given by the algorithm as $b_1 b_2 (x_2^3 + 2x_1^2 + 3x_1 + x_2) + b_1 \bar{b}_2 (2x_1^2) + \bar{b}_1 b_2 (x_2^3)$ which can be simplified to $2x_1^2 + x_2^3 + b_1 b_2 (3x_1 + x_2)$. Similarly, a representative in $(p[B,X]/\asymp) \times_e (q[B,X]/\asymp)$ is given by $b_1 b_2 ((2x_1^2 + x_2) \cdot (3x_1 + x_2^3)) + b_1 \bar{b}_2 (2x_1^2) + \bar{b}_1 b_2 (x_2^3)$. Finally, a representative in $(p[B,X]/\asymp) -_e (q[B,X]/\asymp)$ is given by $b_1 \bar{b}_2 (2x_1^2)$. □

[2]A practical implementation of this algorithm would, in fact, only need to compute a part of these normal forms, involving just the variables appearing in the two extended polynomials.

Algorithm 1 Universal spm-semiring computations.

Input: Extended polynomials $p[B,X]$ and $q[B,X]$.
Output: Representative in $(p[B,X]/\asymp \text{op} (q[B,X]/\asymp)$ for $\text{op} \in \{+_e, \times_e, -_e\}$.

1: Compute $p_{nf}[B,X] = \sum_{\beta \in BIM} \mathbf{b}^\beta p_\beta[X]$ of $p[B,X]$;
2: Compute $q_{nf}[B,X] = \sum_{\beta \in BIM} \mathbf{b}^\beta q_\beta[X]$ of $q[B,X]$;
3: **if** $\text{op} = +_e$ **then**
4: **return** SIMPLIFY$(\sum_{\beta \in BIM} \mathbf{b}^\beta (p_\beta[X] + q_\beta[X]))$;
5: **end if**
6: **if** $\text{op} = \times_e$ **then**
7: **return** SIMPLIFY$(\sum_{\beta \in BIM} \mathbf{b}^\beta (p_\beta[X] \cdot q_\beta[X]))$;
8: **end if**
9: **if** $\text{op} = -_e$ **then**
10: **return** SIMPLIFY$(\sum_{\beta \in BIM \setminus S} \mathbf{b}^\beta p_\beta[X])$, where S is the support of $q_{nf}[B,X]$.
11: **end if**

By the semantics of SPARQL on spm-semiring annotated RDF graphs, Algorithm 1 suffices to compute $[\![P]\!]_{G_a/X}$ for a SPARQL expression P. Indeed, addition, multiplication and difference suffice to compute the annotations of SPARQL query results (cf. Section 3).

EXAMPLE 15. Consider the SPARQL expressions $\Omega_1, \ldots, \Omega_4$, $\Omega_4 \bowtie \Omega_1$, $\Omega_4 \setminus \Omega_1$ and $\Omega_4 \mathbin{⟕} \Omega_1$ described in Example 1 and shown in Fig. 1. Fig. 3(c),(d),(e) and (f) show $[\![\Omega_i]\!]_{G_a/X}$, for $i = 1,2,3,4$, respectively. Furthermore, $[\![\Omega_4 \bowtie \Omega_1]\!]_{G_a/X}$, $[\![\Omega_4 \setminus \Omega_1]\!]_{G_a/X}$, and $[\![\Omega_4 \mathbin{⟕} \Omega_1]\!]_{G_a/X}$ are shown in Fig. 3(g),(h) and (i), respectively. Consider the mapping μ_{11} in Ω_3. Its annotation is a representative in $(x_4/\asymp) +_e (x_5/\asymp)$. It is readily verified that Algorithm 1 returns $x_4 + x_5$ Similarly, μ_{16} in $\Omega_4 \bowtie \Omega_1$ is annotated by the representative $x_1(x_4 + x_5)$ in $(x_1/\asymp) \times_e ((x_4 + x_5)/\asymp)$ as returned by the algorithm. Consider next μ_{18} in $\Omega_4 \setminus \Omega_1$. This mapping is annotated by a representative in $((x_4 + x_5)/\asymp) -_e (x_1 // \asymp)$. By definition of $-_e$ this is equal to the normal form of $(x_4 + x_5)$ multiplied (using $\times_e$) with f_S where S is the complement of the support of the normal form of x_1. It is readily verified that S consists of all boolean indices β in *BIM* except those with $\beta(1) = 1$. Hence, the annotation of μ_{18} is $\bar{b}_1(x_4 + x_5)$. □

Proposition 9 tells that one can deduce the right annotations for any spm-semiring K and any SPARQL expression P, given $[\![P]\!]_{G_a/X}$ and a valuation $\nu: X/\asymp \to K$. We illustrate this for the bag, trust (set) and fuzzy setting.

EXAMPLE 16. Consider the mapping μ_{18} in Figure 3(h). It has $\bar{b}_1(x_4 + x_5)$ as annotation in the universal spm-semiring. In the bag semantics in Example 1, we have that $\nu_b: (x_i/\asymp) \to 1$ for all x_i. Since we know that $Eval_{\nu_b}(\bar{b}_1(x_4 + x_5)) = h_s(\bar{b}_1(x_4 + x_5))$ where h_s is the seba-homomorphism from the universal seba-structure to the seba-structure corresponding to $(\mathbb{N}, +, \times, -_{bag}, 0, 1)$ given in Example 7, we get that $h_s(\bar{b}_1(x_4 + x_5)) = 0 \times (1 + 1) = 0$, as desired. Similarly, for the trust semantics in Example 1, $\nu_t: (x_i/\asymp) \to \tau_i$ for all x_i. Taking the seba-structure corresponding

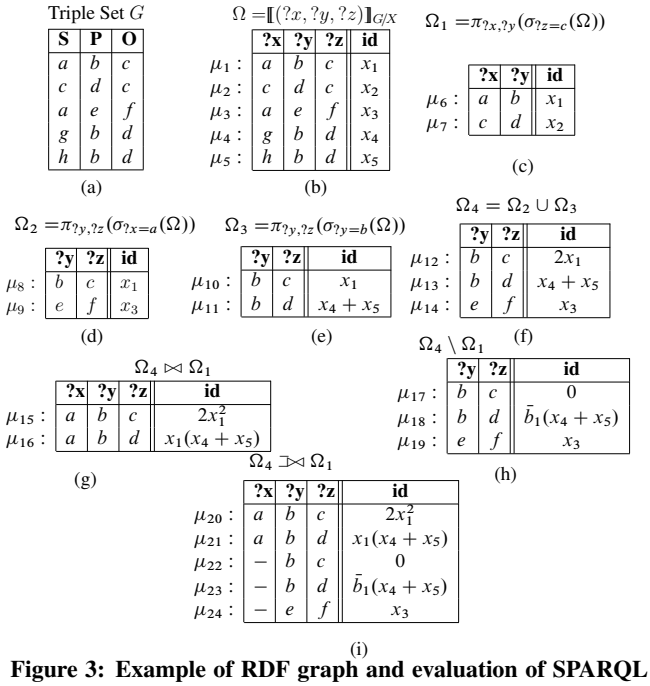

Figure 3: Example of RDF graph and evaluation of SPARQL algebra operators with annotations in the universal spm-semiring.

to $(\{\mathsf{true},\mathsf{false}\},\vee,\wedge,-_{\mathit{trust}},\mathsf{true},\mathsf{false})$ given in Example 5, we thus get that $\mathit{Eval}_{\nu_t}(\bar{b}_1(x_4+x_5)) = \bar{\tau}_1 \wedge (\tau_4 \vee \tau_5)$, as desired. The fuzzy semantics is verified similarly using the seba-structure corresponding to $([0,1],\min,\max,-_{\mathit{fuzzy}},0,1)$ given in Example 7. □

We may thus conclude that the expressions in $(\mathbb{N}_e[B,X]/\asymp, +_e, \times_e, -_e, 0_e, 1_e)$ generalize all spm-semiring computations, and in addition, these expressions reveal to some extent how mappings in the query result were created. For instance, Example 15 shows that μ_{18} is annotated with $\bar{b}_1(x_4+x_5)$ which is equivalent $(x_4+x_5)-_e b_1$. In other words, μ_{18} is obtained by first taking a union, followed by a difference. That is, the universal spm-semiring records the *how-provenance* in the SPARQL setting.

8. RELATED WORK

This work is inspired by the algebraic approach to modeling data provenance, initiated by Green et al. [16]. They consider various forms of annotated relational data and their transformations by means of positive relational queries. In that setting, they show that standard positive relational algebra equivalences hold on annotated relational data if and only if the annotations have the structure of a (commutative) semiring [16]. Furthermore, they propose the semiring of *polynomials*—the *universal* semiring—as a provenance model that generalizes many forms of annotations and previously proposed provenance models [16, 14]. Due to its universality, all annotation computations were shown to factor through the semiring of polynomials. This work has been extended to the semi-structured setting in which transformations are expressed in a subset of XQuery [11], for which the authors showed that semirings are still an appropriate annotation structure. Similarly, semirings are sufficient for *positive* SPARQL queries on annotated RDF data, as described in Proposition 1 (Section 4) and also observed in [26].

The situation becomes more challenging when non-monotone query operators are taken into account [12, 1, 2, 26, 13]. In the relational setting, Geerts et al. [12] extended semirings with a *monus* operation, that captures the semantics of relational difference, and proposed a universal *monus*-semiring, or m-semiring for short, as a provenance model. However, this universal m-semiring does not allow for a simple representation of its elements. Indeed, it is built up from formal terms that require the arbitrary nesting of expressions of the form $p[X]-_m q[X]$, where $p[X]$ and $q[X]$ are terms that are built up from variables in X, $+$, $\times$ and the monus $-_m$. A study of the properties of m-semirings can be found in [1]. More specifically, a set of identities $\mathcal{E}_m$ is identified that characterizes m-semirings. Since the relational difference satisfies two sets of incompatible equivalences, $\mathcal{E}_s$ and $\mathcal{E}_b$, in the set and bag semantics, respectively, $\mathcal{E}_m$ only considers the common identities in these sets. As a consequence, certain intuitive equivalences of the relational algebra, such as $R \bowtie (S \setminus T) = (R \bowtie S) \setminus (R \bowtie T)$ ($A13$ in [1]) are not necessarily satisfied on m-semiring annotated relational data [1].

The classes of m-semirings and spm-semirings are incomparable, however. Indeed, there are identities that hold for m-semirings but not for spm-semirings, and vice versa. For example, spm-semirings satisfy $\mathsf{id}_{11} : k_1 \otimes (k_2 \ominus k_3) = (k_1 \otimes k_2) \ominus k_3$, a stronger identity than $k_1 \otimes (k_2 \ominus k_3) = (k_1 \otimes k_2) \ominus (k_1 \otimes k_3)$ ($A13$ in [1]) implied by the above equivalence. It is readily shown that m-semirings do not satisfy this property. Conversely, $k_1 \oplus (k_2 \ominus k_1) = k_2 \oplus (k_1 \ominus k_2)$ ($A11$ in [1]) holds for m-semirings but is not satisfied by spm-semirings. Furthermore, the identities of spm-semirings imply that the elements in the universal spm-semiring have a normal form representation, in contrast to elements in the universal m-semiring.

Amsterdamer et al. [2] obtained an altermative semantics for relational difference based on their semantics for queries with *aggregation* on annotated relations, through an encoding of difference using aggregation. Interestingly, the semantics of the difference defined in this manner seems similar to the semantics of SPARQL difference. However, the resulting annotations reflect the encoding of difference through aggregation and thus do not provide a very intuitive description of the actual operations in the original query. Moreover, they do not propose a universal object that could be used as the provenance model for queries with difference under these semantics. Similarly to SPARQL difference, aggregation-based difference fails to satisfy $A11$ that holds for m-semirings. However, it satisfies $A13$, which is not satisfied by all spm-semirings. This implies that even if semirings equipped with the aggregate-based difference would be spm-semirings, which is not clear, it is necessarily a strict subset of spm-semirings.

The Perm system [13] employs a provenance model that captures relational difference and outer join under set and bag semantics. However, it cannot capture SPARQL `DIFFERENCE` and `OPTIONAL` directly, since their semantics differs from that of the corresponding relational operators, as explained in Section 2. Moreover, the provenance model of Perm, which is akin to why-provenance extended with $\wedge\neg$ for difference, is less informative than spm-semirings and does not suffice for computing annotations such as ranked trust or multiplicities for bag semantics.

In the Semantic Web community, Damasio et al. [9] employ m-semirings to capture the semantics of SPARQL query answering over annotated RDF. More precisely, they use (m,δ)-semirings which are m-semirings extended with a duplicate elimination operator δ, as introduced in [12]. Then, they encode SPARQL difference through a complex relational expression involving joins, relational set difference and duplicate elimination. In fact, their encoding of the SPARQL difference resembles somewhat that of Amsterdamer et al. [2]. However, (m,δ)-semirings have the same deficiency as m-semirings: their universal structure does not allow for a simple representation of its elements and is completely symbolic and not amenable to algebraic manipulation. For this reason, it is not as well-suited to be used as a provenance model

as the structure we propose in Section 6. Indeed, in order to use the resulting expressions to compute, e.g., trust annotations, the authors resort to a simpler model, by fixing the duplicate elimination function δ, thereby disregarding all (m, δ)-semirings with a more complex δ. Furthermore, similar to the approach taken in [2], the resulting expressions do not reflect the structure of operators in the original SPARQL query. Unfortunately, it is also unclear what identities are satisfied by this encoding of SPARQL difference, so a formal comparison between the resulting structure and spm-semirings is difficult to obtain.

As already mentioned in the introduction, our work aims to unify the semantics of SPARQL query answering on various notions of annotated RDF. These various semantics include: ownership [8], truth of imprecise (fuzzy) information [22], trust [4, 18]. Some of these works focus only on representation formats for such annotations, while others also propose semantics for query answering and inference. They do not attempt to generalize the various SPARQL semantics nor do they use algebraic structures to model the annotations.

There has also been work on using algebraic structures to model and unify such annotations [27, 6, 28], but these focus on transformations of annotated RDF through RDFS rules, used to infer implicit information. However, inference rules do not involve the use of the SPARQL `OPTIONAL` operator, which is the main focus of our work, and thus, the algebraic structures employed in these works do not capture its semantics. Moreover, [28] extends SPARQL such that queries can *explicit*ly manipulate both data and annotations. In contrast, we consider *implicit* provenance [5] in which annotations are simply carried along when the data is queried using standard SPARQL queries. Implicit provenance for SPARQL has also been studied by Dividino et al. [10], but they only consider a limited fragment of SPARQL that does not include the `OPTIONAL` (or `DIFFERENCE`) operators, in contrast to our work.

9. CONCLUSIONS AND FUTURE WORK

We have presented spm-semirings, an extension of semirings to capture the semantics of SPARQL queries, involving the non-monotone operator `OPTIONAL`, on annotated RDF data. Moreover, we showed that spm-semirings have a universal structure that provides a concise representation of the provenance of RDF data and SPARQL queries with the `OPTIONAL` operator. As in the relational case with provenance polynomials [16], provenance expressions from this universal structure can be recorded during query answering and later be evaluated in appropriate spm-semirings in order to compute different forms of annotations for a variety of applications.

Some of these applications may not require the full expressiveness of this universal structure. As in the relational case, for such applications it may be desirable to record provenance expressions from a less informative model instead, e.g., if such expressions are more efficient to store and evaluate than those of more informative provenance models. For this reason, in future work we intend to explore whether simpler provenance models for relational queries, such as why-provenance, can be extended to capture the semantics of SPARQL queries for certain kinds of annotated RDF. More generally, we want to perform a formal comparison of spm-semirings with various algebraic structures intended to capture the semantics of SPARQL `OPTIONAL` [9, 2] and explore whether those structures, as well as the simpler ones we expect to obtain from extensions of relational provenance models, can be expressed as spm-semirings and arranged in a hierarchy in the style of relational annotation structures and provenance models [14].

Finally, SPARQL 1.1 [17] includes non-monotone operators `NOT EXISTS` and `MINUS`, that differ from the `DIFFERENCE` operator considered in this paper. Indeed, `MINUS` considers a stronger notion of compatibility of mappings by requiring the presence of common variables; `NOT EXISTS` asks for the absence of certain patterns as part of a `FILTER` construct. The study of SPARQL query equivalences involving `NOT EXISTS` and `MINUS`, as well as the search for an appropriate algebraic annotation structure and provenance model for SPARQL 1.1. is deferred to future work.

10. REFERENCES

[1] Y. Amsterdamer, D. Deutch, and V. Tannen. On the limitations of provenance for queries with difference. In *TaPP*, 2011.

[2] Y. Amsterdamer, D. Deutch, and V. Tannen. Provenance for aggregate queries. In *PODS*, 2011.

[3] M. Arenas and J. Pérez. Querying semantic web data with SPARQL. In *PODS*, 2011.

[4] D. Artz and Y. Gil. A survey of trust in computer science and the semantic web. *Web Semantics*, 5(2), 2007.

[5] P. Buneman, J. Cheney, and S. Vansummeren. On the expressiveness of implicit provenance in query and update languages. *ACM TODS*, 33(4), 2008.

[6] P. Buneman and E. V. Kostylev. Annotation algebras for RDFS. In *SWPM*, 2011.

[7] S. Burris and H. P. Sankappanavar. *A Course in Universal Algebra*. Number 78 in Graduate Texts in Mathematics. Springer-Verlag, 1981.

[8] J. J. Carroll, C. Bizer, P. J. Hayes, and P. Stickler. Named graphs. *J. Web Semantics*, 3(4), 2005.

[9] C. Damasio, A. Analyti, and G. Antoniou. Provenance for SPARQL queries. In *ISWC*, 2012.

[10] R. Dividino, S. Sizov, S. Staab, and B. Schueler. Querying for provenance, trust, uncertainty and other meta knowledge in RDF. *J. Web Semantics*, 7(3), 2009.

[11] J. N. Foster, T. J. Green, and V. Tannen. Annotated XML: queries and provenance. In *PODS*, 2008.

[12] F. Geerts and A. Poggi. On database query languages for k-relations. *Journal of Applied Logic*, 8(2), 2010.

[13] B. Glavic and G. Alonso. Perm: Processing provenance and data on the same data model through query rewriting. In *ICDE*, 2009.

[14] T. J. Green. Containment of conjunctive queries on annotated relations. In *ICDT*, 2009.

[15] T. J. Green, G. Karvounarakis, Z. G. Ives, and V. Tannen. Update exchange with mappings and provenance. In *VLDB*, 2007.

[16] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, 2007.

[17] S. Harris and A. Seaborne. SPARQL 1.1 query language. `www.w3.org/TR/sparql11-query`, November 2012.

[18] O. Hartig. Querying trust in RDF data with tSPARQL. In *ESWC*, 2009.

[19] H. Huang and C. Liu. Query evaluation on probabilistic RDF databases. In *WISE*, 2009.

[20] G. Karvounarakis, Z. G. Ives, and V. Tannen. Querying data provenance. In *SIGMOD*, 2010.

[21] F. Manola, E. Miller, and B. McBride. RDF primer. `www.w3.org/TR/rdf-primer`, February 2004.

[22] M. Mazzieri and A. F. Dragoni. A fuzzy semantics for the resource description framework. In *URSW*, 2008.

[23] J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. *ACM TODS*, 34(3), 2009.

[24] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. `www.w3.org/TR/rdf-sparql-query`, January 2008.

[25] M. Schmidt, M. Meier, and G. Lausen. Foundations of SPARQL query optimization. In *ICDT*, 2010.

[26] Y. Theoharis, I. Fundulaki, G. Karvounarakis, and V. Christophides. On provenance of queries on semantic web data. *IEEE IC*, 2011.

[27] O. Udrea, D. R. Recupero, and V. S. Subrahmanian. Annotated RDF. *ACM Trans. Comput. Log.*, 11(2), 2010.

[28] A. Zimmermann, N. Lopes, A. Polleres, and U. Straccia. A general framework for representing, reasoning and querying with annotated semantic web data. *J. Web Semantics*, 11, 2012.