Algebraic Structures for Capturing the Provenance of SPARQL Queries

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

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

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

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 to 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. More specifically, for any two RDF data sets \(G_1\) and \(G_2\), \((G_1\text{ OPTIONAL }G_2)\) can be written as \((G_1 \text{ AND } G_2) \text{ UNION } (G_1 \text{ DIFFERENCE } G_2)\) [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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

EDBT/ICDT ’13, March 18 – 22 2013, Genoa, Italy
Copyright 2013 ACM 978-1-4503-1598-2/13/03 ...$15.00.

\footnote{This work was partially supported by the EU FP7 project IP-601043}

\footnote{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 \( \ominus \), \( \ominus \) and \( \ominus \), and constants 0 and 1, that accommodate all \( K \)-annotated RDF 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, \((K, \ominus, \ominus, \ominus, 0, 1)\) 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 \( \sigma \) and \( \pi \) that correspond to the SPARQL constructs \( \text{FILTER} \) and \( \text{SELECT} \), respectively, and b) binary operators \( \cup \), \( \otimes \), and \( \ominus \) for the SPARQL constructs \( \text{UNION} \), \( \text{AND} \), and \( \text{OPTIONAL} \), respectively. The operator \( \ominus \) can be defined in terms of \( \cup \), \( \otimes \) and \( \ominus \), 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 \( \Omega \) 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 \((\text{trust})\) and \((\text{full}\{?)\}) can be ignored for now, as can the gray shaded entries in Tables (b) and (i). We further employ symbols \( \mu \) to identify individual mappings.

Tables (c–e) illustrate the evaluation of the operators \( \sigma \) and \( \pi \). The output mapping bags are denoted by \( \Omega_1 \), \( \Omega_2 \) and \( \Omega_3 \). For instance, mapping \( \mu_{11} \) of \( \Omega_1 \) has two derivations originating from mappings in \( \Omega \), namely one by projecting \( \mu_4 \) and another one by projecting \( \mu_6 \). The multiplicity of \( \mu_{11} \) is obtained by adding the multiplicities of \( \mu_4 \) and \( \mu_6 \).

Table (f) shows the result of \( \Omega_2 = \Omega_2 \cup \Omega_3 \), where \( \Omega_2 \) and \( \Omega_3 \) 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 \( \Omega_4 \), mapping \( \mu_{12} \) has two derivations, both of them originating from \( \Omega_4 \), while the two derivations of \( \mu_{13} \) originate from \( \Omega_4 \) (\( \mu_5 \)) and \( \Omega_4 \) (\( \mu_7 \)). The multiplicity of \( \mu_{13} \) is obtained by adding the multiplicities of \( \mu_5 \) and \( \mu_7 \).

Table (g) depicts the result of \( \Omega_4 = \Omega_4 \cap \Omega_1 \). For instance, mapping \( \mu_{16} \) is derived by joining \( \mu_{13} \in \Omega_4 \) and \( \mu_6 \in \Omega_4 \). Mappings can be joined only if they are compatible [23]. In our example, \( \mu_6 \) and \( \mu_{13} \) are compatible because they agree on their common variable, i.e., they both bind \(?y\) to \( b \). The multiplicity of \( \mu_{16} \) 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., \( \Omega_4 \setminus \Omega_4 \). Note that neither \( \mu_{12} \) nor \( \mu_{13} \) in \( \Omega_4 \) are in the result – recall that we ignore the gray shaded entries for now – because \( \Omega_1 \) contains a mapping \( (\mu_0 \) that is compatible with the mappings \( \mu_{12} \) and \( \mu_{13} \) in \( \Omega_4 \). On the other hand, there is no mapping in \( \Omega_1 \) that is compatible with \( \mu_{14} \). As a consequence, \( \mu_{14} \) appears in the result as \( \mu_{10} \) in Table (b). Similarly to the union, the difference operator can be applied on bags of mappings defined on different variables.
Finally, Table (i) depicts the result of $\Omega_4 \triangleright \Omega_1$, which contains all mappings from Tables (g) and (h). Indeed, $\Omega_4 \triangleright \Omega_1 = (\Omega_4 \setminus \Omega_1) \cup (\Omega_1 \setminus \Omega_1)$. The symbol $\triangleright$ denotes that $？x$ is not bound to a constant in the mappings $\mu_{22}$, $\mu_{23}$ and $\mu_{24}$. 

The previous example shows that the multiplicities of mappings in the result of $\sigma$, $\pi$, $\cup$ and $\triangleright$ 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 $\cap$ or $\cup$) of a mapping, its multiplicity equals the sum of the multiplicities of the different derivations. In the case of $\triangleright$, 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 ($\setminus_{\text{bag}}$) 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, $(R \setminus S)(t) := \max(0, R(t) - S(t))$ 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 $\Omega_4$ is in the output as long as there is no mapping in $\Omega_1$ that is compatible with it. That is, $(\Omega_4 \setminus \Omega_1)(t) = \Omega_4(t) - \hat{\mu}_{\text{bag}}(\Omega_1(t))$, where for any two natural numbers $n$ and $m$, $n - \hat{\mu}_{\text{bag}} m = n$ in case that $m = 0$, and $n - \hat{\mu}_{\text{bag}} m = 0$ 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 OPTIMAL.

EXAMPLE 2 (BAGS CONT’D). Consider Tables (c), (f) and (h) of Fig. 1. Observe that although $\mu_{12}$ and $\mu_{13}$ both have multiplicity 2 in $\Omega_4$, and the compatible mappings $\mu_6$ and $\mu_7$ in $\Omega_4$ have multiplicity 1, neither $\mu_{12}$ nor $\mu_{13}$ is present in $\Omega_4 \setminus \Omega_1$. In contrast, in the relational bag semantics for $\setminus_{\text{bag}}$, both $\mu_{12}$ and $\mu_{13}$ 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 $\text{trust}$ column. For example, each triple in Table (b) comes with a boolean trust value ($\tau_t$) 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 ($\lor$) and conjunction ($\land$), instead of addition and multiplication, respectively, used to compute multiplicities in Example 1. For example, $\mu_{14}$ in Table (c) is trusted if only one of the mappings $\mu_4$ or $\mu_5$ is trusted. Similarly, $\mu_{16}$ in Table (g) is trusted if $\mu_4$ is trusted and either $\mu_4$ or $\mu_5$ is trusted. To deal with $\setminus$, we consider boolean negation (denoted by $\bar{\tau}$ for a trust variable $\tau$). Indeed, consider the gray shaded entry $\mu_{16}$ in Table (b). This mapping is trusted only if $\mu_4$ is untrusted and either $\mu_4$ or $\mu_5$ is trusted. This is expressed by $(\mu_4 \lor \mu_5) - \mu_4 \land \mu_5$. Hence, the gray shaded mappings can be part of the result depending on the trust information of the source mappings. In general, if $\varphi$ and $\psi$ are two propositional formulas over boolean variables, then their difference is defined as $\varphi - \mu_{\text{bag}} \psi := \overline{\varphi \land \psi}$. Note that this is similar to the notion of difference given in the bag semantics (cf. Example 1); $\varphi \lor \psi$ equals $\varphi$ in case that $\psi$ 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 $\text{fuzzy}$ column. For example, each triple in Table (b) comes with a probability ($p_t$) 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).
Table (h) appears with probability $p_k$ mappings that take values in some abstract set of annotations as follows: First, we extend mappings to annotated Summary and lookahead $-\mu$.

To deal with $\gamma$, we need to consider an additional operator $-\text{fuzzy}$ on probabilities that is defined as $p - \text{fuzzy} q := 0$ in case that $q \neq 0$, and $p - \text{fuzzy} q := p$ otherwise. For example, the gray shaded entry $\mu_{18}$ in Table (h) appears with probability $\max(p_k, p_{18}) - \text{fuzzy} p_{18}$. That is, if $\mu_{18}$ has non-zero probability, then $\mu_{18}$ will not appear in the query result.

Similarly, if both $p_k$ and $p_{18}$ have zero probability, then $\mu_{18}$ is not part of the output. In all other cases, $\mu_{18}$ is a result mapping with probability $\max(p_k, p_{18})$. Note again the similarity between $-\text{fuzzy}$, $-\text{bag}$ and $-\text{fuzzy}$

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 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 $\odot$ for modeling $+$, $\lor$ and $\max$, among others, for the operators $\pi$ and $\cup$.
- A binary operator $\odot$ for modeling $\times$, $\land$ and $\min$, among others, for the operators $\times$ and $\otimes$; and
- A binary operator $\odot$ for modeling $-\text{bag}$, $-\text{bag}$ and $-\text{fuzzy}$, among others, for the operators $\setminus$ and $\otimes$.

Finally, based on SPARQL query equivalence 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 $K$, $\odot$, $\otimes$ and $\odot$ 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 $(s, p, o) \in (I \cup B) \times (I \cup B \cup L)$ 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 $(s, p, o) \rightarrow k$ 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 $(P_1 \cup P_2)$, $(P_1 \setminus P_2)$, $(P_1 \setminus P_2)$ and $(P_1 \setminus P_2)$ are also graph patterns.
3. If $P$ is a graph pattern and $R$ is a SPARQL built-in condition, then $(P \text{FILTER} R)$ 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 $\text{SELECT}_{S}(P)$ 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 $\varphi(t)$ 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 $\mu$ from $V$ to $T$ is a partial function $\mu : V \rightarrow T$. The domain of $\mu$, denoted by $\text{dom}(\mu)$, is the subset of $V$ on which $\mu$ is defined. We say that two mappings $\mu_1$ and $\mu_2$ are compatible when for all $v \in \text{dom}(\mu_1) \cap \text{dom}(\mu_2)$, 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 $T$.

**Definition 2.** Let $K$ be a set of annotations and let $0$ denote a distinguished element from $K$. A $K$-annotated mapping set on $M$ is a total function $k : M \rightarrow K$ such that its support $\{ \mu \in M, k(\mu) \neq 0 \}$ 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 $\sigma$, $\pi$, $\cup$, $\otimes$, and $\odot$ on $K$-annotated mapping sets. As indicated at the end of the previous section, we assume the presence of binary operators $\odot$, $\odot$ and $\odot$ and distinguished constant elements $0$ and $1$ in the set of annotations. At this point $K, \odot, \odot, \odot, 0, 1$ 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 $k$ be a $K$-annotated mapping set on $M$ and let $S \subseteq V$. Then for all $\mu \in M$ and $\nu \in M$ such that $\text{dom}(\nu) \subseteq S$:

$$(\sigma_k(\nu))(\mu) := \left\{ \begin{array}{ll} k(\mu) & \text{if } \nu \sim \mu \text{, and} \mu \in M \cap \text{dom}(\mu') \end{array} \right.$$
has finite support since RDF graphs are finite objects. The semantics of the remaining SPARQL graph patterns is defined as follows:

\[
\begin{align*}
&P_1 \text{ FILTER } R \varnothing_a := \pi_R (P_1 \varnothing_a) \\
&P_1 \text{ SELECT } E \varnothing_a := \{ e \in P_1 \varnothing_a \mid \sigma_e \} \\
&P_1 \text{ UNION } P_2 \varnothing_a := P_1 \varnothing_a \cup P_2 \varnothing_a \\
&P_1 \text{ AND } P_2 \varnothing_a := P_1 \varnothing_a \sqcap P_2 \varnothing_a \\
&P_1 \text{ DIFFERENCE } P_2 \varnothing_a := \{ P_1 \varnothing_a \setminus P_2 \varnothing_a \} \\
&P_1 \text{ OPTIONAL } P_2 \varnothing_a := \{ P_1 \varnothing_a \sqcup P_2 \varnothing_a \}
\end{align*}
\]

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. Indeed, recall that a \( K \)-annotated mapping set has a finite support but the choice of \( \oplus \) causes infinitely many result tuples. 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 \( F \) of \( \{ 0, 1, \oplus, \ominus, \otimes, \bigcirc \} \), \{ true, false \}, \{ \land, \lor \}, and \( \{ 0, 1 \} \): 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 \((K, \oplus, \ominus, \otimes, \bigcirc, 0, 1)\).

Similar to the relational case, SPARQL optimization techniques rely on SPARQL algebra equivalences and identities on \((K, \oplus, \ominus, \otimes, \bigcirc, 0, 1)\). For the bag semantics of SPARQL, an extensive list of such equivalences is identified by Schmidt et al. [25]. For example, the equivalence 

\[ P_1 \cup P_2 \equiv P_1 \cup P_2 \forall \text{ holds for all SPARQL algebra expressions } P_1 \text{ and } P_2. \]

That is, for any RDF graph \( G \), \( [P_1]_G \cup [P_2]_G = [P_1 \cup P_2]_G \). On the other hand, \( \bigcirc \) identities are commonly used to express conditions on algebraic structures such as \((K, \oplus, \ominus, \otimes, 0, 1)\). For example, the identity 

\[ k_1 \oplus k_2 = k_2 \ominus k_1 \]

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 quantifiacation.

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 \( k_1, \ldots, k_n \),

\[ P_1(k_1, \ldots, k_n) = P_2(k_1, \ldots, k_n). \]

We turn such an equivalence into an identity on \((K, \oplus, \ominus, \otimes, \bigcirc, 0, 1)\) 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 \( \pi_{\sigma_{\text{triple}}} \), indicating that all triples are selected; \( P \) is not allowed to contain projections, as these do not have a direct counterpart in \((K, \oplus, \ominus, \otimes, \bigcirc, 0, 1)\); 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 \( \text{ann}(P) \) inductively as follows: (i) if \( P = m_i \), then \( \text{ann}(m_i) = k_i \), where \( k_i \) is a variable to be interpreted as a value in \( K \); (ii) if \( P = Q \cup R \), then \( \text{ann}(P) = \text{ann}(Q) \oplus \text{ann}(R) \); if \( P = Q \sqcap R \), then \( \text{ann}(P) = \text{ann}(Q) \ominus \text{ann}(R) \); (iii) if \( P = Q \bigcirc R \), then \( \text{ann}(P) = \text{ann}(Q) \otimes \text{ann}(R) \); (iv) if \( P = Q \bigcirc R \), then \( \text{ann}(P) = \text{ann}(Q) \ominus (\text{ann}(Q) \oplus \text{ann}(R)) \); (v) if \( P = P_1 \ominus P_2 \), then \( \text{ann}(P) = \text{ann}(Q) \ominus 1 \); and finally, (vi) if \( P = \emptyset \), then \( \text{ann}(P) = 0 \).

By similar reasoning, given expression \( e \) using variables \( k_1, \ldots, k_n \), operations \( \ominus, \otimes, \bigcirc \), and constants \( 0 \) and \( 1 \), we can define a SPARQL expression \( \text{exp}(e) \) 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 \( \text{ann}(P_1) = \text{ann}(P_2) \) is an identity that holds on \((K, \oplus, \ominus, \otimes, \bigcirc, 0, 1)\). Conversely, if \( e_1 = e_2 \) is an identity that holds on \((K, \oplus, \ominus, \otimes, \bigcirc, 0, 1)\), then \( \exp(e_1) = \exp(e_2) \) 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 \((K, \oplus, \ominus, \otimes, \bigcirc, 0, 1)\). 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 \( \land \) and \( \forall \). 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 \( P_1, P_2, P_3, P_4 \), we have that

\[
\begin{align*}
E_1 = & \left\{ \sigma_{\text{triple}}(P_1) = P_1 \right\} \\
& \left\{ P_1 \sqcup P_2 = P_1 \right\} \\
& \left\{ P_1 \sqcap P_2 = P_1 \right\} \\
& \left\{ P_1 \bigcirc P_2 = P_1 \right\} \\
& \left\{ P_1 \bigcirc P_2 = P_2 \right\} \\
& \left\{ P_1 \bigcirc P_2 = P_1 \bigcirc P_3 \right\} \\
& \left\{ P_1 \sqcap (P_2 \sqcup P_3) = (P_1 \sqcap P_2) \sqcup P_3 \right\} \\
& \left\{ P_1 \bigcirc (P_2 \sqcup P_3) = (P_1 \bigcirc P_2) \sqcup P_3 \right\} \\
& \left\{ P_1 \bigcirc (P_2 \sqcup P_3) = P_1 \bigcirc (P_2 \sqcup P_3) \right\}
\end{align*}
\]

The following Proposition is a direct consequence of Lemma 1. Indeed, the SPARQL equivalences in \( E_1 \) precisely correspond to the set of semiring identities, shown as \( i_1 \cdots i_6 \) in Fig. 2.

**Proposition 1.** The positive fragment of the SPARQL \( K \)-annotated algebra satisfies the equivalences in \( E_1 \) if and only if \((K, \oplus, \ominus, \otimes, \bigcirc, 0, 1)\) is a semiring. Furthermore, for any positive SPARQL algebra expression \( P \) and \( K \)-annotated RDF graph \( G \), \( [P]_G \) has a finite support.

**Proof.** The correspondence between equivalences in \( E_1 \) and the semiring identities \( i_1 \cdots i_6 \) 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.
Section 2 suggest that the operation $\odot$ should satisfy $k_1 \odot k_2 = k_1$ in case that $k_2 = 0$ and $k_1 \odot k_2 = k_2$ if $k_2 \neq 0$ otherwise.

We first observe that the non-monotone operation $\ominus$ cannot be derived from the operations $+$ and $\odot$ 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 $(K, \odot, \ominus, \ominus, 0, 1)$ such that for any $k_1, k_2 \in K$, $k_1 \odot k_2 = k_2$ in case that $k_2 = 0$, and $k_1 \odot k_2 = k_1$ if $k_2 \neq 0$ 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 $V$ the corresponding variety. Let $(K, \odot, K, \ominus, K, 0, 1)$ and $(L, \odot, L, \ominus, L, 0, 1)$ be two algebraic structures in $V$ for which, in addition, $0_K \neq 1_K$ and $0_L \neq 1_L$ hold. Consider the canonical $\odot, \ominus$ on the product $K \times L$ defined as $(k_1, k_2) \odot (l_1, l_2) = (k_1 \odot k_2, l_1 \ominus l_2)$. The operations $\odot, \ominus, K, \ominus, K$ and constants $0_K, 1_K$ are similarly defined. Then, by assumption, $(K \times L, \odot, K, \ominus, K, 0_K, 1_K)$ is a structure in $V$ and thus $(1_K, 1_L) \odot (0_K, 1_L) = (1_K, 0_L) \neq (0_K, 0_L) = 0_{K, L}$ by the definition of $\odot, \ominus, K, \ominus, L$. This shows that $\ominus$ is not closed under taking products, a contradiction. 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 $P_3$:

$$E_2 = \left\{ P_1 \setminus P_2 \equiv P_1 \setminus (P_2 \cup P_3) \equiv (P_1 \setminus P_2) \cup P_3 \right\}.$$

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 $P_2$ that we have that

$$E_3 = \left\{ (P_1 \setminus (P_2 \cup P_3)) \cup (P_1 \setminus P_2) \equiv P_1 \right\}.$$

This equivalence also holds in all settings we have considered so far. By imposing $E_2$, $E_3$ and $E_3$ on the SPARQL $K$-annotated algebra, we obtain a generalization of Proposition 1 that accommodates for the difference $(\setminus)$ and optional operator $(\odot)$. The proof again relies on Lemma 1.

**Proposition 3.** The SPARQL $K$-annotated algebra satisfies the equivalences in $E_2$, $E_3$ and $E_3$ if and only if $(K, \odot, \ominus, 0, 1)$ is an algebraic structure satisfying the identities shown in Figure 2. Furthermore, for any SPARQL expression $P$ and $K$-annotated RDF graph $G$, $[P]_{G}$ has a finite support.

**Definition 3.** An spm-semiring is an algebraic structure $(K, \odot, \ominus, 0, 1)$ that satisfies the identities $\delta_{1} \Rightarrow \delta_{2}$ 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 $(\mathbb{N}, +, \times, -_{fuzz}, 0, 1), \{\text{true, false}\}, \lor, \land, \neg, \text{max}, \text{min}, -_{fuzz}, 0, 1)$ 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 $\mathcal{A} \setminus \{\delta_{2}, \delta_{3}\}$ is minimal.

Proof. Let $\mathcal{A}' = \mathcal{A} \setminus \{\delta_{2}, \delta_{3}\}$. It suffices to show the following: (i) for any identity $e \in \mathcal{A}'$, there exists a structure $K_e$ such that $K_e \models (\mathcal{A}' \cup \{e\})$ but $K_e \not\models e$; and (ii) for any $K'$ such that $K' \models \mathcal{A}'$, we have that $K' \models \delta_{2}$ and $K' \models \delta_{3}$.

**5. CHARACTERIZATION OF SPM-SEMI-RINGS**

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 $(K, B, d, i)$ where $K$ is a commutative semiring $(K, \odot, \ominus, 0, 1), B$ a boolean algebra $(B, \lor, \land, \neg, 1, 0, \top, \bot)$, $d: K \to B$ and $i: B \to K$ are mappings such that

$$d(0) = \bot$$

and $d(1) = \top$. For $e \models (\mathcal{A}' \cup \{e\})$ but $K_e \not\models e$; and (ii) for any $K'$ such that $K' \models \mathcal{A}'$, we have that $K' \models \delta_{2}$ and $K' \models \delta_{3}$.

**Intuitively, a seba-structure consists of a semiring $K$ in which a boolean algebra $B$ is embedded (by means of the mapping $i$), 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 $(L, \odot, \ominus, 0, 1)$ is derived from a seba-structure $(K, B, d, i)$ if $(L, \odot, 0, 1)$ and $(K, \oplus, 0, 1)$ coincide and for any $k, l \in L$, $k \ominus k = \bot$.

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 semiring, and vice versa, every structure derived from a semiring is an spm-semiring.

Proof. We first show that every spm-semiring $(K,\circ,\oplus,0,1)$ is derived from some semiring. More specifically, such a semiring can be constructed as follows. Define (i) the semiring $(K,\oplus,\circ,0,1)$; (ii) the boolean algebra $(B,\lor,\land,\neg,T)$, where $B:=\{b|b\in K\land b=0(1\lor 0)\}$ and $b_1\lor b_2:=1(0\cup 1\cup b)$; (iii) the mapping $v:B\rightarrow K$ such that for any $k\in K, v(k)=1(0\cup 1\cup k)$; and (iv) the identity mapping $i:B\rightarrow K$. Letting the function $f$ be a semiring and $v$ the identity mapping $f:B\rightarrow K$, we can verify that the operations $\lor\land\neg$ are well-defined and that together with $B$ they constitute a boolean algebra. Furthermore, the conditions on $d$ and $i$ in the definition of semiring are clearly satisfied to hold. Finally, we show that for any $k,\ell\in K, k\land \ell = k\land v(d(f))$. 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 $T=\{(true, false), \lor, \land, \neg, true\}$ [16]. The semiring $T$ is readily extended to a boolean algebra $T_0$, by incorporating complementation $\neg$. Consider $T_0$, with two elements $d:T\rightarrow T_0$ and $i:T_0\rightarrow T$, of which both are the identity mappings. It is readily verified that $\neg T_0$ is a semiring. The derived spm-semiring consists of $T_0$ with the additional operator $b_1\lor b_2:=b_1\land b_2$. Note that $\neg$ coincides with $\neg semiring (cf. Example 3).

Example 6 (Boolean Semirings). The previous example generalizes to any semiring $K$ that can be extended to a boolean algebra $K_0$. As in the previous example, $(K,\circ,\oplus,0,1)$ is associated with $d$ and $i$ the identity mapping, forms a semiring-like structure. The derived spm-semiring is then given by $K_0\oplus K_0\rightarrow K_0\land K_0$. In other words, $\circ$ coincides with the standard notion of complementation in boolean algebras. For example, consider the relational probability semiring [16] $(P(E),\lor,\land,\neg,B,E)$ for a set $E$. Clearly, $\circ$ 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 $E_1\land \neg E_2$, where $E_n$ denotes the complement of $E_n$.

The previous example shows that $\circ$ in semiring $\circ$ does not always satisfy that $k_1\circ k_2=k_1$ in case that $k_2=0$ and $k_1\oplus k_2=0$. Indeed, consider the sets $A=\{a,b\}$ and $B=\{c,d\}$. Then $A\times B=A$ despite the fact that $B\not=\emptyset$. 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 $\oplus$ is associated with $+=\circ$.

Example 7 (Bags, Fuzzy). Let $B_2=\{(0,1),\lor,\land,\neg,0,1\}$ be the two-element boolean algebra and let $(N,+,\times,0,1)$ be the semiring of natural numbers. Define $d:N\rightarrow \{0,1\}$ as $d(x)=0$ if $x=0$ and $d(x)=1$ if $x\neq 0$. Let $\iota:[0,1]\rightarrow \mathbb{N}$ be the identity mapping. Then clearly, $d(0)=0$, $d(1)=1$, and $\iota(1)=1$. Furthermore, it is readily verified that for any $x,y\in \mathbb{N}$ and $b_1,b_2\in [0,1]$, $d(x\times y)=d(x)\times d(y)$, $d(x\land b)=d(x)\land b$, $\iota(b_1\lor b_2)=\iota(b_1)\lor \iota(b_2)$, and $\iota(b_1\land b_2)\iota(b_1\land b_2)=\iota(b_1)\land \iota(b_2)\iota(b_1\land b_2)$. We also have that $\times d(x)\times d(0)=0$ and that $\circ$ is the identity mapping. Hence,

![Table 1: SPM-semirings and their applications](image)

<table>
<thead>
<tr>
<th>Application</th>
<th>Trust/Set Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(N,\min,\max,\neg,\min,\max,\oplus,\ominus)$</td>
<td>Ranked Trust</td>
</tr>
<tr>
<td>$(P(E),\lor,\land,\neg,\ominus,\oplus)$</td>
<td>Access control</td>
</tr>
<tr>
<td>$({0,1},\max,\min,\neg,\ominus,\oplus)$</td>
<td>Event Tables</td>
</tr>
<tr>
<td>$({0,1},\max,\min,\neg,\ominus,\oplus)$</td>
<td>Bags</td>
</tr>
<tr>
<td>$({0,1},\max,\min,\neg,\ominus,\oplus)$</td>
<td>Fuzzy</td>
</tr>
</tbody>
</table>

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 $\ell$ are $0$. It is readily verified that for such $K$, the derived spm-semiring $P(E),\lor,\land,\neg,B,E)$, where $d(k)=0$ if $k$ and $d(\ell)=0$ if $\ell$ are $0$, and $\iota(0)=0_k$ and $\iota(1)=1_k$. The minus in the derived structure is consequently defined as $k\oplus \ell=k\oplus \iota(d(k))$. Clearly, this restriction to zero-sum free semirings in the previous example is necessary. Indeed, let $(K,\circ,\oplus,0,1)$ be a semiring and assume that $k\oplus \ell\neq 0$. Then also $d(k\oplus \ell)=1=d(\iota(\ell))$ and hence both $d(\iota(\ell))\neq 0$ and $d(\iota(\ell))\neq 1$. We claim that $d(k)=1$ if $k=0$. Suppose, for the sake of contradiction, that $d(k)=1$ if $k\neq 0$. Then, from $0\neq k\oplus \iota(d(k))=0$ we obtain a contradiction. Similarly, for $d(\iota(\ell))\neq 1$.

Example 9 (Probability). Let us reconsider the probability semiring [16] $(P(E),\lor,\land,\neg,B,E)$. Clearly this semiring is zero-sum free and the previous example tells us that a semiring can be extended 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 $\iota(0)=0$ and $\iota(1)=1$. It is readily verified that the minus in the derived spm-semiring satisfies $E_i\oplus E_j=E_i$ when $E_j\not=0$ and $E_i\oplus E_j=\emptyset$ otherwise. We also denote $\circ$ as an example.

Example 10 (Ranked Trust, Access Policy). Consider the tropical semiring $(N_\infty,\min,+,\max,\oplus,\ominus)$ [16], where $N_\infty$ is short for $N\cup\{\infty\}$. 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 trusted, and should be disregarded, have $\infty$ 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 $\min,\max,\oplus,\ominus$ if $n\neq \infty$ and $\min,\max,\ominus$ if $n=\infty$.

Along the same lines, one can extend the semiring $(C,\min,\max,\ominus,\oplus,\ominus)$, with $C=\{P\text{(public)}, C\text{(confidential)}\}, S(\text{secret}), T(\text{top secret}), \emptyset\}$, used in the context of confidentiality policies, to an spm-semiring. Here, the order between the levels of access is specified for $P<C<S<T<0$ and the operators $\min$ and $\max$ are defined relative to this order. It is readily verified that in the resulting spm-semiring, $v\ominus v=0$ if $w$ is 0 and can thus be ignored, and $v\ominus v=0$ 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 \( \langle \mathbb{N}[X],[+,\times,0,1] \rangle \) 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 \( \langle \mathbb{N}[X],[+,\times,0,1] \rangle \). 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 \((K_1,B_1,d_1,t_1)\) and \((K_2,B_2,d_2,t_2)\) be two seba-structures. We say that a mapping \((h_1,h_2)\) between the two seba-structures is a seba-homomorphism if the following conditions are satisfied:

- \(h_1\) is a semiring homomorphism from \(K_1\) to \(K_2\);
- \(h_2 = d_2h_0\), where \(h_0\) is a boolean algebra morphism from \(B_1\) to \(B_2\);
- \(d_2h_0 = h_0d_1\) and \(h_0\cdot t_1 = t_2h_0\).

A seba-structure \((K,B,d,t)\) 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 \((K',B',d',t')\) and any valuation \(\nu: X \rightarrow K'\), we can uniquely extend \(\nu\) to a seba-homomorphism \((h_1,h_2)\) from \((K,B,d,t)\) to \((K',B',d',t')\) such that \(h_1\) coincides with \(\nu\) 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 \(t\) between the semiring and boolean algebra and show that, all combined, these form a seba-structure.

### 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 \(\kappa \in \mathbb{N}^*\) be an integer-valued multi-index and \(\beta \in \{0,1\}^2^n\) be a binary multi-index. For \(i \in \{1, n\}\) and \(j \in \{1, 2n\}\), we denote by \(\kappa(i)\) and \(\beta(j)\) the \(i\)th and \(j\)th entry in \(\kappa\) and \(\beta\), respectively. Given \((\kappa, \beta)\) and a natural number \(a \in \mathbb{N}\), the corresponding extended monomial is the expression

\[a \cdot b_1^{\beta(1)} \cdot b_2^{\beta(2)} \cdots b_n^{\beta(2n)} \cdot x_1^{\kappa(1)} \cdots x_n^{\kappa(n)},\]

which is succinctly denoted by \(a \cdot b^\beta \cdot x^\kappa\). Consequently, an extended polynomial with integer coefficients is an expression of the form

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

where \(a_{\kappa, \beta} \in \mathbb{N}\) 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 \(\mathbb{N}[B,X]\). That is, \(\mathbb{N}[B,X]\) is the subset of \(\mathbb{N}[X]\) in which the exponents of boolean variables are restricted to 0 or 1.

**Example 11.** Let \(X = \{x_1, x_2, x_3\}\) and \(B = \{b_1, b_2, b_3, \bar{b}_1, \bar{b}_2, \bar{b}_3\}\). Consider the indices \((\beta_1, \kappa_1)\), for \(i = 1, 2, 3, 4, \) with \(\beta_1 = (1, 0, 0, 0, 1, 0), \quad \beta_2 = (0, 0, 0, 0, 1, 0), \quad \beta_3 = (1, 0, 0, 0, 0, 0), \quad \beta_4 = (1, 0, 0, 1, 0, 0), \quad \kappa_1 = (3, 0, 0), \quad \kappa_2 = (1, 2, 0), \quad \kappa_3 = (0, 0, 4)\) and \(\kappa_4 = (3, 0, 0, 0, 0)\). Then give the coefficients \(a_{\beta_1, \kappa_1} = 1, a_{\beta_2, \kappa_2} = 2, a_{\beta_3, \kappa_3} = 3\) and \(a_{\beta_4, \kappa_4} = 6\), the corresponding extended polynomial is given by \(b_1 \cdot b_2 x_1^2 + 2b_2 x_2 x_3 + 4b_1 x_2^3 + 6b_1 \cdot b_2 x_1^2 x_2 \).

**Example 12.** Consider the extended polynomial \(b_1 b_2 x_1^3 + 2b_2 x_1 x_2 + 4b_1 x_2^3 + 6b_1 \cdot b_2 x_1\) from Example 11. It is readily verified that the equivalent normal form is given by \(4b_1 b_2 b_3 x_1^3 + 0b_1 b_2 b_3 + b_1 b_2 b_3 x_1 + 0b_1 b_2 b_3 + 0b_1 b_2 b_3 + 0b_1 b_2 b_3\). The support of this polynomial is \(\{1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1\}\).
Consider the quotient structure \( \mathbb{N}_e[B,X]/\sim \) and denote by \( p[B,X]/\sim \) the equivalence class in \( \mathbb{N}_e[B,X]/\sim \) corresponding to the extended polynomial \( p[B,X] \). Define the following operations: 
\[
(p[B,X]/\sim) -\bigcup_{\alpha \in [q[B,X]/\sim]} := (p[B,X] + q[B,X])/\sim
\]
and 
\[
(p[B,X]/\sim) \cdot \bigcup_{\alpha \in [q[B,X]/\sim]} := (p[B,X] \cdot q[B,X])/\sim,
\]
and define the following constants: \( 0_\sim = 0/\sim \) and \( 1_\sim = 1/\sim \).

**Theorem 2.** The seba-structure \( \{\mathbb{N}_e[B,X]/\sim, \mathbb{B}, \mathbb{D}, \tau_e\} \) is universal in the class of all seba-structures, relative to the generator set \( X \).

**Proof.** Let \( (K,B,d,i) \) be a seba-structure consisting of a semiring \( (K,\oplus,\otimes,0,1) \), boolean algebra \( (B,\lor,\land,\top) \) and mappings \( d:K \to B \) and \( i:B \to K \). Let \( X = \{x_1, \ldots, x_n\} \) be a set of variables. Let \( \{x_i/\sim\}_{i=1}^n \) be the embedding of \( X \) in \( \mathbb{N}_e[B,X]/\sim \). Note that \( 0/\sim = 0_e \) and \( 1/\sim = 1_e \). Consider the standard embedding \( \mu:\{X/\sim\} \to K \) defined by \( \mu(0_e) = 0, \mu(1_e) = 1 \) and \( \mu(k/\sim) = k \odot \mu(1_e) \). Let \( \nu:K \to B \) be a valuation. We need to show that there exists a unique seba-morphism \((h_e,h_0)\) from \( \{\mathbb{N}_e[B,X]/\sim, \mathbb{B}, \mathbb{D}, \tau_e\} \) to \((K,B,d,i)\). This is verified by showing that \( \mu \) and \( \nu \) completely specify the value of this morphism on every other element in the structure \( \{\mathbb{N}_e[B,X]/\sim, \mathbb{B}, \mathbb{D}, \tau_e\} \).

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

Let \( (K,\oplus,\otimes,0,1) \) and \( (L,\delta,\gamma,\kappa,0_1,1_1) \) be two spm-semirings. A homomorphism between these structures is a mapping \( h:K \to L \) such that \( h(0_k) = 0_1 \) and \( h(1_k) = 1_1 \).

**Definition 7.** Consider the spm-semiring \( \{\mathbb{N}_e[B,X]/\sim, \mathbb{B}, \mathbb{D}, \tau_e\} \) where 
\[
(p[B,X]/\sim) -\bigcup_{\alpha \in [q[B,X]/\sim]} := (p[B,X]/\sim) \cdot \bigcup_{\alpha \in [q[B,X]/\sim]},
\]
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 \( \{\mathbb{N}_e[B,X]/\sim, \mathbb{B}, \mathbb{D}, \tau_e\} \).

**Proposition 7.** The spm-semiring \( \{\mathbb{N}_e[B,X]/\sim, \mathbb{B}, \mathbb{D}, \tau_e\} \) 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 hom-provenance [16], among other things, of SPARQL query results.
Factorization of SPARQL query evaluation. Let $K$ and $L$ be two spm-semirings. A homomorphism $h: K \rightarrow L$ is said to commute with a SPARQL expression $P$ if for any $K$-annotated RDF graph $G_a$: 

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

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 $(K, \oplus, K, \otimes, K, 0, K, 1, K)$ and $(L, \oplus, L, \otimes, L, 0, L, 1, L)$. The transformation given by $h$ from $G_a$ to $h(G_a)$ 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 $G_a$ 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. Let $(N_{[\mathbb{B}, \mathbb{X}]})_{X/\sim} = \{x_1, x_2, \ldots, x_n\}$ and by $\mathbb{E}$ the unique homomorphism from $(N_{[\mathbb{B}, \mathbb{X}]})_{X/\sim}$ to $K$ that coincides with $\nu$ on $X_{/\sim}$. 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

$$[P]_{G_a} = \mathbb{E}([P]_{(G_a/X)}),$$

where $G_a/X$ is the abstractly tagged version of $G_a$ and $\nu: X_{/\sim} \rightarrow K$ is the function that associates with each equivalence class $x_{/\sim}$ the unique annotation of the triple in $G_a$ tagged with $x_{/\sim}$.

Computing with the universal spm-semiring. We next show how to compute the annotations in $[P]_{(G_a/X)}$. Although the elements in $(N_{[\mathbb{B}, \mathbb{X}]})_{X/\sim}$ 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 $N_{[\mathbb{B}, \mathbb{X}]}$. Clearly, $p[B, X]$ belongs to $p[B, X]_{X/\sim}$ and $q[B, X]$ belongs to $q[B, X]_{X/\sim}$. Algorithm 1 shows how to find a representative in the class corresponding to the addition, multiplication, and difference of $p[B, X]_{X/\sim}$ and $q[B, X]_{X/\sim}$. In a nutshell, the algorithm first computes the normal forms of $p[B, X]$ and $q[B, X]$ and applies the definition $+_{\sim}$, $\times_{\sim}$, and $-_{\sim}$ on these normal forms. A simplification procedure $\mathbb{SIMPLIFY}$ 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 $b_i p[B, X] + b_i^2 p[B, X]$ for some polynomial $p[B, X]$ in which $x_i$ does not appear and such that $\beta(j) = \beta'(j + 1)$ for all $j \in [1, n]$, $i \neq j$, and $\beta(0) = 0$ and $\beta'(0 + 1) = 1$.

**Example 14.** Let $p[B, X] = x_1 x_2 + b_1 x_2$ and $q[B, X] = 3b_2 x_1 + x_2^2$. Then $\mathbb{SIMPLIFY}(p[B, X]) = b_1 b_2 (2x_1^2 + x_2^2) + b_1 b_2 (2x_2^2)$ and $\mathbb{SIMPLIFY}(q[B, X]) = b_1 b_2 (3x_1 + x_2)$.

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 $(p[B, X]_{X/\sim} \oplus q[B, X]_{X/\sim})$ for $\oplus \in \{+, \times_{\sim}, -_{\sim}\}$.  

1. Compute $p_d[\mathbb{B}, \mathbb{X}] = \sum_{b \in \mathbb{B}} b^2 p[B, X]$ of $p[B, X]$;  
2. Compute $q_d[\mathbb{B}, \mathbb{X}] = \sum_{b \in \mathbb{B}} b^2 q[B, X]$ of $q[B, X]$;  
3. If $\oplus = +_{\sim}$ then  
4. return $\mathbb{SIMPLIFY}(\sum_{b \in \mathbb{B}} b p_d[B, X] + q_d[B, X])$;  
5. end if  
6. If $\oplus = \times_{\sim}$ then  
7. return $\mathbb{SIMPLIFY}(\sum_{b \in \mathbb{B}} b p_d[B, X] q_d[B, X])$;  
8. end if  
9. If $\oplus = -_{\sim}$ then  
10. return $\mathbb{SIMPLIFY}(\sum_{b \in \mathbb{B}} b p_d[B, X] q_d[B, X])$, where $S$ is the support of $q_d[\mathbb{B}, \mathbb{X}]$;  
11. end if

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 $\nu: X_{/\sim} \rightarrow K$. We illustrate this for the bag, (set) and fuzzy setting.

**Example 15.** Consider the SPARQL expressions $\Omega_1 \ldots \Omega_4$, $\Omega_5$ and $\Omega_6$ described in Example 1 and shown in Fig. 1. Fig. 3(c),(d),(e) and (f) show $\Omega_{i} \mathcal{C} \mathcal{G}_{a/X}$, for $i = 1, 2, 3, 4$, respectively. Furthermore, $\mathcal{O}_{i} \mathcal{M}_{j} \mathcal{G}_{a/X}$, $\mathcal{O}_{i} \mathcal{M}_{j} \mathcal{G}_{a/X}$, and $\mathcal{O}_{i} \mathcal{M}_{j} \mathcal{G}_{a/X}$ are shown in Fig. 3(g), where $\mathcal{O}_{i} \mathcal{M}_{j}$ (i) and (ii), respectively. Consider the mapping $\nu_{11}$ in $\mathcal{O}_{3}$, its annotation is a representative in $(x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 + x_7^2 + x_8^2 + x_9^2)_{/\sim}$. It is readily verified that Algorithm 1 returns $x_1^2 + x_2^2$. Similarly, $\nu_{12}$ in $\mathcal{O}_{2}$ is annotated by the representative $x_1 x_2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 + x_7^2 + x_8^2 + x_9^2$ as returned by the algorithm. Consider next $\nu_{18}$ in $\mathcal{O}_{1} \mathcal{M}_{1} \mathcal{O}_{1}$. This mapping is annotated by a representative in $(x_1 x_2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 + x_7^2 + x_8^2 + x_9^2)_{/\sim}$ and $-_{\sim}$ this is equal to the normal form of $(x_4 + x_5 + x_6 + x_7 + x_8 + x_9)_{/\sim}$ multiplied (using $\times_{\sim}$) with $f_5$ where $S$ is the complement of the support of the normal form of $x_i$. It is readily verified that $S$ consists of all boolean indices $\beta$ in $BIM$ except those with $\beta(1) = 1$. Hence, the annotation of $\nu_{18}$ is $b_3 (x_4 + x_5)$. □
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 \( X, +, \cdot \) and the monus \( -m \). A study of the properties of \( m \)-semirings can be found in [1]. More specifically, a set of identities \( E_m \) is identified that characterizes \( m \)-semirings. Since the relational difference satisfies two sets of incompatible equivalences, \( E \) and \( E_m \), in the set and bag semantics, respectively, \( E_m \) only considers the common identities in these sets. As a consequence, certain intuitive equivalences of the relational algebra, such as \( R \times (S \setminus T) = (R \times S) \setminus (R \times T) \) (A13 in [1]) are not necessarily satisfied on \( m \)-semirings 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 \( \text{id}_{11} \bot (k_1 \oplus (k_2 \ominus k_3)) = (k_1 \oplus (k_2 \ominus k_3)) \), a stronger identity than \( k_1 \ominus (k_2 \ominus k_3) = (k_1 \ominus (k_2 \ominus k_3)) \) (A13 in [1]) implied by the above equivalence. It is readily shown that \( m \)-semirings do not satisfy this property. Conversely, \( k_1 \ominus (k_2 \ominus k_3) = k_2 \ominus (k_1 \ominus k_3) \) (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 alternative 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 Differencing and Optionals 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 \( \land \) and \( \neg \) 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 \( (m, \delta) \)-semirings which are \( m \)-semirings extended with a duplicate elimination operator \( \delta \), as introduced in [13]. 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, \( (m, \delta) \)-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.

Figure 3: Example of RDF graph and evaluation of SPARQL algebra operators with annotations in the universal spm-semiring.
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 $\delta$, thereby disregarding all $(m, \delta)$-semirings with a more complex $\delta$. 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 RDF rules, used to infer implicit information. However, inference rules do not involve the use of the SPARQL $\text{OPTIONAL}$ 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 explicitly 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 $\text{OPTIONAL}$ (or $\text{DIFFERENCE}$) 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 $\text{OPTIONAL}$ 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 $\text{OPTIONAL}$ 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 expressive-ness 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 $\text{OPTIONAL}$ [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 $\text{NOT EXISTS}$ and $\text{MINUS}$, that differ from the $\text{DIFFERENCE}$ operator considered in this paper. Indeed, $\text{MINUS}$ considers a stronger notion of compatibility of mappings by requiring the presence of common variables; $\text{NOT EXISTS}$ asks for the absence of certain patterns as part of a $\text{FILTER}$ construct. The study of SPARQL query equivalences involving $\text{NOT EXISTS}$ and $\text{MINUS}$, 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