Semantic Web 0 (2014) 0
IOS Press

A logical characterisation of SPARQL federation

Editor(s): Oscar Corcho, Universidad Politécnica de Madrid, Spain
Solicited review(s): Jürgen Umbrich, Vienna University of Economics and Business, Austria; Carlos Buil Aranda, Pontificia Universidad Católica de Chile, Chile; Maria-Esther Vidal, Universidad Simón Bolivar, Venezuela

Audun Stolpe
Norwegian Defence Research Establishment (FFI), Postboks 25, 2027 Kjeller, Norway
E-mail: audun.stolpe@ffi.no

Abstract. The paper provides a logical characterisation of distributed processing of SPARQL queries. The principal notion analysed is that of a distribution scheme; a pair consisting of an evaluation rule and a distribution function. Different choices of evaluation rules and distribution functions give different federation schemes that are proved to be sound and complete wrt. different selections of RDF datasets. Three distribution functions are singled out for special attention, called the even-, standard- and prudent distribution respectively. All yield sound and complete federation schemes when combined with an evaluation rule named the collect-and-combine rule. The completeness results thus obtained are next compared to existing federation systems with the aim of illustrating the potential utility of a framework in which salient properties can be formulated and explored. The final section, relates distribution schemes to the heuristic notion of a join-ordering to yield sound and complete execution plans for the aforementioned federation schemes.

Keywords: SPARQL federation, distributed query processing, logical foundation, completeness, distribution scheme

1. Introduction

That the exploitation and dissemination of Semantic Web data requires powerful federation engines, is a claim that hardly needs an argument. Unedited and decentralized, the Semantic Web is also a singularly fluid computational environment. Therefore any federation engine that aims to be generic must be capable of operating in a dynamic network topology where datasets may be discovered at run time, and where little is known about the structure and content of these datasets prior to querying.

SPARQL federation—the decomposition and distribution of SPARQL queries—is one approach that seems both principled and promising. Its emerging importance is mirrored by the number of publications devoted to this topic in the last five to ten years. However, the wealth of experimental systems and empirical research contrasts sharply with the paucity of foundational work, and attempts to formulate a theoretical framework in which to explore and compare different approaches mathematically are largely absent.

This remains true even when the query language is abstracted away so as to include the comparatively rich literature on distributed databases. Thus, whilst the 150+ references in Kossman's excellent "The State of the Art in Distributed Query Processing" [22] together with the 150+ references in Hose et al.'s "Database foundations for scalable RDF processing" [19] span more than 30 years of research, none of the listed references gives a general logical characterisation of distributed query answering.

The present paper provides some tentative indications that the working out of such a framework may be both an interesting and worthwhile pursuit. Although the concepts introduced ought to be of more general validity, focus will be placed on federation in a zero-knowledge environment, that is, on federation performed in the absence of specific information about

the structure and content of contributing datasets (apart from their signatures—a concept to be defined).

Also, only the fragment of SPARQL that consists of simple conjunctive queries aka. basic graph patterns will be considered. The operative assumption is that these restrictions form a natural base-line for a theory, from which more complex cases can be derived.

The present paper does not advocate 'an approach' to SPARQL federation. Rather, it is more aptly seen as giving expression to properties and observations that are already part of the 'folklore' of the Semantic Web community, and in the process developing a formal apparatus in which the relevant distinctions can be drawn and their ramifications explored.

Indeed, the results that are to be presented are not even essentially tied to SPARQL, but could easily be generalized to apply to any class of conjunctive queries [1, chap. 4] expressible in first-order logic. Thus, the relationship between the present theory and the SPARQL 1.1. federation extension [11,30] is essentially one of abstraction: the latter, by providing the `SERVICE` keyword for source selection, provides one way of implementing the former.

The paper is organized as follows: After a review of related work in section 2, section 3 recalls a few essential notions of SPARQL semantics, fixes notation and declares theoretical assumptions for subsequent developments. Section 4 introduces the principal unit of analysis, which is that of a *federation scheme* defined as pair consisting of a *distribution function* and an *evaluation* rule. A number of concrete federation schemes are presented, obtained by combining a particularly natural evaluation rule, called collect-and-combine-rule, with three different distribution functions, called the *even-*, *standard-* and *prudent distribution* respectively. Section 5 investigates the notions of soundness and completeness as they apply to federation schemes, where soundness/completeness is measured wrt. families (i.e. sets) of sets of RDF graphs. Intuitively different families represent different admissible ways of selecting RDF sources for federation, whence they will also be called, more suggestively, sets of *selections* of RDF datasets. The investigation is restricted to soundness and completeness in a zero-knowledge environment—that is, to the case where the distribution of data over the sources in a selection is unconstrained. This corresponds to the case where *all* sets of RDF graphs constitute admissible selections. Next, section 6 compares the completeness results of section 5 with some examples of existing federation systems, with the principal aim of illustrating the potential utility of having a framework in which salient properties can be formulated and explored. Finally, since no federation engine can be expected to perform well without a carefully crafted optimizer, the paper is rounded off in section 7 by showing how the notion of a join-ordering fits together with that of a federation scheme. This section can be considered a corollary to the preceding ones.

2. Related work

Fairly recent surveys of the state of the art wrt. RDF federation can be found in Betz et al. [8], Görlitz and Staab [14] and Hose et al. [19]. Based on these sources, it is possible to distinguish between at least three major trends in RDF federation, namely lookup-based federation, warehousing and distributed query processing. Lookup-based federation, which may also be called federation-by-traversal (or follow-your-nose traversal, as in [4]), iteratively downloads and evaluates data based on links (usually as prescribed by the Linked Data principles [9]) from one dataset into another. The answer to a query is composed by cumulatively adding answers from incoming data during the traversal process. This approach is exemplified by e.g. [17,18] and [23].

Approaches based on warehousing, on the other hand, collect all data in advance and combines it into a central triple store against which queries are evaluated. Recent efforts in this direction focus on the use of cluster technology such as MapReduce and hadoop, e.g. [12,20,21] and [29].

Finally, approaches based on distributed query processing rely on analysing a query to identify a set of relevant RDF graphs—with or without the aid of statistics—to which subqueries can be assigned. Like federation-by-traversal and unlike warehousing, queries are evaluated remotely. Recent example of this approach include [3,7,27,31] and [34].

I should be said that this classification is very rough, cutting across important and interesting topics such as adaptivity and opportunism, indexing, join-order optimization, statistics-generation and more. The reader is referred to the aforementioned surveys for more detail.

The present paper is most comfortably subsumed under the category of distributed query processing, although it ought to be of relevance to the other two as well. To the author's knowledge it is the first foundational study of distributed query processing that ap-

proaches the discipline from a purely logical point of view. There do exist a few formal studies of what can be considered the more general topic of querying the Web, for instance Abiteboul and Vianu [1] and Bouquet et al. [10]. The former is concerned with the computability of queries over the Web expressed in Datalog, and predates the SPARQL query language. The latter describes three different ways in which a query can be answered and characterizes their answers semantically in complete abstraction from query languages and distribution algorithms. These references have little in common with the present study, apart from being formal and about queries on the Semantic Web.

3. Preliminaries

In order to lighten the notation somewhat, curly braces will be omitted from singletons in set-theoretic expressions and applications of functions if no confusion is likely to ensue, e.g. $P \cup t$ instead of $P \cup \{t\}$ and $f(t)$ instead of $f(\{t\})$. Also, when f is a function and A a subset of f's domain, then $f(A)$ will be used as a shorthand for the set of elements b such that $b = f(a)$ for some $a \in A$.

Turning now to RDF specifics, let I, B and L denote pairwise disjoint infinite sets of IRIs, blank nodes, and literals respectively. An RDF triple is an element $(s, p, o) \in (I \cup B) \times I \times (I \cup B \cup L)$ where s is the *subject*, p the *predicate* and o the *object*. In conformity with the nomenclature of [5], IL abbreviates $I \cup L$ and T abbreviates $I \cup B \cup L$. These latter sets will be referred to collectively as *RDF terms*.

Definition 3.1 *An RDF graph is a finite set of RDF triples.*

An RDF graph will sometimes also be referred to as an *RDF dataset*, or simply a *dataset* [5] or even a *source*. These are all interchangeable, but will tend to be used in different contexts for the purpose of making the terminology more suggestive. The notation G_i will be used to denote an RDF graph. The subscript may be omitted when idle.

Let V be an infinite set of variables. Variables will be denoted by capital letters prefixed by question marks ?X, ?Y, ?Z and so on.

Definition 3.2 *A basic graph pattern (BGP), aka. a conjunctive graph pattern, is defined as follows:*

1. *Any* triple pattern *in* $(IL \cup V) \times (I \cup V) \times (IL \cup V)$ *is a BGP.*
2. *if* P_1 *and* P_2 *are BGPs, then so is* $P_1 \cup P_2$.

The notation t_i is used to denote triple patterns. Again, subscripts may be omitted when they serve no purpose. In order to avoid tedious limiting cases in proofs, it will be assumed that the set of triple patterns is disjoint from the set of RDF triples, that is, a triple pattern is assumed to contain at least one variable.

As in [5] conjunctive graph patterns are conceived of as queries matching RDF graphs. Answers are eligible substitutions of values from the RDF graph in question for variables in the graph pattern in question. In conformity to [5], sets of answers will be formalized in terms of partial functions $\mu : V \rightarrow T$. The domain of μ is the subset of V where μ is defined. Two mappings μ_1 and μ_2 are compatible when for all $?X \in dom(\mu_1) \cap dom(\mu_2)$ it is the case that $\mu_1(?X) = \mu_2(?X)$. Thus two mappings with disjoint domains are always compatible. Let Ω_1 and Ω_2 be sets of mappings. The join of Ω_1 and Ω_2 is denoted $\Omega_1 \bowtie \Omega_2$ and is defined as the set of compatible pairs in $\Omega_1 \times \Omega_2$.

The set of answers to a BGP P over a graph G is denoted $[\![P]\!]_G$ and is defined as follows:

Definition 3.3 *Let* G *be an RDF dataset, and* P *a BGP. Let* $\mu(t)$ *denote the triple obtained by replacing the variables in* t *according to* μ*, and let* $var(t)$ *denote the set of variables occurring in* t*. Then*

1. $[\![P]\!]_G =_{df} \{\mu \mid dom(\mu) = var(t)$ *and* $\mu(t) \in G\}$*, if* P *is a triple pattern* t*, or*
2. $[\![P_1]\!]_G \bowtie [\![P_2]\!]_G$*, if* $P = P_1 \cup P_2$.

Taking stock, the reader should note two things: first, as announced in the introduction, the present theory only considers the fragment of SPARQL that consists of conjunctive queries. Secondly, the distinction between the syntax and the semantics of SPARQL queries has been blurred, and this is deliberate. Usually, one would treat a SPARQL graph pattern as a syntactic object, and the query mapping that links it to an RDF dataset as a semantic object. However, defining a basic graph pattern as a *set* of triple patterns, as in definition 3.2, is mathematically convenient, and it simplifies formal developments without loss of precision. Abiteboul et al. [2] established this precedent in database theory.

Now, considered from an abstract or logical point of view, the problem of characterizing SPARQL federation is essentially that of generalizing definition 3.3 to the case that G is a finite *set* of RDF graphs—sets of RDF graphs will henceforth be denoted $\mathscr{G}$, possibly with subscripts. This generalization will proceed along the following two lines:

Definition 3.4 *Let $\mathscr{G}$ be a set of RDF datasets and P a BGP. Then:*

1. $[\![P]\!]_{\mathscr{G}} =_{df} [\![P]\!]_{\bigcup \mathscr{G}}$
2. $[P]_{\mathscr{G}} =_{df} \bigcup_{G \in \mathscr{G}} [\![P]\!]_G$

Here $\bigcup \mathscr{G}$ denotes the union of all the elements of $\mathscr{G}$. It will be assumed that blank nodes are never shared between datasets. This is no real restriction since one can always rename blank nodes without altering the information content of the datasets in question.

4. Federation schemes

Presupposing a selection of RDF datasets $\mathscr{G}$, the problem of evaluating a query Q reduces to two interrelated sub-problems: 1) 1) how to assign to members of $\mathscr{G}$ the proper subquery of Q, and B) how to combine the answers to these subqueries into an answer to Q itself. Varying either of these parameters, including the selection of sources, may change the final answer. That is, the outcome of the federation process will in general be sensitive to how the contributing datasets are selected, how subqueries of the global query are allocated to these datasets and how the partial results are subsequently combined.

Therefore any *general* property of distributed query processing must be similarly parametrised, that is, in terms of a *decomposition- or distribution function*, an *evaluation rule* and some notion capturing the permissible ways of selecting contributing datasets.

The last of these concepts will be discussed in more detail in the next section. As regards the former two, the general idea expounded in the following is to distribute a (global) query over a selection of datasets by decomposing it into subqueries that can be evaluated directly against the sources that cover its *signature*. Using signatures to route subqueries is a common stratagem in the literature (cf. [13] and [34]).

Definition 4.1 *Let S be a BGP or a set of RDF triples. The signature of S written $sig(S)$ is defined as*

$$sig(S) =_{df} S^2 \setminus (V \cup B)$$

where S^2 denotes the projection of S onto its second coordinates.

Note that variables and blank nodes do not add to the signature of a graph pattern, and that blank nodes do not add to the signature of an RDF graph.

Definition 4.2 *If P is non-empty, then*

$$\Sigma_{\mathscr{G}}(P) =_{df} \{G \in \mathscr{G} : sig(P) \subseteq sig(G)\}$$

Otherwise $\Sigma_{\mathscr{G}}(P) =_{df} \emptyset$.

It is a simple consequence of definitions 4.1 and 4.2 that if $t^2 \in V \cup B$ for some triple pattern t, that is, if t has a variable or blank node in predicate position, then $\Sigma_{\mathscr{G}}(t) = \mathscr{G}$ for any $\mathscr{G}$. A consequence of this is that a triple pattern that has a blank node or variable in predicate position will be routed to all datasets in a selection.

Definition 4.3 (Distribution function) *A distribution function is a binary function δ that takes a set of RDF datasets $\mathscr{G}$ and a BGP P and returns a set of pairs $(P_i, \mathscr{G}_j)$ such $P_i \neq \emptyset$ and such that both of the following hold:*

1. $\mathscr{G}_j \subseteq \Sigma_{\mathscr{G}}(P_i)$
2. *if* $(P_1, \mathscr{G}_i), (P_2, \mathscr{G}_j) \in \delta(\mathscr{G}, P)$ *and* $P_1 = P_2$ *then* $\mathscr{G}_i = \mathscr{G}_j$
3. $\bigcup_{(P_i, \mathscr{G}_j) \in \delta(\mathscr{G}, P)} P_i = P$

Clause (1) prevents a distribution function from behaving erratically when it comes to assigning subqueries to datasets. That is, a BGP is only assigned to datasets that can potentially answer it. Clause (2) ensures that the value of a distribution function on any selection of datasets $\mathscr{G}$ and any BGP P is itself a function, that is, that every subquery of P is assigned to exactly one subset of $\mathscr{G}$. Finally, clause (3) expresses a necessary condition for soundness: δ distributes P over $\mathscr{G}$ only if P is decomposed into sub-queries that jointly exhaust P. Note that δ as so defined is a partial function, that is, (1) and (3) can not be satisfied for arbitrary choices of $\mathscr{G}$ and P, reflecting the fact that the accumulated signature of the datasets may not cover the signature of the global query. If it does then δ will henceforth be said to be defined *at* $\mathscr{G}$ and P.

A distribution function will be said to be *egalitarian* if it satisfies the converse of definition 4.3(2):

Definition 4.4 (Egalitarian distribution) *A distribution function is* egalitarian *if $(P_i, \mathscr{G}_j) \in \delta(\mathscr{G}, P)$ implies $\Sigma_{\mathscr{G}}(P_i) \subseteq \mathscr{G}_j$.*

Egalitarianism has an important bearing on zero-knowledge completeness. It is therefore investigated in more detail in section 5. Several of the distribution functions introduced in the next section are egalitarian. If δ is egalitarian, then condition 4.3(2) is redundant.

4.1. Examples of distribution functions

Let the notion of an *exclusive subset* of P pertaining to a given dataset G relative to a selection of datasets $\mathscr{G}$ be defined as follows:

Definition 4.5 *The subpattern of BGP P that is exclusive to a dataset G, relative to a set of RDF datasets $\mathscr{G}$, is the set:*

$$E_G(P) =_{df} \{t \in P : \Sigma_{\mathscr{G}}(t) = \{G\}\}$$

A triple pattern that is not an element of an exclusive group is called a non-exclusive triple pattern relative to the same selection of datasets.

In this form, definition 4.5 goes back to [34].

The following theorem gives examples of egalitarian distribution functions:

Theorem 4.1 *Let $\mathscr{G}$ be any set of RDF datasets and P any BGP. Stipulate that $(P_i, \mathscr{G}_i) \in \delta(\mathscr{G}, P)$ iff $\mathscr{G}_i = \Sigma_{\mathscr{G}}(P_i)$ and one of the following conditions hold:*

1. *Even distribution: P_i is a singleton $t \in P$.*
2. *Standard distribution: either*
   a) *P_i is a non-exclusive singleton $t \in P$, or*
   b) *$P_i = E_G(P)$ for some $G \in \mathscr{G}$*
3. *Prudent distribution: either*
   a) *P_i is a non-exclusive singleton $t \in P$, or*
   b) *P_i is a maximal join-connected subset of $E_G(P)$ for some $G \in \mathscr{G}$*

Then δ is a distribution function in the sense of definition 4.3.

Proof The proof considers only the standard distribution. The other cases are similar. So suppose $(P_i, \mathscr{G}_i) \in \delta(\mathscr{G}, P)$ iff $\mathscr{G}_i = \Sigma_{\mathscr{G}}(P_i)$. If the signature of P is not covered by $\mathscr{G}$, then δ is vacuously a distribution function. Suppose the opposite. Since $\mathscr{G}_i = \Sigma_{\mathscr{G}}(P_i)$ for all $(P_i, \mathscr{G}_i) \in \delta(\mathscr{G}, P)$, definition 4.3(1) is satisfied and definition 4.3(2) is entailed. For 4.3(3) it suffices to note that all exclusive and non-exclusive patterns, which together exhaust P, are assigned to some dataset in $\mathscr{G}$, by condition 2(a) and 2(b).

Here, the join-connectedness of a BGP P means:

Definition 4.6 *Let $P := \{t_i\}_{i \in I}$ be a BGP. Then the join-graph induced by P is the set of pairs (t_i, t_j) such that $var(t_i) \cap var(t_j) \neq \emptyset$. A subset of P is join-connected if it induces a connected join-graph.*

Stated in words, a BGP being join-connected means that it cannot be split into sub-patterns that do not share a variable.

Anticipating subsequents developments, the even-, standard- and prudent distribution are all suitable for a zero-knowledge environment insofar as they yield sound and complete federation schemes (wrt. the set of all selections of RDF graphs) when paired with the collect-and-combine evaluation rule that will be introduced shortly.

Moreover, all three distribution functions can be said to heed the fact that one dataset may be able to generate new results when joining with *partial* results from another [31, p. 120]. Stated differently, a join may span two or more datasets, in which case the evaluation of a non-singleton graph pattern over a given dataset may come to suppress intermediate results that are needed for cross-site joins: in the simplest case, the answer to a pair of triple patterns $P := \{t_1, t_2\}$ over a source G_1 may not contain every triple g_1 that matches t_1. Since, there could be a triple pattern g_2 in a different source G_2 that matches t_2, there may very well be an answer to P in the union of G_1 and G_2. Yet, this answer may happen to escape the distributed query answering process, since the distribution does not guarantee intermediate results from G_1 and G_2 respectively that when combined yield the cross-site join $g_1 \cup g_2$.

The even distribution averts this risk by breaking a graph pattern into its singleton constituents, but this may be considered to be overly cautious. The standard distribution, in contrast, carves a query at its natural joints in the sense that whenever the pattern in question is exclusive to a dataset, then it is assigned to that dataset as-is. This is safe too, since grouping exclusive patterns cannot incur a loss of information [13,34]. Finally, the prudent distribution, which does not seem to have been mentioned in the literature yet, is halfway between the other two. Like the even distribution it treats non-exclusive triples as singletons, and like the standard distribution it groups together (some) exclusive triples. However, unlike the standard distribution, the prudent distribution splits exclusive groups into maximal join-connected components, each of which is assigned to the dataset for which it is exclusive.

```
SELECT ?drug ?keggUrl ?chebiImage WHERE {
   ?drug rdf:type drugbank:drugs .
   ?drug drugbank:keggCompoundId ?keggDrug .
   ?drug drugbank:genericName ?drugBankName .
   ?keggDrug bio2rdf:url ?keggUrl .
   ?chebiDrug purl:title ?drugBankName .
   ?chebiDrug bio2rdf:image ?chebiImage .
}
```

Listing 1: Query LS5 from FedBench

Example 4.1 *Consider the query in listing 1. This is query LS5 from the FedBench suite [33] which asks for all drugs from Drugbank, together with the URL of the corresponding page stored in KEGG and the URL to the image derived from ChEBI.*

The signature of the query divides among the FedBench datasets as specified in table 1. Presupposing this division, table 2 gives the even, standard and prudent distributions respectively. Blocks in a column correspond to subqueries and are labelled with the datasets to which that subquery is assigned. For instance, the standard distribution assigns the subquery consisting of line 2 and 3 to KEGG, since 2 and 3 form and exclusive group for it. The non-exclusive triple pattern in line 4, in contrast, is assigned to both KEGG and ChEBI.

Table 1
Distribution of the signature of query LS5.

Signature element/property	Endpoint
`rdf:type`	All
`drugbank:keggCompundId`	Drugbank
`drugbank:genericName`	Drugbank
`bio2rdf:url`	KEGG, ChEBI
`purl:title`	KEGG, ChEBI
`bio2rdf:image`	ChEBI

The standard- and prudent distributions of LS5 over KEGG, ChEBI and Drugbank are identical for this particular selection of datasets. The difference between them is revealed by reducing the selection to ChEBI and Drugbank only, viz. table 3. Here, the standard distribution assigns the subquery consisting of line 4,5 and 6 to ChEBI, since this larger set now constitutes an an exclusive group for ChEBI. Note, however, that its join-graph is not connected since none of the variables in line 4 occur in line 5 or 6. Therefore, the prudent distribution divides $\{4, 5, 6\}$ *into its two join-connected components* $\{4\}$ *and* $\{5, 6\}$ *which are both assigned to ChEBI, only this time as separate subqueries.*

The relative merits of these distribution schemes is not really visible unless looked at from the perspective of join-order heuristics—a topic which is otherwise postponed until section 7. If a join-ordering is perceived as a partial order over the power set of a graph pattern P then the even distribution is the finest ordering of P. In computational terms, the even distribution makes each triple pattern in P executable independently of every other, and therefore leaves maximum control to query planning on the side of the federator. The standard distribution, on the other hand, strikes a balance between query optimization on the side of the federator and on the side of the contributing datasets. That is, exclusive patterns effectively 'bracket' joins, thus suspending optimization and delegating it to the RDF source in question. The prudent distribution, being halfway between the other two, is more discerning than the even distribution, but, one might say, less dogmatic about the heuristic value of exclusive subpatterns than the standard distribution. Specifically, if an exclusive subpattern can be split into several components without severing joins, then the prudent distribution hands responsibility for preferring between these components back to the federator in order, possibly, to avoid requesting cartesian products from the remote RDF source.

Turning now to the notion of an evaluation rule, the second proposed component of federation schemes, it suffices for now to define it in complete abstraction simply as a function that produces a set of answers from a distribution:

Definition 4.7 (Evaluation rule) *An evaluation rule is a function* $\mathbb{E}$ *that takes any distribution* $\delta(\mathscr{G}, P)$ *and returns a set of variable mappings* $\mu : V \to T$.

Obviously, this definition hides great variety. Next:

Definition 4.8 *A federation scheme is a pair* $(\mathbb{E}, \delta)$ *consisting of an evaluation rule and a distribution function.*

The central claim of the present paper is that a federation scheme is the proper unit of analysis for distributed query processing—or at any rate, for the purposes of logical analysis. That means that meta-properties such as soundness and completeness are relations that hold between federation schemes on the

Table 2
Distribution of LS5 over KEGG, ChEBI and Drugbank

Nr.	Triple pattern	even	standard	prudent
1	?drug rdf:type drugbank:drugs	All	All	All
2	?drug drugbank:keggCompoundId ?keggDrug	Drugbank		
3	?drug drugbank:genericName ?drugBankName	Drugbank	Drugbank	Drugbank
4	?keggDrug bio2rdf:url ?keggUrl	KEGG, ChEBI	KEGG, ChEBI	KEGG, ChEBI
5	?chebiDrug purl:title ?drugBankName	KEGG, ChEBI	KEGG, ChEBI	KEGG, ChEBI
6	?chebiDrug bio2rdf:image ?chebiImage	ChEBI	ChEBI	ChEBI

Table 3
Distributions of LS5 over ChEBI and Drugbank

Nr.	Triple pattern	even	standard	prudent
1	?drug rdf:type drugbank:drugs	Both	Both	Both
2	?drug drugbank:keggCompoundId ?keggDrug	Drugbank		
3	?drug drugbank:genericName ?drugBankName	Drugbank	Drugbank	Drugbank
4	?keggDrug bio2rdf:url ?keggUrl	ChEBI		ChEBI
5	?chebiDrug purl:title ?drugBankName	ChEBI		
6	?chebiDrug bio2rdf:image ?chebiImage	ChEBI	ChEBI	ChEBI

one hand and sets of selections of RDF datasets on the other. The former specifies how a graph pattern is to be distributed and evaluated, the latter demarcates the permissible ways of selecting sources over which to distribute the decomposed query:

Definition 4.9 *Let $\{\mathscr{G}_i\}_{i\in I}$ be a set of selections of RDF graphs. A federation scheme $(\mathbb{E}, \delta)$ is* sound *wrt. $\{\mathscr{G}_i\}_{i\in I}$ if $\mathbb{E}(\delta(\mathscr{G}_i, P)) \subseteq [\![P]\!]_{\mathscr{G}_i}$ for any BGP P and any $i \in I$. It is complete wrt. $\{\mathscr{G}_i\}_{i\in I}$ if the converse inclusion holds.*

Bordering on circularity, one might say that a set of selections of RDF datasets represents a *way* of choosing RDF datasets that keeps a federation scheme correct and exhaustive. The role of soundness and completeness theorems is to act as chopsticks (borrowing a metaphor from Makinson [26]) to pin down a federation scheme and a set of selections that is perfectly matched in the sense that the federation scheme neither invents nor ignores answers when restricted to that set.

Table 4
Summary of notation

Nomenclature	description
t_i	a triple pattern
P_i	a conjunctive graph pattern
G_i	an RDF dataset
$\mathscr{G}$	a set of RDF datasets
$var(P)$	the variables occurring in P
μ	partial function from V to T
$dom(\mu)$	the domain of μ
$[\![P]\!]_G$	P evaluated over G
$[\![P]\!]_{\mathscr{G}}$	P evaluated over the union of $\mathscr{G}$
$[P]_{\mathscr{G}}$	the union of evaluating P over *each* $G \in \mathscr{G}$
$sig(P)$	the signature/the set of predicates of P
$\Sigma_{\mathscr{G}}(P)$	the datasets in $\mathscr{G}$ whose signature cover P

5. Completeness in the zero-knowledge case

As mentioned in section 3, the problem of giving a logical characterisation of SPARQL federation can be construed as the problem of generalizing the semantics of conjunctive queries to multiple RDF graphs. Standard SPARQL semantics defines the evaluation $[\![P]\!]_G$ of a BGP P over an RDF graph G in terms of partial functions or maps from variables to RDF terms. The conjunction of different sets of answers $[\![P_1]\!]_{G_1} \bowtie [\![P_2]\!]_{G_2}$ is defined in terms of the union of pairs of compatible maps, where compatibility means agreement on shared variables. However, the standard semantics requires that a) G_1 and G_2 are singletons and that b) $G_1 = G_2$.

The distributed query processing semantics that is developed in the present paper sticks to the general idiom of the standard semantics but relaxes restric-

tions a) and b). Considered in the abstract, this involves working out an account of combinations

$$f(P_1)_{\mathscr{G}_i} \mathbin{\overline{\wedge}} g(P_2)_{\mathscr{G}_j} \tag{1}$$

where f and g is either of the evaluation functions $[\cdot]$ and $[\![\cdot]\!]$ (not necessarily different) and the operator $\overline{\wedge}$ is either $\cup$ or $\bowtie$. Given the present emphasis on completeness, not all such combinations are of interest, only those for which

$$f(P_1)_{\mathscr{G}_i} \mathbin{\overline{\wedge}} g(P_2)_{\mathscr{G}_j} = [\![P_1 \cup P_2]\!]_{\mathscr{G}_j \cup \mathscr{G}_j}$$

—an equation that places severe restrictions on eligible pairs of distribution functions and evaluation rules.

This section focuses on one particularly natural set of such pairs, namely those formed from distributions functions that are egalitarian and satisfy a condition called *granularity* combined with the announced collect-and-combine evaluation rule that will be defined shortly.

The set of federation schemes that are sound and complete in a zero-knowledge environment—that is, sound and complete wrt. to the set of all selections of RDF datasets—is larger then this, though, and in fact does not presuppose neither granularity nor egalitarianism. Examples of non-granular and non-egalitarian, but zero-knowledge sound and complete, federation schemes are provided in subsection 5.2.

The account is built up from simple principles, starting with the following lemma:

Lemma 5.1 (Dataset monotony) $[\![P_1]\!]_{\mathscr{G}_i} \subseteq [\![P_2]\!]_{\mathscr{G}_j}$ *whenever* $\mathscr{G}_i \subseteq \mathscr{G}_j$.

Proof Immediate from the assumption that no RDF graphs share blank nodes.

Although dataset monotony does require the non-trivial property of disjointness for every pair of elements in a set of RDF graphs wrt. blank nodes, this is not a real restriction since renaming of blank nodes does not change the semantics of an RDF graph. Hence, one can always assume that blank nodes have been standardized apart in a pre-processing step.

The next lemma gives the case of equation (1) for $f = g = [\![\cdot]\!]$ and $\overline{\wedge} = \bowtie$:

Lemma 5.2 *Let* $\Sigma_{\mathscr{G}}(t_1) = \mathscr{G}_i$ *and* $\Sigma_{\mathscr{G}}(t_2) = \mathscr{G}_j$ *for some set of RDF graphs* $\mathscr{G}$. *Then* $[\![t_1 \cup t_2]\!]_{(\mathscr{G}_i \cup \mathscr{G}_j)} = [\![t_1]\!]_{\mathscr{G}_i} \bowtie [\![t_2]\!]_{\mathscr{G}_j}$.

Proof For the left-to-right direction suppose $\mu \in [\![t_1 \cup t_2]\!]_{(\mathscr{G}_i \cup \mathscr{G}_j)}$. Then $\mu \in [\![t_1]\!]_{(\mathscr{G}_i \cup \mathscr{G}_j)} \bowtie [\![t_2]\!]_{(\mathscr{G}_i \cup \mathscr{G}_j)}$, whence $\mu = \mu_1 \cup \mu_2$ for $\mu_1 \in [\![t_1]\!]_{(\mathscr{G}_i \cup \mathscr{G}_j)}$ and $\mu_2 \in [\![t_2]\!]_{(\mathscr{G}_i \cup \mathscr{G}_j)}$. Since t_1 and t_2 are triple patterns, and μ_1 and μ_2 are functional, there is exactly one triple $g_1 \in G_m$ s.t. $\mu_1(t_1) = g_1$ and exactly one triple $g_2 \in G_n$ s.t. $\mu_2(t_2) = g_2$ for some $G_m, G_n \in \mathscr{G}_i \cup \mathscr{G}_j$. If t_1^2 is a blank node then $sig(t_1) = \emptyset$ otherwise $sig(t_1) = sig(g_1)$. Both cases entail $sig(t_1) \subseteq sig(G_m)$. A similar argument establishes $sig(t_2) \subseteq sig(G_n)$. Therefore, since by the supposition of the lemma $\Sigma_{\mathscr{G}}(t_1) = \mathscr{G}_i$ and $\Sigma_{\mathscr{G}}(t_2) = \mathscr{G}_j$ it follows that $G_m \in \mathscr{G}_i$ and $G_n \in \mathscr{G}_j$. Hence $\mu_1 \in [\![t_1]\!]_{\mathscr{G}_i}$ and $\mu_2 \in [\![t_2]\!]_{\mathscr{G}_j}$, so $\mu \in [\![t_1]\!]_{\mathscr{G}_i} \bowtie [\![t_2]\!]_{\mathscr{G}_j}$ as desired. The converse direction follows immediately from dataset monotony (lemma 5.1).

The restriction on the families of datasets $\mathscr{G}_i$ and $\mathscr{G}_j$ is essential. That is, lemma 5.2 does not hold for arbitrary pairs $(t_a, \mathscr{G}_k)$ and $(t_b, \mathscr{G}_l)$, which is as one would expect.

As regards the interplay between $[\cdot]$ and $[\![\cdot]\!]$, the following property holds in general:

Lemma 5.3 $[P_1]_{\mathscr{G}_i} \bowtie [P_2]_{\mathscr{G}_j} \subseteq [\![P_1 \cup P_2]\!]_{(\mathscr{G}_i \cup \mathscr{G}_j)}$

Proof Suppose $[P_1]_{\mathscr{G}_i} \bowtie [P_2]_{\mathscr{G}_j}$. Then there are $\mu_1 \in [P_1]_{\mathscr{G}_i}$ and $\mu_2 \in [P_2]_{\mathscr{G}_j}$ s.t. $\mu = \mu_1 \cup \mu_2$. By definition 3.4 (2) there are $G_m \in \mathscr{G}_i$ and $G_n \in \mathscr{G}_j$ such that $\mu_1 \in [\![P_1]\!]_{G_m}$ and $\mu_2 \in [\![P_2]\!]_{G_n}$. Thus, applying lemma 5.1 gives $\mu_1 \in [\![P_1]\!]_{(\mathscr{G}_i \cup \mathscr{G}_j)}$ and $\mu_2 \in [\![P_2]\!]_{(\mathscr{G}_i \cup \mathscr{G}_j)}$, whence $\mu \in [\![P_1 \cup P_2]\!]_{(\mathscr{G}_i \cup \mathscr{G}_j)}$ as desired.

The converse does not, but there is the following qualified version of it:

Lemma 5.4 *Put* $\Sigma_{\mathscr{G}}(P_1) = \mathscr{G}_i$ *and* $\Sigma_{\mathscr{G}}(P_2) = \mathscr{G}_j$ *for some set of selections of RDF datasets* $\mathscr{G}$ *and suppose* $[\![P_1]\!]_{\mathscr{G}_i} = [P_1]_{\mathscr{G}_i}$ *and* $[\![P_2]\!]_{\mathscr{G}_j} = [P_2]_{\mathscr{G}_j}$. *Then* $[\![P_1 \cup P_2]\!]_{(\mathscr{G}_i \cup \mathscr{G}_j)} \subseteq [P_1]_{\mathscr{G}_i} \bowtie [P_2]_{\mathscr{G}_j}$.

Proof By definition 3.4(1), $[\![P_1 \cup P_2]\!]_{(\mathscr{G}_i \cup \mathscr{G}_j)} = [\![P_1]\!]_{(\mathscr{G}_i \cup \mathscr{G}_j)} \bowtie [\![P_2]\!]_{(\mathscr{G}_i \cup \mathscr{G}_j)}$. By the supposition of the lemma $\Sigma_{\mathscr{G}}(P_1) = \mathscr{G}_i$ and $\Sigma_{\mathscr{G}}(P_2) = \mathscr{G}_j$ so $[\![P_1]\!]_{(\mathscr{G}_i \cup \mathscr{G}_j)} \bowtie [\![P_2]\!]_{(\mathscr{G}_i \cup \mathscr{G}_j)} = [\![P_1]\!]_{\mathscr{G}_i} \bowtie [\![P_2]\!]_{\mathscr{G}_j}$. Since $[\![P_1]\!]_{\mathscr{G}_i} = [P_1]_{\mathscr{G}_i}$ and $[\![P_2]\!]_{\mathscr{G}_j} = [P_2]_{\mathscr{G}_j}$, it follows that $[\![P_1 \cup P_2]\!]_{(\mathscr{G}_i \cup \mathscr{G}_j)} = [\![P_1]\!]_{\mathscr{G}_i} \bowtie [\![P_2]\!]_{\mathscr{G}_j} = [P_1]_{\mathscr{G}_i} \bowtie [P_2]_{\mathscr{G}_j}$, so $[\![P_1 \cup P_2]\!]_{(\mathscr{G}_i \cup \mathscr{G}_j)} = [P_1]_{\mathscr{G}_i} \bowtie [P_2]_{\mathscr{G}_j}$ which is clearly sufficient.

A distribution function that satisfies both directions will be said to be *agnostic*:

Definition 5.1 (Agnosticism) *A distribution function δ is agnostic if for every set of selections of RDF datasets $\mathscr{G}$, every BGP P, and every $(P_1, \mathscr{G}_i), (P_2, \mathscr{G}_j) \in \delta(\mathscr{G}, P)$ it follows that $[P_1]_{\mathscr{G}_i} \bowtie [P_2]_{\mathscr{G}_j} = [\![P_1 \cup P_2]\!]_{(\mathscr{G}_i \cup \mathscr{G}_j)}$*

Example 5.1 *As an example of a distribution function that is not agnostic, consider the trivial distribution defined by putting $\delta(\mathscr{G}, P) = \{(P, \mathscr{G})\}$. Since $[P]_{\mathscr{G}}$ does not in general coincide with $[\![P]\!]_{\mathscr{G}}$, δ is not agnostic.*

The significance of the property of agnosticism consists in the fact that for agnostic distribution functions there exists a natural and simple evaluation rule that yields a sound and complete federation scheme for the zero-knowledge case. This is the announced collect-and-combine rule:

Definition 5.2 (Collect-and-combine rule) *Put $\Delta := \delta(\mathscr{G}, P)$. If δ is not defined at $\mathscr{G}$ and P we put $\mathbb{E}^c(\Delta) =_{df} \emptyset$, otherwise*

$$\mathbb{E}^c(\Delta) =_{df} \bowtie \{[P_1]_{\mathscr{G}_i} : (P_1, \mathscr{G}_i) \in \Delta\}$$

For a concrete example of how the collect-and-combine rule is implemented in a popular SPARQL 1.0 federation system, the reader may want to peak ahead to subsection 6.1. From a theoretical point of view the rule may be motivated follows: To produce a result set from a distribution, each cell in the distribution has to be evaluated against its designated datasets and the partial answers that are returned have to be combined in order to yield an answer to the global query. The question, is how to unpack 'combine'. Note that just fixating on one operator will not do. Consider for instance, the case where $\mu_i \in [\![t]\!]_{G_i}$ and $\mu_j \in [\![t]\!]_{G_j}$ for incompatible μ_i and μ_j (i.e. suppose t contains a variable $?X$ such that $\mu_i(?X) \neq \mu_j(?X)$). Then neither μ_i or μ_j is in $[\![t]\!]_{G_i} \bowtie [\![t]\!]_{G_j}$, even though they are both answers to the query t. Alternatively, form the union of partial answers, and consider the case where $\mu = \mu_i \cup \mu_j$ for compatible $\mu_i \in [\![P_1]\!]_{G_i}$ and $\mu_j \in [\![P_2]\!]_{G_j}$. Then μ need not be in $[\![P_1]\!]_{G_i} \cup [\![P_2]\!]_{G_j}$, whereas μ_i and μ_j always are, despite the fact that μ but not μ_i and μ_j may be an answer to $P_1 \cup P_2$.

The collect-and-combine rule is designed to balance the need for both $\cup$ and $\bowtie$ as result-set forming operators, and to apply them in equal measure so to speak.

Theorem 5.5 *If δ is agnostic and egalitarian then $(\mathbb{E}^c, \delta)$ is sound and complete wrt. to the set of all selections of RDF graphs.*

Proof Let $\mathscr{G}$ be any selection of datasets and P any BGP. In the limiting case that δ is not defined at $\mathscr{G}$ and P we have $\mathbb{E}^c(\delta(\mathscr{G}, P)) = \emptyset = [\![P]\!]_{\mathscr{G}}$ by the limiting case of definition 5.2. For the principal case, suppose $\delta(\mathscr{G}, P) = (P_1, \mathscr{G}_1), \ldots, (P_n, \mathscr{G}_n)$. Then

$$\begin{aligned}
\mathbb{E}^c(\delta(\mathscr{G}, P)) &= [P_1]_{\mathscr{G}_1} \bowtie \ldots \bowtie [P_n]_{\mathscr{G}_n} && (1)\\
&= [\![P_1 \cup \ldots \cup P_n]\!]_{(\mathscr{G}_1 \cup \ldots \cup \mathscr{G}_n)} && (2)\\
&= [\![P]\!]_{(\mathscr{G}_1 \cup \ldots \cup \mathscr{G}_n)} && (3)\\
&= [\![P]\!]_{\mathscr{G}} && (4)
\end{aligned}$$

(2) follows from (1) by applying agnosticism in a straightforward induction on the number of arguments to the operator $\bowtie$, (3) follows from (2) by definition 4.3(3), and (4) follows from (3) by definition 4.3(1) and egalitarianism (its converse).

As the next subsection will show, the even-, standard- and prudent distributions are all agnostic and egalitarian in this sense. Hence, table 2 and 3 give examples of agnostic and egalitarian decompositions of FedBench query LS5.

5.1. The even-, standard- and prudent distribution

Theorem 5.5 specialises in a straightforward way to the even-, standard- and prudent distribution by way of the property egalitarianism (definition 4.4) and the following property of granularity:

Definition 5.3 (Granularity) *A distribution function δ is granular iff for every $(P_1, \mathscr{G}_i) \in \delta(\mathscr{G}, P)$, either P_1 or $\mathscr{G}_i$ is a singleton.*

It is evident by inspection of definition 4.3 that the even-, standard- and prudent distribution are all granular in this sense. Stretching the terminology somewhat, the federation scheme itself will be said to be granular if its distribution function is.

Granular distribution functions have the important property that they make sets $[P]_{\mathscr{G}}$ and $[\![P]\!]_{\mathscr{G}}$ coincide:

Lemma 5.6 *If $\mathscr{G}$ is a singleton then $[P]_{\mathscr{G}} = [\![P]\!]_{\mathscr{G}}$ for any BGP P*

Proof

$$\begin{aligned}[P]_{\mathscr{G}} &= \bigcup_{G\in\mathscr{G}} [\![P]\!]_G \\ &= \bigcup_{G\in\{G\}} [\![P]\!]_G \\ &= [\![P]\!]_G \\ &= [\![P]\!]_{\bigcup\{G\}} \\ &= [\![P]\!]_{\mathscr{G}}\end{aligned}$$

Lemma 5.7 *If P is a singleton then $[P]_{\mathscr{G}} = [\![P]\!]_{\mathscr{G}}$ for any selection of RDF graphs $\mathscr{G}$.*

Proof suppose P is a singleton t. The limiting case that $[P]_{\mathscr{G}} = \emptyset = [\![P]\!]_{\mathscr{G}}$ is immediate since P contains no joins. Assume the opposite, then there is a $G_i \in \mathscr{G}$ such that $\mu(t) \in G_i$ whence $\mu \in [\![t]\!]_{G_i}$. Now, $[\![t]\!]_{G_i} \subseteq \bigcup_{G_k\in\mathscr{G}} [\![t]\!]_{G_k} = [t]_{\mathscr{G}}$ so $\mu \in [t]_{\mathscr{G}}$. For the converse, suppose $\mu \in [t]_{\mathscr{G}} = \bigcup_{G_k\in\mathscr{G}} [\![t]\!]_{G_k}$ then for some $G_i \in \mathscr{G}$ it follows that $\mu \in [\![t]\!]_{G_i} \subseteq [\![t]\!]_{\mathscr{G}}$, by dataset monotony.

Thus,

Corollary 5.8 *Suppose δ is granular and that $(P_1, \mathscr{G}) \in \delta(\mathscr{G}, P)$, then $[\![P_1]\!]_{\mathscr{G}} = [P_1]_{\mathscr{G}}$.*

Now, if a distribution function is egalitarian then granularity suffices for agnosticism:

Theorem 5.9 *If δ is granular and egalitarian then δ is agnostic.*

Proof Let $(P_1, \mathscr{G}_i), (P_2, \mathscr{G}_j) \in \delta(\mathscr{G}, P)$ and assume that δ is granular. We show that $[P_1]_{\mathscr{G}_i} \bowtie [P_2]_{\mathscr{G}_j} = [\![P_1 \cup P_2]\!]_{(\mathscr{G}_i\cup\mathscr{G}_j)}$. Lemma 5.3 gives the left to right direction, so it suffices to prove the converse. Since δ is egalitarian it follows that $\Sigma_{\mathscr{G}}(P_1) = \mathscr{G}_i$ and $\Sigma_{\mathscr{G}}(P_2) = \mathscr{G}_j$. Since δ is granular we also have that $[\![P_1]\!]_{\mathscr{G}_i} = [P_1]_{\mathscr{G}_i}$ and $[\![P_2]\!]_{\mathscr{G}_j} = [P_2]_{\mathscr{G}_j}$. Thus, the conditions of lemma 5.4 are satisfied, whence $[\![P_1 \cup P_2]\!]_{(\mathscr{G}_i\cup\mathscr{G}_j)} \subseteq [P_1]_{\mathscr{G}_i} \bowtie [P_2]_{\mathscr{G}_j}$ as desired.

Corollary 5.10 *Suppose δ is the even-, standard or prudent distribution. Then $(\mathbb{E}^c, \delta)$ is sound and complete wrt. the set of all selections of RDF graphs.*

5.2. Non-granular and non-egalitarian federation schemes

It is a fact of some importance—and one that there shall be occasion to refer back to in the next section—that neither granularity nor egalitarianism are *necessary* conditions for the completeness of a federation scheme in a zero-knowledge environment. It also depends on the choice of evaluation rule.

In order to show examples of this, it will be necessary to introduce a couple of new notational conventions: first, given a set of variables $W \subseteq V$, by the restriction of μ to W, denoted $\mu_{|w}$, will be meant a partial function such that $dom(\mu_{|w}) = dom(\mu) \cap W$ and $\mu_{|W}(?X) = \mu(?X)$ for every $?X \in dom(\mu) \cap W$ [5]. Next, given a tuple $(\mu_1, \ldots, \mu_n)$ of compatible partial functions from V to U, it is a simple fact of general set theory that the union of all the elements in any permutation of this sequence is also a partial function from V to U. Moreover, all these permutations yield the same function modulo the union of its elements. Therefore, any permutation of $(\mu_1, \ldots, \mu_n)$, considered as a function, can be identified with $(\mu_1 \cup \ldots \cup \mu_n)$. By extension, if Ω_1 and Ω_2 are two sets of partial functions of the requisite type and every element of Ω_1 is compatible with every element of Ω_2, then $\Omega_1 \times \Omega_2$ will be treated as a set of partial functions of that type. Note that $\times$ considered in this way as a function-forming operator is both associative and commutative.

Now, consider BGPs that are as unconstrained as can be in the sense of having no joins and only variables or blank nodes in subject or object position:

Definition 5.4 *An* unconstrained *BGP P, is any BGP satisfying the following conditions:*

1. *if $t_1, t_2 \in P$ then $var(t_1) \cap var(t_2) = \emptyset$*
2. *$t_i^1, t_i^3 \in V \cup B$ for $i \in \{1, 2\}$*

Referring back to table 2, the BGP that consists of row 3 and 4 is unconstrained in this sense, as no two triples in it share a variable (5.4(1)) and as none of the terms in subject or object position is a URL or a literal value (5.4(2)). Indeed, it is a maximal unconstrained subset of 1-6, for row 1 has a URL in object position, whereas all other triple patterns share a variable with either 3 or 4.

A distribution function that lumps together triple patterns in unconstrained BGPs, in one way or the other (there are of course many), will henceforth be said to be *coalescent*:

Definition 5.5 *Call δ a* coalescent *distribution function if for every set of RDF graphs $\mathscr{G}$ and every BGP P_1, if $(P_2, \mathscr{G}_i) \in \delta(\mathscr{G}, P_1)$ and $|\mathscr{G}_i| \geq 2$ then P_2 is unconstrained.*

Example 5.2 *Consider the decomposition of the LS5 query from the FedBench suite given by table 5. This decomposition is coalescent but not granular: line 2 and 3 form an exclusive group relative to the Drugbank dataset, line 6 is an exclusive group relative to the ChEBI dataset—albeit a singleton one—whereas line 1 is a non-exclusive triple pattern assigned to all involved datasets. As regards lines 4 and 5, they form a group that is neither a singleton nor assigned to a single dataset, whence it is a group that violates the property of granularity. The two lines do not share variables, however, and do not have instantiated subjects or objects, and is therefore unconstrained in the sense of definition 5.4. It follows that the decomposition is coalescent in the sence of definition 5.5.*

The thing about an unconstrained BGP P is that it can be evaluated as-is against any RDF source G that covers its signature without any real loss of information. More precisely, the answer to each $t \in P$ over G can be reconstructed from the answers to P by *splitting* off the relevant chunks of the latter:

Definition 5.6 *Let $\mathscr{G}$ be any set of RDF graphs and P a conjunction of triple patterns $t_1, \ldots, t_n$. The* splitting *of $[P]_{\mathscr{G}}$ is defined as the empty tuple if $[P]_{\mathscr{G}} = \emptyset$, otherwise it is a sequence of sets $\Omega_1, \ldots, \Omega_n$ where $\Omega_i =_{df} \{\mu_{|var(t_i)} \,|\, \mu \in [P]_{\mathscr{G}}\}$.*

The following lemma verifies that the splitting of a result set suffices to recover all information:

Lemma 5.11 *Let P be an unconstrained BGP, $\mathscr{G}$ a set of RDF graphs such that $sig(P) \subseteq sig(G)$ for every $G \in \mathscr{G}$. Then*

$$[\![P]\!]_{\mathscr{G}} = \prod_{i=1}^{|P|} \overleftrightarrow{[P]}_{\mathscr{G}}$$

Proof By induction on the complexity of P. For the base case let $P := t_1 \cup t_2$. It needs to be shown that

$$[\![t_1 \cup t_2]\!]_{\mathscr{G}} = \prod_{i=1}^{2} \overleftrightarrow{[t_1 \cup t_2]}_{\mathscr{G}}$$

Consider first the limiting case that $[\![t_1 \cup t_2]\!]_{\mathscr{G}} = \emptyset$. Then, since $[\![t_1 \cup t_2]\!]_{\mathscr{G}}$ has been defined as $[\![t_1 \cup t_2]\!]_{\bigcup \mathscr{G}}$ and $G \in \mathscr{G}$, it follows by dataset monotony (lemma 5.1) that $[\![t_1 \cup t_2]\!]_G = \emptyset$ for every $G \in \mathscr{G}$. Therefore $[t_1 \cup t_2]_{\mathscr{G}} = \emptyset$ by definition 3.4, whence $\prod_{i=1}^{2} \overleftrightarrow{[t_1 \cup t_2]}_{\mathscr{G}}$ as desired.

For the left-to-right direction of the principal case put $\overleftrightarrow{[t_1 \cup t_2]}_{\mathscr{G}} := \Omega_1, \Omega_2$ and suppose $\mu \in [\![t_1 \cup t_2]\!]_{\mathscr{G}}$. Since $\mu \in [\![t_1 \cup t_2]\!]_{\mathscr{G}} = [\![t_1]\!]_{\mathscr{G}} \bowtie [\![t_2]\!]_{\mathscr{G}}$ it follows that $\mu = \mu_1 \cup \mu_2$ for some $\mu_1 \in [\![t_1]\!]_{G_1}$ and some $\mu_2 \in [\![t_2]\!]_{G_2}$. Suppose for reductio ad absurdum that $[\![t_1 \cup t_2]\!]_{G_1} = [\![t_1]\!]_{G_1} \bowtie [\![t_2]\!]_{G_1} = \emptyset$. Then $[\![t_2]\!]_{G_1} = \emptyset$, which since $Sig(t_2) \subseteq Sig(t_1 \cup t_2) \subseteq Sig(G_1)$ can only happen if $t_2^j \notin V \cup B$ for $j = 1$ or $j = 3$. However this contradicts definition 5.4(2). Assume therefore that there is an element $\mu' \in [\![t_2]\!]_{G_1}$. Since $var(t_1) \cap var(t_2) = \emptyset$ by definition definition 5.4(1) it follows that μ_1 and μ' are compatible, and thus that $\mu_1 \cup \mu' \in [\![t_1 \cup t_2]\!]_{G_1}$. Therefore $\mu_1 = (\mu_1 \cup \mu')_{|var(t_1)}$, i.e. μ_1 is the restriction of a partial function in $[\![t_1 \cup t_2]\!]_{G_1} \subseteq [t_1 \cup t_2]_{\mathscr{G}}$ from which it follows, by definition 5.6, that $\mu_1 \in \Omega_1$. A similar argument establishes $\mu_2 \in \Omega_2$. Hence $\mu_1 \cup \mu_2 = (\mu_1, \mu_2) = \Omega_1 \times \Omega_2 = \prod_{i=1}^{2} \overleftrightarrow{[t_1 \cup t_2]}_{\mathscr{G}}$, which completes the case. The converse direction is straightforward and is left for the reader.

For the induction step, suppose $P = P_1 \cup P_2$ and assume as the induction hypothesis that the theorem holds for P_1 and P_2. Then

$$\begin{aligned}
[\![P]\!]_{\mathscr{G}} &= [\![P_1 \cup P_2]\!]_{\mathscr{G}} && (2)\\
&= [\![P_1]\!]_{\mathscr{G}} \bowtie [\![P_2]\!]_{\mathscr{G}} && (3)\\
&= [\![P_1]\!]_{\mathscr{G}} \times [\![P_2]\!]_{\mathscr{G}} && (4)\\
&= \prod_{i=1}^{|P_1|} \overleftrightarrow{[P_1]}_{\mathscr{G}} \times \prod_{i=1}^{|P_2|} \overleftrightarrow{[P_2]}_{\mathscr{G}} && (5)\\
&= \prod_{i=1}^{|P|} \overleftrightarrow{[P]}_{\mathscr{G}} && (6)
\end{aligned}$$

(4) follows from (3) by the assumption that $var(P_1) \cap var(P_2) = \emptyset$ and (6) follows from (5) by the fact that $\times$ considered as a function-forming operator is both commutative and associative.

Theorem 5.11 suggests the following evaluation rule:

Definition 5.7 *Put $\Delta := \delta(\mathscr{G}, P)$. If δ is not defined at $\mathscr{G}$ and P we put $\mathbb{E}^{\times}(\Delta) =_{df} \emptyset$, otherwise*

$$\mathbb{E}^{\times}(\Delta) =_{df} \bowtie \{f(P_1, \mathscr{G}_i) : (P_1, \mathscr{G}_i) \in \Delta\}$$

Table 5
A coalescent distribution of LS5 over KEGG, ChEBI and Drugbank

Nr.	Triple pattern	a non-granular distribution
1	?drug rdf:type drugbank:drugs	All
2	?drug drugbank:keggCompoundId ?keggDrug	
3	?drug drugbank:genericName ?drugBankName	Drugbank
4	?keggDrug bio2rdf:url ?keggUrl	
5	?chebiDrug purl:title ?drugBankName	KEGG, ChEBI
6	?chebiDrug bio2rdf:image ?chebiImage	ChEBI

where

$$f(P_1, \mathscr{G}_i) = \begin{cases} \prod_{i=1}^{|P_1|} \overleftrightarrow{[P_1]}_{\mathscr{G}_i} & \textit{if } P_1 \textit{ is unconstrained} \\ [P_1]_{\mathscr{G}_i} & \textit{otherwise} \end{cases}$$

Given the conditions on the definition of a distribution function, it is not difficult to show that:

Theorem 5.12 *Let δ be a coalescent distribution function. Then the federation scheme $(\mathbb{E}^{\times}, \delta)$ is sound and complete wrt. the set of all selections of RDF graphs.*

The proof is a straightforward rerun of theorem 5.5 based on generalizing the property of agnosticism so that it reflects the $\mathbb{E}^{\times}$ rather than the $\mathbb{E}^{c}$ rule. Since the bulk of the work consists in proving lemma 5.11, the rest has been omitted for reasons of economy.

A similar result can easily be shown to hold for the property of egalitarianism, that is, there are non-egalitarian distribution functions that induce complete federation schemes wrt. the set of all selections of RDF graphs. Indeed, there are non-egalitarian δ such that $(\mathbb{E}^{\times}, \delta)$, specifically, is complete in this sense.

To see this, let δ be any coalescent distribution function and let δ' be exactly like δ, except that it also satisfies the following condition: if $(P_1, \mathscr{G}_i) \in \delta(\mathscr{G}, P)$ and $|P_1| \geq 2 \leq |\mathscr{G}_i|$ then $(t, G) \in \delta'(\mathscr{G}, P)$ for some $t \in P_1$ and some $G \in \mathscr{G}_i$.

Explained in words, for every unconstrained subquery P_1 of P which is assigned by δ to a set $\mathscr{G}_i$ with strictly higher cardinality than 1, δ' selects a random triple $t \in P_1$ and assigns it as a singleton subquery to a randomly selected dataset $G \in \mathscr{G}_i$. For example, assigning the triple pattern in line 5 of table 5 as a singleton subquery to, say, ChEBI would make that distribution both non-granular and non-egalitarian. Yet, it is entirely routine, given the results already presented, to prove that $\mathbb{E}^{\times}$ would still preserve zero-knowledge completeness.

Although these examples of non-granular and non-egalitarian distribution functions are admittedly somewhat contrived, their mere existence is a fact of some importance insofar as they prove that granularity and egalitarianism are not necessary for completeness. Contrapositively therefore, non-granularity and non-egalitarianism are not sufficient for *in*completeness—not even in combination.

6. Some examples from the literature.

With new federation engines being developed virtually by the hour, only a small sample can be discussed in the present section. The systems that *are* mentioned are not here claimed to be representative of ones left out. They have been selected merely because they are described, in text, with a sufficient level of precision to be amenable to an analysis using the concepts and techniques developed in the preceding sections.

The primary concern of the present section is to illustrate the application of the conceptual framework with an eye to the completeness of a federation scheme wrt. a set of selections of RDF graphs. It is only secondarily an evaluation of existing systems, and not at all an *empirical* evaluation. Specifically, none of the arguments and opinions adduced in the following are based on testing or inspecting source code.

Yet, if the said arguments miss their mark for that reason alone, that is if the systems discussed turn out to behave differently than described, then one way of looking at that fact is to take it as underlining the general theme of this paper: there is a need for a framework to allow implementation to be derived from theory rather than vice versa.

Having said that, it should be stressed that there are many approaches to federation for which completeness legitimately is not an important property. This applies to traversal-based federation [16,17,18,23], to top-k query processing [25,36,38], and in general to any approach based on the idea of probing the network

in an exploratory fashion. For such systems it is usually either not clear what completeness would be completeness with respect to, or it is simply not feasible to attain it.

Keeping the possibility open that there are also other valid reasons for deviating from completeness, the material that follows should not be taken as an argument to the effect that any of the approaches being discussed is inherently flawed.

6.1. FedX, DARQ and SPLENDID

The standard distribution has emerged as a point of convergence between several distributed query processing engines that have been proposed in the last half decade or so. Although one should probably be cautious about crediting any single reference with the idea, an early comer seems to have been the DARQ system proposed in Quilitz and Leser [31]. Later FedX [34] and SPLENDID [13] were based on the same idea—no dependencies implied.

All these approaches share three key observations: 1) that RDF properties can be used to assign subqueries to RDF sources, 2) that if the property of triple pattern occurs in multiple datasets, then that triple pattern must be sent as a singleton subquery to each of datasets in question in order not to neglect cross-site joins, and 3) if the signature of a subquery is covered by exactly one subquery, then that subquery does not need to be decomposed further but can be assigned to the dataset in question as-is. These three steps taken together amount to the standard distribution function.

It is has proved difficult to find a clear statement of evaluation rules—the importance of which seems largely to be ignored. However, Schwarte et al. [34] at least offer an example of how the Life Sciences 6 query from the FedBench suite would be evaluated according to their algorithm. The LS6 query is here reproduced in table 6 which also displays the signature coverage of each triple wrt. to the datasets KEGG, drugbank and DBpedia. The query execution plan from [34] is given in figure 1, with minor adjustments of notation: the leaves are numbered according to table 6, the brackets indicate the grouping of the sub-queries, and the '@'-tag above a leaf indicates a selection against the dataset suffixed to the tag.

There is reason to believe, however, that this execution plan contains a mistake, for the group $\{3, 4\}$ is not exclusive to drugbank and should therefore not, according to the express intents of Schwarte et al. [34], be evaluated as a single subquery.

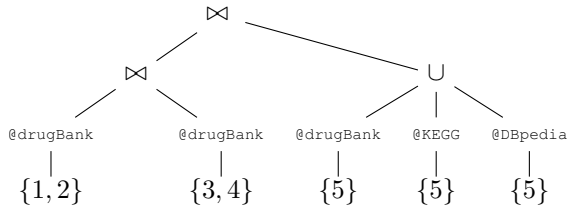
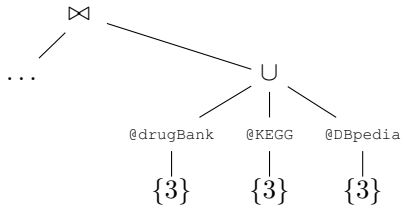


Fig. 1. Execution plan for LS6 from Schwarte et al. [34].

Triple pattern 3 is ?keggDrug `rdf:type` `kegg:Drug` whose signature is `rdf:type`, and that property belongs to *all* datasets in question. Therefore, the execution plan in figure 1 is not compatible with the standard distribution.

However, by separating 3 from 4 and inserting the branch



somewhere below the root node of the execution tree, the tree does instantiate an evaluation rule for the standard distribution which is in fact the collect-and-combine-rule $\mathbb{E}^c$. It is a particular version of the $\mathbb{E}^c$ rule in which joins are ordered by preference and exclusive groups come first—there will be a few more things to say about this in section 7.

It is reasonable to believe that it is this rule that is intended by [34] in which case the federation scheme proposed by FedX is $(\delta, \mathbb{E}^c)$, where δ is the standard distribution. This makes FedX a zero-knowledge approach.

6.2. DEFENDER/ANAPSID

DEFENDER [28] is a query decomposition engine that is built on top of the better known ANAPSID federator [3]. The main feature of the latter is that it adapts query execution plans to data availability and run-time conditions by providing physical SPARQL operators that detect when a source becomes blocked. These operators produce answers opportunistically as soon as data becomes available.

Given the opportunistic nature of DEFENDER that it inherits from ANAPSID, completeness of query an-

Table 6
FedBench setup for the Life Sciences 6 query.

Nr.	Triple pattern	Signature coverage
1	?drug drugBank:drugCategory drugbank-category:micronutrient	drugBank
2	?drug drugbank:casRegistryNumber ?id	drugBank
3	?keggDrug rdf:type kegg:Drug	All
4	?keggDrug bio2rdf:xRef ?id	KEGG
5	?keggDrug dc:title ?title	All

swering wrt. to the selected datasets is not necessarily an overriding concern—the discussion towards the end of [28] indicates that it is not. Nevertheless, it is natural to expect DEFENDER to at least approximate completeness in the limiting case that the contributing sources are sufficiently responsive. That is, it seems reasonable to expect completeness to act as a norm from which DEFENDER as well as ANAPSID itself will deviate only if circumstances dictate it.

The DEFENDER decomposition algorithm is based on evaluating so-called star-shaped query patterns, a notion which in turn derives from Vidal et al. [37]:

Definition 6.1 *Every pattern* $(?X, p, o)$ *or* $(s, p, ?X)$ *such that* $s \neq ?X$, $p \neq ?X$ *and* $o \neq ?X$ *is a star-shaped pattern wrt.* $?X$. *If* P_1 *and* P_2 *are star-shaped patterns wrt.* $?X$ *and* $var(P_1) \cap var(P_2) = \{?X\}$, *then* $P_1 \cup P_2$ *is a star-shaped pattern wrt.* $?X$.

Vidal et al. demonstrate that the two-step procedure of evaluating star-shaped patterns jointly before merging the result will typically be more efficient than evaluating and merging singleton triple patterns one-by-one.

DEFENDER implements this idea in a straightforward way. The relevant passage reads: "[DEFENDER] implements a greedy algorithm to group in the same subquery the triple patterns that share one variable and can be executed by the same endpoint. The query decomposer begins creating single sub-queries with triple patterns, then it merges the sub-queries that share exactly one variable, and repeats this process until a fixed-point is reached in the process of creating the sub-queries" [28, p. 2] .

In other words, DEFENDER assigns to each endpoint a *maximal* star whose signature is covered by an endpoint. One has to be careful with the maxtalk here, though, as the preceding sentence conceals two distinct senses: in the first sense a star $P' \subseteq P$ is maximal *with respect to an endpoint* G, i.e. there is no $P'' \subseteq P$ s.t. $P' \subset P''$ and $sig(P'') \subseteq sig(G)$. In the second sense P' being maximal means that there is no P'' with $P' \subset P'' \subseteq P$ such there is *some* G with $sig(P') \subseteq sig(P)$. One could call the former *local maximality* and the latter *global maximality*. The point is that a star can be locally maximal without being globally maximal: suppose P' is maximal wrt. G in the local sense. Then there is no strictly larger subpattern P'' of P whose signature is included in the signature of G. Yet, the signature of P'' may be included in the signature of *some* other endpoint G', in which case P' is not a globally maximal star.

The DEFENDER decomposition algorithm can now be described with more precision: it decomposes a query into *globally* maximal stars and assigns these stars to each endpoint whose signature includes that of the star. It is easy to verify that this algorithm is indeed a distribution function in the sense of definition 4.3. Table 7 shows the result of applying this algorithm to the LS5 query from the FedBench suite. Here each column of crosses represents a star-shaped pattern, and the header under which it is subsumed indicates the endpoint/dataset to which it is assigned.

Certain features of the induced distribution function can be read off from this table directly. For one, the distribution is egalitarian: the set of endpoints that a star is assigned to is precisely the set of endpoints that cover its signature. Indeed, this property generalizes and is due to the fact that stars are maximal in the global sense—interestingly, maximality in the local sense does *not* induce egalitarianism.

Moreover the decomposition in table 4.3 is granular too: every subquery is either a singleton or assigned to a singleton. Yet, this is incidental to the example: to see this, add another dataset encoded in, say, the ChEBI vocabulary to the datasets over which the LS5 query is distributed, call the two ChEBI datasets ChEBI 1 and ChEBI 2. In terms of the decomposition algorithm this has the effect of copying the stars assigned to ChEBI 1 into the columns of ChEBI 2 yielding the distribution in table 8 which is *not* granular since the non-singleton pattern consisting of row 5 and 6 is assigned to both of the ChEBI sources.

Note that column-copying, which corresponds to the distribtuion of a query over a set of sources some of

Table 7
Distribution of LS5 over KEGG, ChEBI and Drugbank according to DEFENDER.

		Maximal stars			
Nr.	Triple pattern	ChEBI		KEGG	Drugbank
1	?drug rdf:type drugbank:drugs				×
2	?drug drugbank:keggCompoundId ?keggDrug				×
3	?drug drugbank:genericName ?drugBankName				×
4	?keggDrug bio2rdf:url ?keggUrl	×		×	
5	?chebiDrug purl:title ?drugBankName		×		
6	?chebiDrug bio2rdf:image ?chebiImage		×		

which have the same signature, is not a far-fetched possibility. Relating to the example, ChEBI is part of the Open Biomedical Ontologies effort—an effort to create controlled vocabularies for *shared* use across different biological and medical domains. Indeed, the eventuality of multiple sources encoded in the same vocabulary is arguably both a desirable and an inevitable feature of the Semantic Web.

By the results of subsection 5.2, the non-granularity and unegalitarianism of a distribution function is not sufficient warrant to conclude that a federator is incomplete wrt. to the set of all selections of RDF graphs. Nevertheless, for DEFENDER this turns out to be so:

Example 6.1 *Put* $G_1 := \{(s_1, p_1, o_1), (s_2, p_2, o_2)\}$, $G_2 := \{(s_2, p_1, o_1), (s_1, p_2, o_2)\}$ *and* $\mathscr{G} := \{G_1, G_2\}$. *Consider the star-shaped pattern* $P := \{(?X, p_1, o_1), (?X, p_2, o_2)\}$. *The DEFENDER distribution function assigns the whole of* P *to both* G_1 *and* G_2*, i.e.* $\delta(\mathscr{G}, P) = \{(P, \mathscr{G})\}$. *By definition 3.4* $[P]_{\mathscr{G}} = [\![P]\!]_{G_1} \cup [\![P]\!]_{G_2}$ *but* $[\![P]\!]_{G_1} \cup [\![P]\!]_{G_2} = \emptyset$ *so* $[P]_{\mathscr{G}} = \emptyset$. *Yet,* $[\![P]\!]_{G_1 \cup G_2}$ *contains two answers* $\mu_1(?X) = s_1$ *and* $\mu_2(?X) = s_2$. *Hence* $[P]_{\mathscr{G}} \neq [\![P]\!]_{\mathscr{G}}$.

It follows immediately that $\mathbb{E}^c(\delta(\mathscr{G}, P)) = [P]_{\mathscr{G}} \neq [\![P]\!]_{\mathscr{G}}$, but the example in fact shows more: Since $[\![P]\!]_{G_1} = [\![P]\!]_{G_2} = \emptyset$, no subsequent manipulation of these sets will suffice to extract a complete answer from $\mathscr{G}$. It follows that the DEFENDER distribution is not complete wrt. the set of all selections of RDF graphs given *any* evaluation rule $\mathbb{E}$ that manipulates only the cells of the distribution it receives as an argument. Thus, if ad hoc evaluation rules are excluded, it follows that DEFENDER is zero-knowledge incomplete essentially.

6.3. ADERIS

The Adaptive Distributed Endpoint RDF Integration System (ADERIS) from Lynden et al. [24] is another example of an adaptive distributed query processor. That is, ADERIS seems to have a principal design goal that is similar to that of DEFENDER/ANAPSID, namely to have join-orderings change during query execution as a means of continual refinement and optimization of the execution plan at run-time.

ADERIS too is a distributed query processing system in the sense described in section 2, which means that it relies on a decomposition algorithm to distribute subqueries to contributing sources.

The ADERIS decomposition algorithm is rather different from that of DEFENDER, however, and is based on generating "for each data source, a source query (...) that contains all of the triple patterns from the federated query that could possibly match the triples in the given data source" [24, p. 182]. Unlike Montoya et al. [28], Lynden et al. [24] state explicitly that completeness is a property they wish the decomposition algorithm to have: "Source queries are constructed for each of these data sources, with the aim of retrieving all the data needed to answer the query" (p. 181).

In mathematical terms, the ADERIS distribution translates to a function which for every RDF source $G \in \mathscr{G}$ and a BGP P assigns the set of triples $t \in P$ whose property occurs in G to G. That is, the ADERIS distributions consists of pairs $(\{t \in P \mid sig(t) \in sig(G)\}, G)$. Again, it is easy to verify that this relation is a distribution function in the sense of definition 4.1.

Table 9 shows the result of applying the ADERIS distribution to the LS5 example. This particular decomposition, like that in table 9, is granular but not egalitarian. This time it is the group consisting of line 4 and 5 that constitutes the counterexample to egalitarianism: since the signature of this group is covered by both KEGG and ChEBI, egalitarianism dictates that it be assigned to both, which it is not.

The parallel extends further, for granularity is again incidental in precisely the same sense as for the distribution in table 7, i.e. non-granularity can be shown

Table 8
Distribution of LS5 over ChEBI 1, ChEBI 2, KEGG and drugbank according to DEFENDER.

		Maximal stars					
Nr.	Triple pattern	ChEBI 1		ChEBI 2		KEGG	Drugbank
1	?drug rdf:type drugbank:drugs						×
2	?drug drugbank:keggCompoundId ?keggDrug						×
3	?drug drugbank:genericName ?drugBankName						×
4	?keggDrug bio2rdf:url ?keggUrl	×		×		×	
5	?chebiDrug purl:title ?drugBankName		×		×		
6	?chebiDrug bio2rdf:image ?chebiImage		×		×		

Table 9
Distribution of LS5 over KEGG, ChEBI and Drugbank according to ADERIS.

Nr.	Triple pattern	ChEBI	KEGG	Drugbank
1	?drug rdf:type drugbank:drugs	×	×	×
2	?drug drugbank:keggCompoundId ?keggDrug			×
3	?drug drugbank:genericName ?drugBankName			×
4	?keggDrug bio2rdf:url ?keggUrl	×	×	
5	?chebiDrug purl:title ?drugBankName	×	×	
6	?chebiDrug bio2rdf:image ?chebiImage	×		

by column-copying. Moreover, ADERIS and DEFENDER are in complete agreement wrt. to example 6.1 insofar as they both yield the same distribution. It follows that the ADERIS too is incomplete wrt. the set of all selections of RDF graphs for any reasonable choice of evaluation rule $\mathbb{E}$.

6.4. AVALANCHE

The AVALANCHE system is a query answering system for the Semantic Web that, in the words of Başca and Bernstein [6], is designed to embrace the WWW uncertainties rather than to try to dispense with them. According to Başca and Bernstein, this means querying the Semantic Web without making any prior assumptions about the data location or distribution, schema-alignment, pertinent statistics, data evolution, or accessibility of servers [6, p. 2].

This is not to say, though, that AVALANCHE is a zero-knowledge federator in the present sense of the term—it is not. It merely means that AVALANCHE does not utilize *prior* information about the availability of endpoints, their schemas and/or their statistical characteristics. Rather AVALANCHE is designed to produce that information as part of an execution pipeline whose overall purpose is to answer an incoming query.

In the general case, AVALANCHE, like ADERIS and DEFENDER/ANAPSID, is opportunistic: it is designed to deliver partial results as they become available, favouring those that are easier to assemble. Like the other two approaches it relies on cost-reducing heuristics that adjust to the current computational environment, but it also relies on heuristics that is applied to query planning regardless.

For instance, the AVALANCHE query planner restricts the search for an optimal plan by considering only those distributions that assign a triple pattern to exactly one source (henceforth this will be called the *'single-source restriction'*). That is, the plans that are considered are such that a triple pattern will never be executed against more than one RDF dataset.

Başca and Bernstein [6, listing 2, page 7] illustrate this decomposition algorithm by giving an example distribution of the LS5 query reproduced here in table 10 (note that the example assumes the `bio2rdf:image` property to belong to the signature of KEGG as well).

It is immediate by inspection of the table that this particular decomposition is granular. This time granularity is not an incidental property of the example but a consequence of the single-source restriction. Equally obviously, and for the same reason, AVALANCHE is not egalitarian—viz. row 6 is assigned only to KEGG although it is also covered by ChEBI.

In fact, AVALANCHE like DEFENDER/ANAPSID and ADERIS, can easily be seen to be incomplete wrt. to the set of all sets of RDF graphs for any reasonable choice of evaluation rule $\mathbb{E}$. Again example 6.1 suffices

Table 10
Distribution of LS5 over KEGG, ChEBI and Drugbank according to AVALANCHE.

Nr.	Triple pattern	ChEBI	KEGG	Drugbank
1	?drug rdf:type drugbank:drugs			×
2	?drug drugbank:keggCompoundId ?keggDrug			×
3	?drug drugbank:genericName ?drugBankName			×
4	?keggDrug bio2rdf:url ?keggUrl	×		
5	?chebiDrug purl:title ?drugBankName	×		
6	?chebiDrug bio2rdf:image ?chebiImage		×	

to show this: if, relating to this example, the single-source restriction is to be observed, then the the distribution needs to choose to evaluate the BGP P_1 against exactly one of the datasets G_1 and G_2. It should be clear that no matter which one is chosen, the result set will be empty, whence no subsequent manipulation of result sets will be able to extract a complete answer.

As an afterthought, it is instructive to contrast this conclusion with the discussion of the homonymous notions of completeness in [6] itself: it cannot be the aim of AVALANCHE to deliver *global completeness*, Başca and Bernstein argue, if a globally complete answer is understood as one that comprises all answers to a query that is to be had on the entire Web. However, considering only the datasets that are selected for the query answering process, so the argument goes, it is natural to aim for *query-contextual completeness*, by which is meant that the answer is complete wrt. the set of selected datasets if none of them go down or become unavailable. If the query execution process is not stopped, [6, p. 4] claim, AVALANCHE is *eventually* complete in the query-contextual sense.

This notion of query-contextual completeness does not coincide with the *formal* notion of completeness promoted in this paper, according to which all answers that are contained by the union of the contributing datasets are always obtained by the federator.

6.5. Discussion

The preceding discussion cannot pretend to do justice to the systems discussed. They are all much broader in scope and ambition than the narrow focus on completeness suggests, covering besides a plethora of interesting and important topics such as adaptivity, opportunism, indexing, join-order optimization, statistics-generation and more.

As to the question of the significance of completeness itself, a two-fold answer is probably most appropriate: from a pragmatic point of view completeness is essentially subservient to user needs. An incomplete answer that returns in reasonable time is usually preferable to a complete one that makes excessive demands on time and memory.

Looked at from a theoretical perspective, however, a completeness result offers an exhaustive characterisation of the *behaviour* of a federation scheme. Being complete wrt. a set of selections of RDF graphs, entails being incomplete wrt. to its complement. Knowing how a federator behaves is knowing the conditions under which it crosses this line. In working out these conditions one is effectively giving an account of when a federation engine will miss answers and why.

As argued throughout this paper, an exhaustive characterisation of a federation strategy is obtained by turning three knobs: the way a query is decomposed into sub-queries, the way the partial answers are subsequently combined, and the way that a selection of RDF graphs is chosen for federation.

Depending on how much one turns one knob, it may not have any effect turning another. For instance, example 6.1 has shown that the DEFENDER/ANAPSID, ADERIS and AVALANCHE distribution functions are incomplete wrt. to the set of all selections of RDF graphs for all reasonable choices of evaluation rule $\mathbb{E}$. Thus, the only way to obtain an exhaustive characterisation of these systems' behaviour is to constrain the selection of RDF datasets to the point where it matches the federation scheme. Although, this intersection point may be too restrictive to adhere to in practise, characterising it will serve to make assumption about the structure, content, and relationship between contributing RDF sources explicit. Examples of what these assumptions may amount to are: do the contributing sources have disjoint sets of subjects and/or objects? Do they have disjoint signatures? Can all subjects be assumed to be typed? Can the RDF graphs be assumed to conform to some predefined schema, pattern or shape (cf. Ryman et al. [32])? And so on and so forth.

Needless to say, there is large variety of ways in which a set of selections of RDF sources can be con-

strained in this manner, and therefor also a large number of distribution schemes to match.

7. A corollary wrt. join-order heuristics

The account of distributed query processing given so far is entirely abstract and declarative, and it is far from obvious how one would move towards the implementation level from the theory. As a stepping stone, this section operationalizes the distribution semantics by showing how to fit a federation scheme into a join-ordering in order to obtain an execution plan for a given distribution that is provably sound and complete. Attention is restricted to zero-knowledge completeness and the collect-and-combine rule.

A join-ordering will be represented as usual as an operator tree, based on the following definitions from [15]:

Definition 7.1 (Tree domain) *A tree domain D is a non-empty subset of the set of all strings $\mathbb{N}^*$ over the natural numbers, satisfying the following conditions:*

1. *For each $u \in D$ every prefix of u is also in D.*
2. *For each $u \in D$ and $i \in \mathbb{N}^*$, if $ui \in D$ then for every j, $1 \leq j \leq i$, $uj \in D$.*

Given a tree domain D the relation $x \leq y$ denotes the prefix relation on strings of D. The covering relation $x \prec y$ is defined by stipulating that if $x \prec y$ and $x \leq z \leq y$ then either $x = z$ or $z = y$.

Definition 7.2 *Given a set Σ of symbols, a tree over Σ is a function $d : D \longrightarrow \Sigma$ where D is a tree domain.*

Definition 7.3 (Subtrees) *Given a tree d and a node u in $dom(d)$, the* subtree rooted at u *is the tree d/u whose domain is the set $\{v : uv \in dom(d)\}$ and such that $(d/u)(v) = d(uv)$ for all v in $dom(d/u)$.*

Every element of a domain D of a tree d will be called a node of d. The *outdegree* $out(u)$ of a node u is the cardinality of the set $\{i : ui \in dom(d)\}$. A node u with $out(u) = 0$ is called a *leaf*. The leaves of d are denoted $L(d)$. Indegrees are defined as $in(u) = \{i : iu \in dom(d)\}$. Of course all nodes in a tree have indegree equal to 1 except one, the root, which has indegree zero. If d is a tree, $r(d)$ denote the root of d.

Definition 7.4 (Ordered distribution) *Let Δ be a distribution. A join-ordering of Δ is a tree d over $\{\bowtie\} \cup \Delta$ such that:*

1. *d is a bijection between $L(d)$ and Δ*
2. *if $out(x) > 1$ then $d(x) = \bowtie$*

Definition 7.4 represents a generalisation of the usual notion of the join-ordering of a BGP. It is a generalization in the sense that the carrier set of the poset is not a set of triples but a distribution, that is, itself a set of BGPs.

In computational terms it is natural to see this generalization as an adaptation in which the join-ordering also distributes the responsibility for executing joins: a distribution assigns a subquery of a global query to a set of contributing sources. If the subquery in question contains more than one triple pattern—which will be the case if it is e.g. an exclusive group—then the subquery is handed over to the contributing sources along with the responsibility for executing the joins involved.

It should be fairly obvious how to relate definition 7.4 to the completeness results from section 5:

Definition 7.5 (Evaluation) *The evaluation of an ordered distribution d, written $[\![d]\!]$, is*

1. *$[P]_{\mathscr{G}}$ if $dom(d)$ contains a single leaf x such that $d(x) = (P, \mathscr{G})$*
2. *otherwise it is the application of $r(d)$ to $[\![d/y_1]\!]$ and $[\![d/y_2]\!]$ for $i \in [1, 2]$*

We have:

Corollary 7.1 *Let δ be either of the even-, standard- or prudent distributions and suppose it is defined at $\mathscr{G}$ and P. Let d be a join-ordering of $\delta(\mathscr{R}, P)$. Then $[\![d]\!] = \mathbb{E}^c(\delta(\mathscr{G}, P))$.*

Proof By definition 7.4(2) and definition 7.4(2) we have that $[\![d]\!] = \bowtie\ (d(L(D))$. Therefore, since we have $d(L(d)) = \delta(\mathscr{G}, P)$ by definition 7.4(2) it follows that $[\![d]\!] = \mathbb{E}^c(\delta(\mathscr{G}, P))$ as desired.

None of this is surprising, which is why it is presented as a corollary.

Nevertheless there are a couple of things to note: First, the unit of heuristic significance in a distributed setting is the subpattern, not the triple pattern, and all subpatterns are by default equally privileged. That is, there is no constraint on the shape of a tree that dictates that e.g. exclusive groups be processed first. As hinted at in subsection 6.1, this contrast with how evaluation is performed by e.g. FedX where exclusive groups are given priority based on the claim that they are usually more constrained than singleton triple patterns [34, p. 607].

This is probably true, but it is not *necessarily* so. Heuristic rules that do not rely on statistics will usually determine a priority ordering between patterns by analysing their lexical form. Typically the heuristic value of a single triple will be determined by assessing how selective that triple pattern is relative to others. For instance a pattern of form `(?X, p, o)` can plausibly taken to be more constrained than `(?X, p, ?Y)` and so forth, whereas the heuristic value of a join $t_1 \bowtie t_2$ will be determined by ranking the position and number of shared variables involved (cf. [35]). There is no *a priori* reason why such a procedure, generalised from triple patterns to BGPs in the obvious manner, should assign a higher accumulated heuristic value to an exclusive group, say. On the contrary it is not too difficult to come up with intuitive counterexamples that involve exclusive groups with a high variable to term ratio and a small number of joins, being ranked below a single triple pattern with an instantiated subject and object.

The second thing to note, is the generality of the collect-and-combine rule which yields sound and complete heuristics for each of the even-, standard and prudent distributions in one fell swoop. Indeed, one may speculate that there is little reason for employing a different evaluation rule with these distributions functions—or, at any rate, at least in the zero knowledge case. If there were, then it would be for heuristic reasons. However, in the absence of detailed knowledge about the contributing sources there seems to be no reason to think that some other particular evaluation order would do better than the collect-and-combine rule. Whilst evaluating the two pairs $(\{t_1, t_2\}, \mathscr{G}_i)$ and $(t_3, \mathscr{G}_j)$, say, by distributing joins over unions $([t_1]_{\mathscr{G}_i} \bowtie [t_3]_{\mathscr{G}_j}) \cup ([t_2]_{\mathscr{G}_i} \bowtie [t_3]_{\mathscr{G}_j})$ would preserve completeness, the lexical form alone of the patterns involved does not lend support to this evaluation procedure over the collect-and-combine rule—nor vice versa of course.

8. Conclusion

This foundational study has provided some first steps towards a logical framework for reasoning about distributed query processing in SPARQL in a formal manner. The analysis is based on the general idea of correlating a 'syntactic' object consisting of a distribution function and an evaluation rule with a 'semantic' object consisting of a family of sets of RDF graphs, each one of which represents an eligible selection of sources for the federation process.

The combination of granular and egalitarian distributions function with the collect-and-combine evaluation rule has emerged as a trait that induces completeness in a zero knowledge computational environment for several natural examples of distribution functions, including the standard distribution, so-called, which occurs frequently in the literature.

Federation schemes have a simple and transparent interface towards the heuristic notion of a join-ordering of a query. This interface connects execution plans to soundness and completeness results in a manner that makes it evident that whole classes of execution plans preserve these properties.

An interesting line of future research would be to investigate natural candidates for some-knowledge federation schemes, and to furnish them with completeness results by restricting the set of eligible selections. This should include a characterization of existing some-knowledge approaches. For instance; what needs to be assumed about the distribution of data over a selection of RDF datasets for a distribution scheme based on maximal stars in the manner of Vidal et al. [37] to be complete? Or, are there any established resource shapes [32] that could be analysed this way? This ought to be a rich field for future work.

Acknowledgements. The author is grateful to Carlos Buil-Aranda, Jürgen Umbrich, and one other anonymous reviewer for constructive criticism and insightful questions regarding earlier drafts of this paper. The author also wishes to thank Jonas Halvorsen and Bjørn Jervell-Hansen for stimulating discussions leading up to and advancing the present line of research.

References

[1] S. Abiteboul and V. Vianu. Queries and computation on the web. In F. Afrati and P. Kolaitis, editors, *Proceedings of the 6th International Conference on Database Theory*, volume 1186 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg 1997, pages 262–275.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[3] M. Acosta, M.-E. Vidal, T. Lampo, J. Castillo, and E. Ruckhaus. Anapsid: An adaptive query processing engine for SPARQL endpoints. In *Proceedings of the 10th International Semantic Web Conference*. Springer Berlin Heidelberg 2011, pages 18–34.

[4] Z. Akar, T.G. Halaç, E.E. Ekinci, and O. Dikenelli. Querying the web of interlinked datasets using VoID descriptions. In C. Bizer et al., editors, *WWW2012 Workshop on Linked Data on the Web*. Lyon France 2012.

[5] M. Arenas, C. Gutierrez, and J. Pérez. Foundations of RDF databases. In S. Tessaris et al., editors, *Reasoning Web. Semantic Technologies for Information Systems*, volume 5689 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg 2009, pages 158–204.

[6] C. Başca and A. Bernstein. Querying a messy web of data with avalanche. *Web Semantics: Science, Services and Agents on the World Wide Web* volume 26, 2014, pages 1–28.

[7] C. Basca and A. Bernstein. Avalanche: Putting the spirit of the web back into semantic web querying. *The 9th International Semantic Web Conference, Posters & Demo Session*. Bonn Germany 2010.

[8] H. Betz, F. Gropengießer, K. Hose, and K.-U. Sattler. Learning from the history of distributed query processing - a heretic view on linked data management. In *Proceedings of the 3rd International Workshop on Consuming Linked Data*. Boston, MA, USA 2012.

[9] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems* volume 5, 2009, pages 1–22.

[10] P. Bouquet, C. Ghidini, and L. Serafini. Querying the web of data: A formal approach. In A. Gómez-Pérez et al., editors, *The Semantic Web*, volume 5926 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg 2009, pages 291–305.

[11] C. Buil-Aranda, M. Arenas, and O. Corcho. Semantics and optimization of the SPARQL 1.1 federation extension. In G. Antoniou et al., editors, *The Semanic Web: Research and Applications*, volume 6644 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2011, pages 1–15.

[12] H. Choi, J. Son, Y. Cho, M.K. Sung, and Y.D. Chung. SPIDER: a system for scalable, parallel / distributed evaluation of large-scale RDF data. In *Proceedings of the 18th ACM conference on Information and knowledge management*. New York, NY, USA, 2009, pages 2087–2088.

[13] O. Görlitz and S. Staab. SPLENDID: SPARQL Endpoint Federation Exploiting VoID Descriptions. In *Proceedings of the 2nd International Workshop on Consuming Linked Data*. Bonn Germany 2011.

[14] O. Görlitz, and S. Staab. Federated data management and query optimization for linked open data. *In* A. Vakali and L. C. Jain, eds, *New Directions in Web Data Management 1*, volume 331 of *Studies in Computational Intelligence*. Springer 2011, pages 109–137.

[15] S. Gorn. Explicit definitions and linguistic dominoes. In J. Hart and S. Takasu, editors, *Systems and computer Science*. University of Toronto Press 1967, pages 77–105.

[16] O. Hartig. SPARQL for a web of linked data: Semantics and computability. In *Proceedings of the 9th International Conference on The Semantic Web: Research and Applications*. Springer-Verlag Berlin Heidelberg 2012, pages 8–23.

[17] O. Hartig. Zero-knowledge query planning for an iterator implementation of link traversal based query execution. In *Proceedings of the 8th Extended Semantic Web Conference on The Semantic Web: Research and Applications*. Springer-Verlag Berlin Heidelberg 2011, pages 154–169.

[18] O. Hartig, C. Bizer, and J.C. Freytag. Executing SPARQL queries over the web of linked data. In *Proceedings of the 8th International Semantic Web Conference*. Springer-Verlag, Berlin Heidelberg 2009, pages 293–309.

[19] K. Hose, R. Schenkel, M. Theobald, and G. Weikum. Database foundations for scalable RDF processing. In *Proceedings of the 7th international conference on Reasoning web: semantic technologies for the web of data*. Springer-Verlag Berlin Heidelberg 2011, pages 202–249.

[20] J. Huang, D. J. Abadi, and K. Ren. Scalable SPARQL querying of large RDF graphs. In J. Blakeley et al., editors, *The Proceedings of the VLDB Endowment*, volume 4 number 11, 2011, pages 1123–1134.

[21] M.F. Husain, J. McGlothlin, M.M. Masud, L.R. Khan, and B. Thuraisingham. Heuristics-based query processing for large RDF graphs using cloud computing. In *IEEE Transactions on Knowledge and Data Engineerin*, volume 23 issue 9, 2011, pages 1312–1327.

[22] D. Kossmann. *The state of the art in distributed query processing*. *ACM Computing Surveys* volume 32 number 4. ACM New York, USA, 2011, pages 422–469.

[23] Günter Ladwig and Thanh Tran. SIHjoin: Querying remote and local linked data. In G. Antoniou et al., editors, *The Semantic Web: Research and Applications*, volume 6643 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg 2011, pages 139–153.

[24] S. Lynden, I. Kojima, A. Matono, and Y. Tanimura. Adaptive integration of distributed semantic web data. In A. Madaan et al., editors, *8th International Workshop on Databases in Networked Information Systems*. Volume 7813 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg 2010, pages 174–193.

[25] S. Magliacane, A. Bozzon, and E. Della Valle. Efficient execution of top-k SPARQL queries. In P. Cudré-Mauroux et al., editors, *Proceedings of the 11th International Semantic Web Conference*, volume 7649 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg 2012, pages 344–360.

[26] D. C. Makinson. Completeness theorems, representation theorems: What's the difference? In Rønnow-Rasmussen et al., editor, *Hommage á Wlodek: Philosophical Papers dedicated to Wlodek Rabinowicz*. Electronic publication, 2007.

[27] G. Montoya, M.-E. Vidal, and M. Acosta. A heuristic-based approach for planning federated SPARQL queries. In J. Sequeda et al., editors, *Proceedings of the 3rd International Workshop on Consuming Linked Data*, volume 905 of *CEUR Workshop Proceedings*. Boston, MA, USA, 2012.

[28] G. Montoya, M.-E. Vidal, and M. Acosta. Defender: a decomposer for queries against federations of endpoints. *The 9th Extended Semantic Web Conference, Workshop and Demo Session*. Heraklion, Crete, 2012.

[29] N. Papailiou, I. Konstantinou, D. Tsoumakos, and N. Koziris. H2rdf: adaptive query processing on RDF data in the cloud. In A. Mille et al., editors, *Proceedings of the 24th International World Wide Web Conference (Companion Volume)*. ACM 2012, pages 397–400.

[30] A. Seaborne et al., ed. E. Prud'hommeux. SPARQL 1.1. federated query. W3C recommendation march 2013. URL `http://www.w3.org/TR/sparql11-federated-query/`.

[31] B. Quilitz and U. Leser. Querying distributed RDF data sources with SPARQL. In S. Bechhofer et al., editors, *Proceedings of the 5th European Semantic Web Conference*, Tenerife, Canary Islands, Spain. Volume 5021 of *Lecture Notes in Computer Science*. Springer 2008.

[32] A. G. Ryman, A. Le Hors, and S. Speicher. OSLC resource shape: A language for defining constraints on linked data. In *Proceedings of the Linked Data on the Web Workshop 2013*.

Rio de Janeiro, Brazil 2013.

[33] M. Schmidt, O. Görlitz, P. Haase, G. Ladwig, A. Schwarte, and T. Tran. FedBench: A benchmark suite for federated semantic data query processing. In L. Aroyo et al., editors, *Proceedings of the 10th International Semantic Web Conference*. Volume 7031 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg 2011, pages 585–600.

[34] A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt. FedX: optimization techniques for federated query processing on linked data. In L. Aroyo et al., editors, *Proceedings of the 10th International Semantic Web Conference*. Springer Berlin Heidelberg 2011, pages pages 601–616.

[35] P. Tsialiamanis, L. Sidirourgos, I. Fundulaki, V. Christophides, and P. Boncz. Heuristics-based query optimisation for SPARQL. In *Proceedings of the 15th International Conference on Extending Database Technology*. ACM New York, NY, USA, 2012, pages 324–335.

[36] E. Della Valle, S. Schlobach, M. Krö tzsch, A. Bozzon, S. Ceri, and I. Horrocks. Order matters! harnessing a world of orderings for reasoning over massive data. *Semantic Web Journal*, volume 4 number 2. IOS Press 2013, pages 219–231.

[37] M.-E. Vidal, E. Ruckhaus, T. Lampo, A. Martinez, J. Sierra, and A. Polleres. Efficiently joining group patterns in SPARQL queries. In L. Aroyo et al., editors, *The Semantic Web: Research and Applications*, volume 6088 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg 2010, pages 228–242.

[38] A. Wagner, T. T. Duc, G. Ladwig,A. Harth, and R. Studer. Top-k linked data query processing. In *Proceedings of the 9th international conference on The Semantic Web: research and applications*. Springer-Verlag Berlin Heidelberg 2012, pages 56–71.