# VoIDext: Vocabulary and Patterns for Enhancing Interoperable Datasets with Virtual Links

Tarcisio Mendes de Farias[1,2](✉), Kurt Stockinger[4],
and Christophe Dessimoz[1,2,3]

[1] SIB Swiss Institute of Bioinformatics, Lausanne, Switzerland
[2] University of Lausanne, Lausanne, Switzerland
tarcisio.mendesdefarias@unil.ch
[3] University College London, London, UK
[4] Zurich University of Applied Sciences, Winterthur, Switzerland

**Abstract.** Semantic heterogeneity remains a problem when interoperating with data from sources of different scopes and knowledge domains. Causes for this challenge are context-specific requirements (i.e. no "one model fits all"), different data modelling decisions, domain-specific purposes, and technical constraints. Moreover, even if the problem of semantic heterogeneity among different RDF publishers and knowledge domains is solved, querying and accessing the data of distributed RDF datasets on the Web is not straightforward. This is because of the complex and fastidious process needed to understand how these datasets can be related or linked, and consequently, queried. To address this issue, we propose to extend the existing Vocabulary of Interlinked Datasets (VoID) by introducing new terms such as the *Virtual Link Set* concept and data model patterns. A virtual link is a connection between resources such as literals and IRIs (Internationalized Resource Identifier) with some commonality where each of these resources is from a different RDF dataset. The links are required in order to understand how to semantically relate datasets. In addition, we describe several benefits of using virtual links to improve interoperability between heterogenous and independent datasets. Finally, we exemplify and apply our approach to multiple world-wide used RDF datasets.

**Keywords:** Data interoperability · Virtual link · Vocabulary of Interlinked Datasets (VoID) · Federated query

## 1 Introduction

To achieve semantic and data interoperability, several data standards, ontologies, thesauri, controlled vocabularies, and taxonomies have been developed and adopted both by academia and industry. For example, the Industry Foundation Classes [7] is an ISO standard to exchange data among Building Information Modelling software tools [16]. In life sciences, we can mention the Gene Ontology (GO) among many other ontologies listed in repositories such as BioPortal

H. Panetto et al. (Eds.): OTM 2019, LNCS 11877, pp. 607–625, 2019.
https://doi.org/10.1007/978-3-030-33246-4_38

[31]. Yet, semantic heterogeneity remains a problem when interoperating with data from various sources which represent the same or related information in different ways [15]. This is mainly due to the lack or difficulty of a common consensus, different modelling decisions, domain scope and purpose, and constraints (e.g. storage, query performance, legacy and new systems).

Semantic reconciliation—i.e. the process of identifying and resolving semantic conflicts [29], for example, by matching concepts from heterogeneous data sources [18]—is recognized as a key process to address the semantic heterogeneity problem. To support this process, ontology matching approaches [26] have been proposed such as YAM++ [24]. Although semantic reconciliation enhances semantic interoperability, it is often not fully applicable or practical when considering distributed and independent RDF (Resource Description Framework) datasets of different domain scopes, knowledge domains, and autonomous publishers. In addition, even if the semantic reconciliation process among different RDF publishers and knowledge domains is complete and possible, querying and accessing the data of multiple distributed RDF datasets on the Web is not straightforward. This is because of the complex, time-consuming and fastidious process of having to understand how the data are structured and how these datasets can be related or linked, and consequently, queried.

To enhance interoperability and to facilitate the understanding of how multiple datasets can be related and queried, we propose to *extend and adapt the existing Vocabulary of Interlinked Datasets* (VoID) [2]. VoID is an RDF Schema vocabulary used to describe metadata about RDF datasets such as structural metadata, access metadata and links between datasets. However, VoID is limited regarding terms and design patterns to model the relationships between datasets in a less verbose, unambiguous and explicit way. To overcome this problem, we introduce the *concept of virtual link set* (VLS). A virtual link is an intersection data point between two RDF datasets. A data point is any node or resource in an RDF graph such as literals and IRIs (Internationalized Resource Identifier). An RDF dataset is a set of RDF triples that are published, maintained or aggregated by a single provider [2]. The links are required in order to comprehend how to semantically relate datasets. The major advantage of the VLS-concept is to facilitate the writing of federated SPARQL queries [19], by acting as joint points between the federated sources. We *exemplify and apply VoIDext to various world-wide used data sets* and discuss both the theoretical and practical implications of these new concepts with the goal of more easily querying heterogeneous and independent datasets.

This article is structured as follows: Sect. 2 presents the relevant related work. Section 3 details our approach to extend the VoID vocabulary. In Sect. 4, we describe the major benefits of using VoIDext, and we apply VoIDext to describe VLSs among three world-wide used bioinformatics RDF data stores. Finally, we conclude this article with future work and perspectives.

## 2 Related Work

Since the release of the SPARQL 1.1 Query Language [19] with federated query support in 2013, numerous federated approaches for data and semantic interoperability have recently been proposed [11,20,33], and [32]. However, to the best of our knowledge, none of them proposes a vocabulary and patterns to extensively, explicitly and formally describe how the data sources can be interlinked for the purpose of facilitating the writing of SPARQL 1.1 federated queries such as discussed in Sect. 3. In effect, existing approaches put the burden on the SPARQL users or systems to find out precisely *how* to write a conjunctive federated query. An emerging research direction entails automatically discovering links between datasets using Word Embeddings [17]. However, the current focus is mostly on relational data or unstructured data [6]. In addition, several link discovery frameworks such as in [25,28], and [21] rely on link specifications to define the conditions necessary for linking resources within datasets. With these specifications, these frameworks describe similarity measures or distance metrics (e.g. Levenshtein, Jaccard and Cosine) as part of conditions to determine, for example, whether two entities should be linked. The approaches of [17,25,28], and [21] are complementary to ours because they can aid in the process of defining virtual link sets—see Definition 2.

In the context of ontology alignment, the Expressive and Declarative Ontology Alignment Language (EDOAL) enables us to represent correspondences between heterogeneous ontological entities [9]. Although, transformations of property values can be specified with EDOAL, the current version of EDOAL solely supports a limited kind of transformations[1]. In [8] and [10], authors also recognize the limited support for data transformation in mapping languages. Moreover, since EDOAL does not focus on supporting the write of SPARQL 1.1 federated queries, the EDOAL data transformation specification requires an extra step to be converted into an equivalent one by using the SPARQL language. Applying data transformations during a federated query execution is often required to be able to link real-world independent and distributed datasets on the Web. As other related work in terms of RDF-based vocabularies, we can also mention VoID and SPARQL 1.1 Service Description (SD)[2]. Although the VoID RDF schema provides the *void:Linkset* term (Definition 1), this concept alone is not sufficient to precisely and explicitly define virtual links between the datasets (discussed in Sect. 3). By precisely, we mean to avoid multiple ways to represent (i.e. triple patterns) and to interpret interlinks. Moreover, by considering Definition 1 extracted from the VoID specification, this definition impedes the use of the *void:Linkset* concept to describe a link set between instances of the same class because both are triple subjects stored in different datasets.

**Definition 1 (link set –** ***void:Linkset***[3]**).** *A collection of RDF links between two datasets. An RDF link is an RDF triple whose subject and object are described in different datasets [2].*

## 3 Contribution

To mitigate the impediments of interoperating with distributed and independent RDF datasets, we first propose design patterns of how to partially model

[1] http://alignapi.gforge.inria.fr/edoal.html.

[2] https://www.w3.org/TR/sparql11-service-description/.

[3] http://vocab.deri.ie/void#Linkset.

virtual links (see Definition 2) with the current VoID vocabulary and expose its drawbacks. To address these drawbacks, we then propose a new vocabulary (i.e. VoIDext) and demonstrate an unambiguous and unique way to extensively and explicitly describe various types of virtual links such as depicted in Subsect. 3.2. The VoIDext vocabulary is fully described in [14], that also includes examples of virtual link types.

**Definition 2 (virtual link set).** *A set of virtual links. A virtual link is a connection between common resources such as literals and instances from two different RDF datasets. Semantic relaxation (see Definition 3) is also considered when identifying common resources between datasets.*

**Definition 3 (semantic relaxation).** *It is the capacity of ignoring semantic and data heterogeneities for the sake of interoperability.*

In this article, the words *vocabulary* and *ontology* are used interchangeably. The methodology applied to develop VoIDext was inspired by the simplified agile methodology for ontology development (SAMOD) [27]. Indeed, the proposed VoID extension is a meta-ontology to explicitly describe interlinks between RDF datasets—virtual links. Further information about how VoIDext was built is given in the Supplementary Material in [13].

Figure 1 illustrates a complex virtual link about Swiss cantons between the LINDAS dataset (Linked Data Service[4]) of the Swiss Government administration and DBpedia [23]. To define this link, some semantic relaxation is applied. This is because heterogeneities are exacerbated when interoperating independent datasets. For example, what is considered a long name of a Swiss canton in LINDAS is actually a short name in DBpedia. In addition, the data types for the name of the canton are not the same in both datasets what impedes exact matching when performing a federated join query. Finally, LINDAS contains a few literals with different concatenated translations of the same canton's name such as "Graubünden / Grigioni / Grischun" that can be matched with the literal "Grisons" asserted as a canton's short name in DBpedia. Indeed, Grison is the French translation of Graubünden—German name. Nevertheless, both datasets share literals with some commonality. By exploring this commonality we are able to define a virtual link set between both datasets. Note that the Swiss cantons' resource IRIs in both datasets are not the same – otherwise defining a virtual link set would be simpler—i.e. a simple link set, see Definition 4. In the next subsections, we incrementally demonstrate with a running example how to model a complex link set (see Definition 6) with VoID and VoIDext terms. Table 1 shows other datasets and SPARQL endpoints considered in our examples in this article.

**Definition 4 (simple link set).** *A simple link set must be either a link set that does not target another link set (i.e. it has exactly one link predicate—Definition 5) or a set with exactly the same shared instances of the same type (i.e. class expression) in both datasets.*

[4] https://lindas-data.ch.

**Table 1.** SPARQL endpoints considered in this article.

| RDF dataset | SPARQL endpoint |
|---|---|
| DBpedia [23] | http://dbpedia.org/sparql |
| LINDAS (see Footnote 4) | https://lindas-data.ch/sparql |
| OMA [3] | https://sparql.omabrowser.org/sparql |
| UniProtKB [30] | https://sparql.uniprot.org/sparql |
| Bgee [5] | http://biosoda.expasy.org/rdf4j-server/repositories/bgeelight |
| EBI RDF [22] | https://www.ebi.ac.uk/rdf/services/sparql |

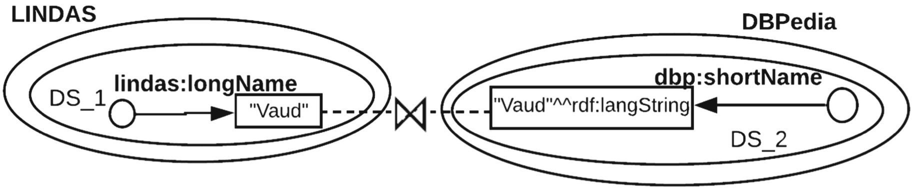

**Fig. 1.** An example of a virtual link between the LINDAS and DBpedia datasets where the $DS_1$ and $DS_2$ datasets are subsets of them, respectively. Circles: different resource IRIs; rectangles: literals; ⋈: virtual link; and edges: RDF predicates.

In practice, a simple link set allows us to model virtual links either between the subjects of two RDF triples in different datasets where their predicate is *rdf:type* with the same object or between link predicate assertions and *rdf:type* triples. Due to the space constraints, the patterns to model simple link sets with VoIDext are available in Supplementary Material Section 4 [13].

**Definition 5 (link predicate).** *According to the VoID specification, a link predicate is the RDF property of the triples in a void:Linkset [2].*

**Definition 6 (complex link set).** *It is a complex virtual link set. A complex link set is composed of exactly two link sets xor two shared instance sets (see Definition 7) where xor is the exclusive or.*

**Definition 7 (shared instance set).** *A shared instance set between exactly two datasets. For example, two datasets that contain the same OWL/RDFS class instances.*

### 3.1 Patterns to *partially* model complex link sets with VoID terms

Since our main goal is to facilitate the writing of federated queries by providing metadata of how the target datasets can be joined, let us suppose that we want to know how to relate Swiss cantons in LINDAS and DBpedia datasets as shown in Fig. 1. In other words, we want to find out the necessary and sufficient graph pattern in the context of Swiss cantons in each dataset to be able to relate and join them. Further triple patterns such as attributes (e.g. canton's population, cities, acronym) depend on the specificity of the requested information what goes beyond the task of joining the two datasets. Let us further assume a SPARQL user without any previous knowledge about these datasets. A possible workflow for this user to find out how to relate LINDAS and DBpedia in terms of Swiss cantons is described as follows:

(1) the user has to dig up the data schema and documentation, if any, looking for the abstract entity "Swiss canton". This task has to be done for both datasets.
(2) if (s)he is lucky, a concept is explicitly defined in the data schema. This is the case of the LINDAS dataset that contains the class *lindas:Canton*—prefixes such as *lindas:* are defined in Table 2. Otherwise the user has to initiate a fastidious quest for assertions and terms that can be used for modeling Swiss canton data. This is the situation of DBpedia where instances are defined as a Swiss canton by assigning the *dbrc:Cantons_of_Switzerland* instance of the *skos:Concept* to the *dct:subject* property such as the following triple (*dbr:Vaud, dct:subject, dbrc:Cantons_of_Switzerland*).
(3) The user has now to browse the RDF graph. For example, by performing additional queries, to be sure that the assertions to the *lindas:Canton* instances can be used as join points with assertions related to Swiss canton instances in DBpedia. Otherwise, the user has to repeat the previous steps.
(4) If data transformations are required because of data and semantic heterogeneities between the datasets, the user has to define data mappings to be able to effectively perform a federated conjunctive query.

**Table 2.** In this article, we assume the namespace prefix bindings in this table.

| Prefix | Namespace Internationalized Resource Identifier (IRI) |
|---|---|
| rdfs: | http://www.w3.org/2000/01/rdf-schema# |
| rdf: | http://www.w3.org/1999/02/22-rdf-syntax-ns# |
| up: | http://purl.uniprot.org/core/ |
| cco: | http://rdf.ebi.ac.uk/terms/chembl# |
| chembl: | http://rdf.ebi.ac.uk/resource/chembl/molecule/ |
| ex: | http://example.org/voidext# |
| dbo: | http://dbpedia.org/ontology/ |
| skos: | http://www.w3.org/2004/02/skos/core# |
| dbr: | http://dbpedia.org/resource/ |
| dbrc: | http://dbpedia.org/resource/Category: |
| dbp: | http://dbpedia.org/property/ |
| lindas: | https://gont.ch/ |
| dcterms: | http://purl.org/dc/terms/ |
| biopax: | http://www.biopax.org/release/biopax-level3.owl# |
| lscr: | http://purl.org/lscr# |
| void: | http://rdfs.org/ns/void# |
| voidext: | http://purl.org/query/voidext# |
| bioquery: | http://purl.org/query/bioquery# |

```
SELECT * WHERE {
SERVICE <http://dbpedia.org/sparql>{
?dbp_inst dct:subject dbrc:Cantons_of_Switzerland.
?dbp_inst dbp:shortName ?dbp_name.
BIND(IF(STR(?dbp_name)="Grisons", "Graubünden / Grigioni / Grischun",
     IF(STR(?dbp_name)="Geneva", "Genève",
     IF(STR(?dbp_name)="Lucerne", "Luzern",
     IF(STR(?dbp_name)="Valais", "Valais / Wallis",
     IF(STR(?dbp_name)="Bern", "Bern / Berne",
     IF(STR(?dbp_name)="Fribourg", "Fribourg / Freiburg",
         STR(?dbp_name) )))))) AS ?lindas_name)}
SERVICE <https://lindas-data.ch/sparql>{
?lindas_inst a  lindas:Canton;
                lindas:longName ?lindas_name.}}
```

**Listing 1.** The initial basic graph patterns represented as a SPARQL federated query to perform join queries between Swiss cantons in the LINDAS and DBpedia datasets. Table 2 contains the IRI prefixes.

Finally, based on that workflow, a **SPARQL user** can draft the minimum set of triple patterns and data transformations to perform the virtual links concerning Swiss cantons between both datasets. This draft is represented as the SPARQL query in Listing 1. The link set built by intersecting the resources (i.e. the values of *lidas:longName* and *dbp:shortName* properties) can then be partially modelled with VoID terms. This enables **other SPARQL users or systems** to reuse this link set knowledge to write specialized queries over the two datasets starting from the Swiss canton context. In doing so, the second user avoid the fastidious, complex and time-consuming task of finding this link set. In addition, to the best of our knowledge there is no system capable of precisely establishing this virtual link set automatically because of the complexity and heterogeneities to be solved.

Listings 2 and 3 depict two different ways named $VL_{m1}$ and $VL_{m2}$ to model the virtual link set with VoID. Note that the examples of RDF graph patterns in this section are defined with the RDF 1.1 Turtle language[5]. On the one hand, $VL_{m1}$ states that a given $LS_1$ link set targets another $LS_2$ link set that targets $LS_1$ back. On the other, $VL_{m2}$ only states datasets as link set targets. By using the $VL_{m1}$ model in Listing 2, the $DS_2$ instance asserts the $DS_1$ instance to its *void:objectsTarget* property. As a reminder, the *void:objectsTarget* value is the dataset describing the objects of the triples contained in the link set, in our example, the objects of *dbp:shortName*. This dataset must contain only the relevant triples to describe the virtual link set. In our example in Listing 2, we define the $DS_1$ dataset (i.e. a subset of LINDAS) as being also a *void:Linkset* that contains triples with the *lindas:longName* predicate. By using the $VL_{m2}$ model in Listing 3, the objects' target dataset of the *dbp:shortName* link predicate is not a *void:Linkset* but a *void:Dataset* (i.e. superclass of *void:Linkset*). $DS_2$ in Listing 3 also contains triples with the *lindas:longName* predicate, however, this predicate is defined as part of a subset and partition of $DS_2$ by using the *void:propertyPartition* and *void:property* terms. Note that solely one

[5] https://www.w3.org/TR/turtle/.

*void:propertyPartition* should be directly assigned to $DS_2$ dataset, otherwise we are not able to know which predicate should be considered when stating the virtual links.

```
#DS_1 is a subset of LINDAS.
ex:LINDAS_DBPEDIA_SWISSCANTON
   rdf:type void:Linkset ;
   void:linkPredicate lindas:longName ;
   void:objectsTarget
         ex:DBPEDIA_LINDAS_SWISSCANTON ;
   void:subset _:DOMAIN_SET0 .
_:DOMAIN_SET0 void:propertyPartition _:b0 ;
   void:classPartition _:b1 .
_:b0 void:property rdfs:domain .
_:b1 void:class lindas:Canton .
                              . . .
```

```
#DS_2 is a subset of DBpedia.
ex:DBPEDIA_LINDAS_SWISSCANTON
   rdf:type void:Linkset ;
   void:linkPredicate dbp:shortName ;
   void:objectsTarget
         ex:LINDAS_DBPEDIA_SWISSCANTON ;
   void:subset _:DOMAIN_SET1 .
_:RANGE_SET1 void:propertyPartition _:b2 ;
   void:classPartition _:b3 .
_:b2 void:property rdfs:range .
_:b3 void:class rdf:langString .
                              . . .
```

**Listing 2.** Patterns to model a complex virtual link set between the LINDAS Linked Data service and DBpedia relying on link sets as targets (e.g. *void:objectsTarget*). As a reminder, *void:Linkset* is a subclass of *void:Dataset.*

```
#DS_2 is a subset of DBpedia.
ex:DBPEDIA_LINDAS_SWISSCANTON rdf:type void:Linkset ;
   void:linkPredicate dbp:shortName ;
   void:objectsTarget _:b4 .
_:b4 rdf:type void:Dataset ;
   void:propertyPartition _:LINDAS_PROPERTY ;
   void:subset _:RANGE_SET2 .
_:LINDAS_PROPERTY void:property lindas:longName .
_:RANGE_SET2 void:propertyPartition _:b5.
_:RANGE_SET2 void:classPartition _:b6 .
   _:b5 void:property rdfs:range .      #it restricts lindas:longName
   _:b6 void:class rdfs:Literal .       #range to rdfs:Literal.
                                 . . .
```

**Listing 3.** Patterns to model a complex virtual link set between the LINDAS Linked Data service and DBpedia relying on link sets and property partitions as targets (e.g. *void:objectsTarget*). For the sake of simplicity, only the link set in DBpedia is depicted because the link set in LINDAS containing the *lindas:longName* link predicate is similarly modelled as the one in DBpedia.

Yet, we also need to describe further information about the virtual link set such as the domain and range of the link predicates (e.g. *lindas:longName* and *dbp:shortName*). This information is used to restrict which resource type must be considered for a given triple that contains the link predicate (e.g. *lindas:longName rdfs:domain lindas:Canton*). By having this information in advance when writing and executing a federated query, we reduce the number of triples to match, if there are statements of the same predicate but with resources of other types. For example, the *lindas:longName* property is asserted to instances of *lindas:MunicipalityVersion*, *lindas:Canton* or *lindas:DistrictEntityVersion.* However, for the context of this virtual link set only *lindas:Canton* instances need to be considered. To restrict the resource types of a given link predicate with VoID, we can state subsets and partitions to a *void:Linkset* such as exemplified in Listing 2 by using $VL_{m1}$.

Note that for each link predicate's domain/range, we have to create one new subset to be sure that the class partitions of the subset correspond to the domain or range of the link predicate. In addition, if there are multiple resource types to be considered as the domain of a link predicate, we can state multiple class partitions to express the union of types—i.e. classes. Or, we can explicitly define it by using the OWL 2 Description Logic (DL) term *owl:unionOf* and related patterns to express class union. To express class intersection or other class expressions, we can rely on OWL 2 DL terms and state these class expressions as class partitions of the subset. Similarly, we can model the domain and range of predicates related to virtual links with $VL_{m2}$ as described in Listing 3. For the sake of simplicity, we do not depict all predicate domains/ranges in Listings 2 and 3. However, there are several limitations when only considering VoID terms to model complex link sets:

(1) **Multiple representations**. The VoID vocabulary and documentation due to the lack of constraints and high generalization imply several ways to model virtual link sets such as $VL_{m1}$ and $VL_{m2}$ graph patterns to represent a complex link set. In addition, there are various ways to define the link predicate's domain and range. For example, class expressions can be defined by using either OWL 2 DL terms to express the union of classes or multiple class partitions (i.e. *void:classPartition* assertions), or by combining both of them. This multitude of graph patterns allowed by VoID makes interoperability more complex because we do not previously know how the virtual link set is modelled. Consequently, it requires to build complex parsers and queries to retrieve the virtual link set metadata.

(2) **Ambiguity**. With VoID, we cannot easily distinguish if a link set or dataset is being instantiated to define a virtual link set. For example, we do not know explicitly if two link sets compose a complex link set. Moreover, the use of property/class partitions to define domains and ranges of link predicates can be mixed with *void:class* assertions that are not part of a domain/range definition. Subsets can also be arbitrarily stated to any link set or dataset what increases the ambiguity to know if a given subset is actually part of a complex link set definition or not. With $VL_{m1}$ and $VL_{m2}$ models strictly based on VoID, we cannot explicitly state that the intersections between two link sets occur by matching the subjects-objects, objects-objects or subjects-subjects of link predicates in different link sets. Nevertheless, this information can be derived from the *void:objectsTarget* and *void:subjectsTarget* assertions, if any.

(3) **Description Logic (DL) compliance** [4]. By stating the domain and range of link predicates with class expressions based on OWL 2 DL (e.g. a range composed of multiple types/classes), we can take advantage of existing DL-parser and reasoner tools to infer instance types (e.g. http://owlcs.github.io/owlapi/). However, since we can mix DL-based class expressions

with *void:class* assertions, the resulting range/domain expressions are non-compliant with DL.

(4) **Verbosity**. The use of class and property partition partners considerably increases the number of triples to state for representing virtual links. This also increases the complexity of writing of queries to retrieve the virtual link set metadata.

(5) **Resource mapping**. VoID does not provide any explicit term and recommendation to state resource mappings. By doing so, we mitigate or even solve heterogeneities when matching resources with some commonality in different datasets.

In the next subsection, we show how to solve these issues with VoIDext terms and patterns.

### 3.2 Patterns to *fully* model complex link sets with VoIDext

To address the issues of modelling virtual link sets solely with VoID, we propose new terms and patterns in VoIDext. Listing 4 illustrates the main VoIDext terms (see terms with *voidext:* prefix) and design patterns to model complex link sets. To assert the range and domain of predicates with VoIDext, we can directly assign the *voidext:linkPredicateRange* and *voidext:linkPredicateDomain* properties to a link set, respectively (see Definitions 8 and 9). Complex link predicates' domains and ranges (e.g. multiple types—union/intersection of classes) must be stated as class expressions by using OWL 2 DL terms (e.g. *owl:unionOf*). To avoid ambiguities when interpreting link sets (i.e. a simple set *versus* a complex one), we can explicitly state that two link sets are indeed part of a complex link set. To do so, we must assign exactly two link sets to a complex link set with the *voidext:intersectAt* property (see Definition 10). In a complex link set, a link set must be connected to another link set by stating either *void:objectsTarget* or *void:subjectsTarget* properties. This allow us to precisely know where the intersection between RDF triples with predicates in different datasets occurs, in other words, the matched RDF resource nodes: object-object, subject-subject, and subject-object. For example, in Listing 4 with *void:objectsTarget* property, we state that the *lindas:longName* predicate's objects in LINDAS match the objects of the *dbp:shortName* link predicate in DBpedia, and *vice-versa.* To explicitly state the intersection type (e.g. object-object), we can assert the *voidext:intersectionType* property (see Definition 11) to a complex link set as shown in Listing 4.

**Definition 8 (link predicate range).** *The link predicate's object type (i.e. class expression or literals), if any. Moreover, a link set (Definition 1) that is not part of a complex link set (see Definition 6) and connects two datasets through the link predicate's object must specify the link predicate range. Indeed, this object matches a second resource in another dataset. Therefore, the type of this second resource is asserted as the link predicate range.*

**Definition 9 (link predicate domain).** *The link predicate's subject type (i.e. class expression), if any.*

**Definition 10 (intersects at).** *It specifies the intersection of either exactly two shared instance sets (see Definition 7) or two link sets, that compose a complex link set.*

**Definition 11 (intersection type).** *It specifies the intersection type between two RDF triples in different datasets. In other words, if the intersection occurs at the subject xor the object node of a link predicate.*

Based on Definition 6, the *voidext:ComplexLinkSet* OWL class is defined with the following DL expression, IRI prefixes are ignored to improve readability:

$$\begin{aligned} ComplexLinkSet \equiv \neg SimpleLinkSet \sqcap \\ ((\forall intersectAt.Linkset \sqcap \ = 2\, intersectAt) \sqcup \\ (\forall intersectAt.SharedInstanceSet \sqcap \ = 2\, intersectAt)) \end{aligned}$$

As a reminder, a *void:Dataset* is "a set of RDF triples that is published, maintained or aggregated by a single provider"[6]. However, a complex link set is composed of resources from two different datasets (e.g. two link predicates). Therefore, we define *voidext:ComplexLinkSet* as being disjoint with *void:Dataset* class. Consequently, a complex link set is not a *void:Dataset* and properties such as *void:propertyPartition* cannot be assigned to it.

To address data heterogeneities, we can implement semantic relaxation by stating the *voidext:resourceMapping* property (Definition 12) with a literal text based on SPARQL language. In Listing 4, $DS_2$ states a mapping in line 7 that converts *dbp:shortName* language-tagged string values into simple literals and maps the values to a corresponding one in LINDAS dataset. Thus, since this mapping is defined using SPARQL language, it can be directly used to build a SPARQL 1.1 federated query to perform the interlinks between datasets. In Listing 4, *voidext:recommendedMapping* (Definition 13) assigns the LINDAS $DS_1$ link set as the one containing the mapping to be considered when interlinking with DBpedia in the context of Swiss cantons.

**Definition 12 (resource mapping).** *It specifies the mapping function ($f_m$) to preprocess a resource (i.e. IRI or literal) in a source dataset in order to match another resource in the target dataset. The resource preprocessing (i.e. mapping) must be defined with the SPARQL language by mainly using SPARQL built-ins for assignments (e.g. BIND), and expression and testing values (e.g. IF and FILTER). The BIND built-in is used to assign the output of $f_m$, if any.*

**Definition 13 (recommended resource mapping).** *It specifies one recommended mapping function, if more than one mapping is defined in the different sets that are part of a complex link set.*

[6] http://vocab.deri.ie/void#Dataset.

```
#DS_1 is a subset of LINDAS.
ex:LINDAS_DBPEDIA_SWISSCANTON
   rdf:type void:Linkset ;
   void:linkPredicate lindas:longName ;
   voidext:linkPredicateRange
          rdfs:Literal ;
   voidext:linkPredicateDomain
          lindas:Canton ;
   voidext:isSubsetOf ex:LINDAS;
   void:objectsTarget
  ex:DBPEDIA_LINDAS_SWISSCANTON ;
   voidext:resourceMapping
      “‘?x a <https://gont.ch/Canton>.
             . . . ’” ;
                  . . .
```

```
#DS_2 is a subset of DBpedia.
ex:DBPEDIA_LINDAS_SWISSCANTON
   rdf:type void:Linkset ;
   void:linkPredicate dbp:shortName ;
   voidext:isSubsetOf ex:DBPEDIA ;
   void:objectsTarget ex:LINDAS_DBPEDIA_SWISSCANTON ;
   voidext:resourceMapping
   “‘?dbpedia_place dcterms:subject dbrc:Cantons_of_Switzerland.
   ?dbpedia_place dbp:shortName ?c.
   BIND(
    IF(STR(?c)=“Grisons”, “Graubünden / Grigioni / Grischun”,
    IF(STR(?c)=“Geneva”, “Genève”,
    IF(STR(?c)=“Lucerne”, “Luzern”,
    IF(STR(?c)=“Valais”, “Valais / Wallis”,
    IF(STR(?c)=“Bern”, “Bern / Berne”,
    IF(STR(?c)=“Fribourg”, “Fribourg / Freiburg”, STR(?c)
        )))))) as ?lindas_objects)’”; . . .
```

```
ex:DBPEDIA_LINDAS_SWISSCANTON_VL rdf:type voidext:ComplexLinkSet ;
   voidext:intersectionType voidext:OBJECT_OBJECT ;
   dcterms:issued “2019-06-30”^^xsd:date ;
   rdfs:label “A virtual link set for cantons in both DBpedia and LINDAS Swiss government datasets.” ;
   voidext:intersectAt ex:LINDAS_DBPEDIA_SWISSCANTON ;
   voidext:intersectAt ex:DBPEDIA_LINDAS_SWISSCANTON ;
   voidext:recommendedMapping ex:LINDAS_DBPEDIA_SWISSCANTON ;        . . .
```

**Listing 4.** VoIDext-based patterns to model a complex virtual link set between the LINDAS Linked Data service and DBpedia relying on link sets as targets. Dashed underlined: one of the two can be chosen as the *voidext:recommendedMapping*; and fully underlined: predicates used to connect the datasets by *void:objectsTarget* predicates.

To exemplify a complex link set composed of shared instance sets (Definition 7), let us consider the UniProt and European Bioinformatics Institute (EBI) RDF datasets (see Table 1). EBI and UniProt RDF data stores use different instance IRIs and classes to represent the organism species, and in a more general way, the taxonomic lineage for organisms. To exemplify this, let us consider the *<http://identifiers.org/taxonomy/9606>* instance of *biopax:BioSource* and the *<http://purl.uniprot.org/taxonomy/9606>* instance of *up:Taxon* in EBI and UniProt datasets, respectively. Although these instances are not exactly the same (i.e. distinct IRIs, property sets, and contexts), they refer to the same organism species at some extent, namely *homo sapiens*—human. By applying a semantic relaxation, we can state a virtual link between these two instances. To establish this link, we need to define a resource mapping function (i.e. $f_m(r)$) either to the EBI or UniProt species-related instances—either $f_m($*<http://identifiers.org/taxonomy/9606>*$) \equiv$ *<http://purl.uniprot.org/taxonomy/9606>* or $f_m($*<http://purl.uniprot.org/taxonomy/9606>*$) \equiv$ *<http://identifiers.org/taxonomy/9606>*. Listing 5 depicts how this complex link set is modelled with VoIDext-based patterns. Note that it is not possible to define a shared instance set by only using VoID terms because there is no link predicate (Definition 5) associated with the interlinks that are different from *rdf:type*. To address this issue, we can assign a shared instance type (Definition 14) with the *voidext:sharedInstanceType* property for each *voidext:SharedInstanceSet* instance (Definition 7). Other examples of complex link sets are available in [13] and [14].

**Definition 14 (shared instance type).** *The type (i.e. class) of the shared instances in a given dataset. Shared instances imply equivalent or similar instance IRIs that belong to different datasets.*

## 4 VoIDext Benefits and Discussions

VoID instances ("assertion box"—ABox) are fully backward compatible with the VoIDext schema since we mainly add new terms. The only change performed in the VoID "terminological box" (TBox) concerns the *void:target*[7] property domain. In VoIDext, this domain is the union of the *void:Linkset* and *voidext:SharedInstanceSet* classes instead of solely *void:Linkset*, as stated in VoID. We did this to avoid the replication of a similar property to state target datasets to shared instance sets. Despite this modification, assertions of *void:target* based on VoID remain compatible with VoIDext.

### 4.1 Retrieving Virtual Link Sets

Once the virtual links are modelled with VoIDext as discussed in Subsect. 3.2 and Supplementary Material Section 4 [13], there may exist at most four kinds of virtual link sets as follows: (i) a *voidext:ComplexLinkSet* composed of *void:Linksets*—e.g. see Listing 4; (ii) a *voidext:ComplexLinkSet* composed of *voidext:SharedInstanceSets*—e.g. see Listing 5; (iii) a *void:Linkset* that is also

**A) a subset of EBI**

```
bioquery:EBI_UNIPROT_10
        rdf:type voidext:SharedInstanceSet ;
   voidext:isSubsetOf bioquery:EBI ;
   voidext:resourceMapping
   “‘?IRI_EBI a biopax:BioSource.
BIND(IRI(CONCAT(“http://purl.uniprot.org/taxonomy/”
   , STRAFTER(STR(?IRI_EBI),
   “http://identifiers.org/taxonomy/”))) as ?IRI_UNIPROT)
   FILTER(STRSTARTS(STR(?IRI_EBI),
   “http://identifiers.org/taxonomy/”))’”;
   voidext:sharedInstanceType biopax:BioSource ;  ...
```

**B) a subset of UniProt**

```
bioquery:EBI_UNIPROT_11
        rdf:type voidext:SharedInstanceSet ;
voidext:isSubsetOf bioquery:UNIPROT ;
voidext:resourceMapping
“‘?IRI_UNIPROT a up:Taxon.
BIND(IRI(CONCAT(“http://identifiers.org/taxonomy/”
, STRAFTER( STR(?IRI_UNIPROT),
“http://purl.uniprot.org/taxonomy/”) ) ) as ?IRI_EBI)
FILTER(STRSTARTS(STR(?IRI_UNIPROT),
“http://purl.uniprot.org/taxonomy/”))’” ;
voidext:sharedInstanceType up:Taxon ;  ...
```

```
bioquery:EBI_UNIPROT_12 rdf:type voidext:ComplexLinkSet ;
   voidext:intersectAt bioquery:EBI_UNIPROT_11 ;
   voidext:intersectAt bioquery:EBI_UNIPROT_10 ;
   voidext:recommendedMapping bioquery:EBI_UNIPROT_10 ;
rdfs:label “Links between EBI and UniProt considering shared similar instances of organism taxonomy”@en;  ...
```

**Listing 5.** VoIDext-based patterns to model a complex virtual link set between EBI and UniProt datasets modelled with shared instance sets (see fully underlined assertions). Dashed underlined: one of the two can be chosen as the *voidext:recommendedMapping*.

[7] https://www.w3.org/TR/void/#target.

a *voidext:SimpleLinkSet*; and (iv) a *voidext:SharedInstanceSet* that is also a *voidext:SimpleLinkSet*. Due to the page limit, the types (iii) and (iv) are exemplified in Fig. 7 and Listing 1.3 in Supplementary Material [13]. For each kind of virtual link set, a SPARQL query template to retrieve the essential information is asserted as an annotation of the *voidext:ComplexLinkSet* and *voidext:SimpleLinkSet* sub-classes of *voidext:VirtualLinkSet*. These annotations are done by asserting the *voidext:queryLinkset* and *voidext:querySharedInstanceSet* properties. Therefore, to retrieve virtual link sets of type (i) and (iii), we can execute the SPARQL queries assigned with *voidext:queryLinkset* to the *voidext:ComplexLinkSet* and *voidext:SimpleLinkSet* classes, respectively. Similarly, to retrieve virtual link sets of type (ii) and (iv), we can execute the SPARQL queries assigned with *voidext:querySharedInstanceSet* to the *voidext:ComplexLinkSet* and *voidext:SimpleLinkSet* classes, respectively. These queries are described in [14].

### 4.2 Writing a Federated SPARQL Query with VoIDext Metadata

To illustrate how VoIDext can facilitate the writing of federated SPARQL queries, let us consider that a SPARQL user wants to perform the $Q_f$ query against the EBI dataset: *"Show me all assays in rodents for the drug Gleevec (i.e. CHEMBL941 identifier)"*. One possible way to write this query is to consider another dataset that contains organismal taxonomy information about rodents such as the UniProt dataset. To be able to write this federated conjunctive query over the EBI and UniProt datasets, the SPARQL user has to find out how to relate these datasets. To do so, the SPARQL user can query the metadata about virtual link sets modelled with VoIDext—see the template queries for this purpose in the VoIDext specification [14] and further details in Subsect. 4.1. Table 3 exemplifies a possible outcome of these queries containing virtual link set descriptions.

Based on the description of virtual link sets, users can select the link set that best fit their needs to write $Q_f$ with SPARQL 1.1. In this example, a user can choose the complex link set *bioquery:EBI_UNIPROT_12* about organism taxonomy depicted in Listing 5 and defined as the query result over the

```
SELECT ?assay WHERE {
 ?activity a cco:Activity ;
          cco:hasMolecule chembl:CHEMBL941 ;
          cco:hasAssay ?assay .
 ?assay cco:taxonomy  ?IRI_EBI .
 ?IRI_EBI a <http://www.biopax.org/release/biopax-level3.owl#BioSource> .
 BIND(IRI(CONCAT("http://purl.uniprot.org/taxonomy/",
  STRAFTER(STR(?IRI_EBI),"http://identifiers.org/taxonomy/"))) as ?IRI_UNIPROT)
 FILTER(STRSTARTS(STR(?IRI_EBI),"http://identifiers.org/taxonomy/"))
 SERVICE<https://sparql.uniprot.org/sparql>{
  ?IRI_UNIPROT a up:Taxon.
  ?IRI_UNIPROT rdfs:subClassOf ?taxon2  .
  ?taxon2 up:otherName "rodents". } }
```

**Listing 6.** A federated query between EBI and UniProt datasets to retrieve assays in rodents for the drug Gleevec (i.e. CHEMBL941). Table 2 contains the IRI prefixes.

VoIDext metadata in Table 3—i.e. the $T_1$ tuple. By considering $T_1$ in Table 3, the SPARQL user can draft $Q_f$ starting with the interlink between EBI and UniProt as shown in bold in Listing 6. The user can now continue the writing of $Q_f$ by solely focusing on each dataset separately—i.e. the non-bold part of the $Q_f$ SPARQL query. Therefore, the fastidious process of finding out interlinks and data transformations between EBI and UniProt to perform a federated query is mitigated with the virtual link sets defined using the VoIDext vocabulary. The query in Listing 6 can be executed in the EBI SPARQL endpoint (see Table 1). Further examples of SPARQL federated queries that were written based on VoIDext metadata are available as part of an application described in Subsect. 4.4.

**Table 3.** The results of querying complex link sets composed of two shared instance sets between EBI and UniProt.

| |
|---|
| **Outcome:** A set of tuples $T = (V_L, ds_1, ds_2, I_{t1}, I_{t2}, A_{ds_1}, A_{ds_2}, f_m)$ where $V_L$ is the virtual link set IRI; $ds_1$ and $ds_2$ are the names of the datasets that contain the instance IRIs; $I_{t1}$ is the type (DL-class expression) of the instance in $ds_1$; $I_{t2}$ is the type (DL-class expression) of the instance in $ds_2$; $A_{ds_1}$ and $A_{ds_2}$ are the access methods such as SPARQL endpoints to the $ds_1$ and $ds_2$ datasets, respectively; $f_m$ is the recommended resource mapping procedure, if any, to be applied to instance IRIs of $I_{t1}$ xor $I_{t2}$ types, where xor is the exclusive or. |
| **Example:**<br>$T_1$ = (bioquery:EBI_UNIPROT_12, "Linked Open Data platform for EBI data.","The Universal Protein Resource (UniProt)", biopax:BioSource, up:Taxon, <https://www.ebi.ac.uk/rdf/services/sparql>, <https://sparql.uniprot.org/sparql/>, $f_m^3(i)$)<br>where $i$ is any instance of *biopax:BioSource* type and $f_m^3(i) \equiv$<br>"'?IRI_EBI a <http://www.biopax.org/release/biopax-level3.owl#BioSource>.<br>BIND(IRI(CONCAT("http://purl.uniprot.org/taxonomy/", STRAFTER(<br>STR(?IRI_EBI), "http://identifiers.org/taxonomy/"))) as ?IRI_UNIPROT)<br>FILTER(STRSTARTS(STR(?IRI_EBI), "http://identifiers.org/taxonomy/"))"' |

## 4.3 Virtual Link Set Maintenance

Although, to manage the virtual link set evolution is out of the scope of this article, we recommend to annotate the link sets with the issued and modified dates such as depicted in Listing 4. This date information helps with the maintenance of virtual link sets. For example, let us suppose the release of a new version of the DBpedia in August, 2019. By checking the difference between the DBpedia new release date and the complex link set issued/modified date (e.g. June 2019, see Listing 4), it might indicate a possible decrease in the virtual link set performance, or even, invalidity of the interlinks due to the fact of being outdated. In addition, for each virtual link set, we can state the performance in terms of precision, recall, true positives, and so on by asserting the *voidext:hasPerformanceMeasure* property. The range of this property is *mex-perf:PerformanceMeasure*[8]. Thus, we can rely on the Mex-perf ontology (see

[8] http://mex.aksw.org/mex-perf.

Footnote 8) to describe the virtual link set performances. The complex link set example about Swiss cantons in Listing 4 has a precision and recall of 100%. In this example, for every Swiss canton in LINDAS exists a corresponding one in DBpedia. Therefore, if this performance is deteriorated after the new release of one of the datasets involved, we should review this virtual link set. Further use cases are exemplified in Supplementary Material Section 5 available in [13].

### 4.4 Benefits and a SIB Swiss Institute of Bioinformatics' Application

***Easing the Task of Writing SPARQL 1.1 Federated Queries.*** The formal description of virtual link sets among multiple RDF datasets on the Web facilitates the manually or (semi-)automatically writing of federated queries. This is because once the virtual link sets are defined between datasets with VoIDext, we can interlink different RDF datasets without requiring to mine this information again from the various ABoxes and TBoxes (including documentation, if any). The mining task becomes more and more complex and fastidious if the TBox is incomplete or missing when comparing with the ABox statements, for example, a triple predicate that is not defined in the TBox.

***Applying Semantic Relaxation Rather than Semantic Reconciliation.*** The *virtual link* statements between datasets are more focused on the meaning of interlinking RDF graph nodes rather than the semantics of each node in the different datasets and knowledge domains. For example, let us consider the *virtual link* illustrated in Fig. 1. When considering solely the LINDAS dataset, the *lindas:longName* is a *rdf:Property* labelled as a "District name or official municipality name". In DBpedia, *dbp:shortName* is a *rdf:Property* labelled as "short name" and in principle can be applied to any instance. Hence, it is not restricted to district names. In addition, one property is about long names while the other one is about short names. However, they state similar literals in the context of Swiss cantons as discussed in Sect. 3. Therefore, although these properties are semantically different (hard to reconcile), we can still ignore heterogeneities for the sake of interlinking DBpedia and LINDAS.

***Facilitating Knowledge Discovery.*** As noticed in [1], yet there are many challenges to address in the semantic web such as the previous knowledge of the existing RDF datasets and how to combine them to process a query. VoIDext mitigates these issues because RDF publishers (including third-party ones) are able to provide virtual link sets which explicitly describe how heterogeneous datasets of distinct domains are related. Without knowing these links, to potentially extract new knowledge that combines these datasets is harder or not even possible. The virtual link sets stated with VoIDext terms provide sufficient machine-readable information to relate the datasets. Nonetheless, the automatic generation of these link sets is out of the scope of this article.

***A SIB Swiss Institute of Bioinformatics application.*** We applied the VoIDext vocabulary in the context of a real case application mainly involving three in production life-sciences datasets available on the Web, namely UniProtKB, OMA and Bgee RDF stores—see SPARQL endpoints in Table 1. The RDF serialization of virtual link sets among these three databases is available in [13] and it can be queried via the SPARQL endpoint in [12] with query templates defined in [14] as described in Subsect. 4.1. Based on these *virtual links*, a set of more than twelve specialized federated query templates over these data stores was defined and are available at https://github.com/biosoda/bioquery/tree/master/Queries. These templates are also available through a template-based search engine, see http://biosoda.expasy.org. Moreover, as an example of facilitating knowledge discovery, we can mention the virtual link sets between OMA and Bgee. These two distinct biological knowledge domains when combined enable to predict gene expression conservation for orthologous genes (i.e. corresponding genes in different species). Finally, new virtual link sets are being created to support other biological databases in the context of SIB—https://sib.swiss.

## 5 Conclusion

We successfully extended the VoID vocabulary (i.e. VoIDext) to be able to formally describe *virtual links* and we provided a set of SPARQL query templates to retrieve them. To do so, we applied an agile methodology based on the SAMOD approach. We described the benefits of defining *virtual links* with VoIDext RDF schema, notably to facilitate the writing of federated queries and knowledge discovery. In addition, with *virtual links* we can enable interoperability among different knowledge domains without imposing any changes in the original RDF datasets. In the future, we intend to use VoIDext to enhance keyword-search engines over multiple distributed and independent RDF datasets. We also envisage to propose tools to semi-automatically create VoIDext virtual link statements between RDF datasets. We believe these tools can leverage the adoption of VoIDext by other communities besides SIB, Quest for Orthologs consortium (https://questfororthologs.org), and Linked Building Data Community (https://www.w3.org/community/lbd) where the authors are involved. We also encourage other communities to collaborate on open issues in the public GitHub of VoIDext in [13] to refine this vocabulary for other use cases that have not been contemplated during this work. Finally, to support virtual link evolution, we aim to develop a tool to automatically detect broken virtual links because of either data schema changes or radical modifications of instances' IRIs and property assertions.

**Acknowledgements.** This work was funded by the Swiss National Research Programme 75 "Big Data" (Grant 167149) and a Swiss National Science Foundation Professorship grant to CD (Grant 150654).

## References

1. Acosta, M., Hartig, O., Sequeda, J.: Federated RDF query processing. In: Sakr, S., Zomaya, A. (eds.) Encyclopedia of Big Data Technologies, pp. 1–10. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-63962-8
2. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing linked datasets. In: Proceedings of the Linked Data on the Web Workshop (LDOW2009), Madrid, Spain, 20 April 2009. CEUR Workshop Proceedings (2009)
3. Altenhoff, A.M., Glover, N.M., Train, C.M., et al.: The OMA orthology database in 2018: retrieving evolutionary relationships among all domains of life through richer web and programmatic interfaces. Nucl. Acids Res. **46**(D1), D477–D485 (2018)
4. Baader, F., Calvanese, D., McGuinness, D., Patel-Schneider, P., Nardi, D.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, Cambridge (2003)
5. Bgee team: Bgee data sources (2019). https://bgee.org/?page=source. Accessed 25 Aug 2019
6. Brunner, U., Stockinger, K.: Entity matching on unstructured data: an active learning approach. In: 6th Swiss Conference on Data Science, Bern, 2019. IEEE (2019)
7. buildingSMART: IFC overview summary. http://www.buildingsmart-tech.org/specifications/ifc-overview. Accessed 25 Aug 2019
8. Crotti Junior, A., Debruyne, C., Brennan, R., O'Sullivan, D.: An evaluation of uplift mapping languages. Int. J. Web Inf. Syst. **13**(4), 405–424 (2017)
9. David, J., Euzenat, J., Scharffe, F., Trojahn dos Santos, C.: The alignment API 4.0. Semant. Web **2**(1), 3–10 (2011)
10. De Meester, B., Maroy, W., Dimou, A., Verborgh, R., Mannens, E.: Declarative data transformations for linked data generation: the case of DBpedia. In: Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., Hartig, O. (eds.) ESWC 2017. LNCS, vol. 10250, pp. 33–48. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58451-5_3
11. Djokic-Petrovic, M., Cvjetkovic, V., Yang, J., Zivanovic, M., Wild, D.J.: PIBAS FedSPARQL: a web-based platform for integration and exploration of bioinformatics datasets. J. Biomed. Semant. **8**(1), 42 (2017)
12. Farias, T.M.: The SPARQL endpoint of the SIB application and VoIDext specification examples. http://biosoda.expasy.org:8890/sparql. Accessed 25 Aug 2019
13. Farias, T.M.: VoIDext GitHub project repository. https://github.com/biosoda/voidext. Accessed 25 Aug 2019
14. Farias, T.M.: VoIDext vocabulary specification draft. https://biosoda.github.io/voidext/. Accessed 25 Aug 2019
15. Farias, T.M., Roxin, A., Nicolle, C.: FOWLA, a federated architecture for ontologies. In: Bassiliades, N., Gottlob, G., Sadri, F., Paschke, A., Roman, D. (eds.) RuleML 2015. LNCS, vol. 9202, pp. 97–111. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21542-6_7
16. de Farias, T.M., Roxin, A., Nicolle, C.: A rule-based methodology to extract building model views. Autom. Constr. **92**, 214–229 (2018)
17. Fernandez, R.C., Mansour, E., Qahtan, A.A., et al.: Seeping semantics: linking datasets using word embeddings for data discovery. In: 2018 IEEE 34th International Conference on Data Engineering (ICDE), pp. 989–1000. IEEE (2018)
18. Gal, A., Modica, G., Jamil, H., Eyal, A.: Automatic ontology matching using application semantics. AI Mag. **26**(1), 21 (2005)

19. Harris, S., Seaborne, A., Prud'hommeaux, E.: SPARQL 1.1 query language (2013). https://www.w3.org/TR/sparql11-federated-query/. Accessed 25 Aug 2019
20. Hasnain, A., Mehmood, Q., e Zainab, S.S., et al.: Biofed: federated query processing over life sciences linked open data. J. Biomed. Semant. **8**(1), 13 (2017)
21. Isele, R., Jentzsch, A., Bizer, C.: Efficient multidimensional blocking for link discovery without losing recall. In: WebDB (2011). https://dblp.org/rec/bibtex2/conf/webdb/IseleJB11
22. Jupp, S., Malone, J., Bolleman, J., et al.: The EBI RDF platform: linked open data for the life sciences. Bioinformatics **30**(9), 1338–1339 (2014)
23. Lehmann, J., Isele, R., Jakob, M., et al.: DBpedia-a large-scale, multilingual knowledge base extracted from wikipedia. Semant. Web **6**(2), 167–195 (2015)
24. Ngo, D., Bellahsene, Z.: Overview of YAM++–(not) yet another matcher for ontology alignment task. J. Web Semant. **41**, 30–49 (2016)
25. Ngomo, A.C.N., Auer, S.: LIMES–a time-efficient approach for large-scale link discovery on the web of data. In: Twenty-Second International Joint Conference on Artificial Intelligence (2011). aaai.org
26. Otero-Cerdeira, L., Rodríguez-Martínez, F.J., Gómez-Rodríguez, A.: Ontology-matching: a literature review. Expert. Syst. Appl. **42**(2), 949 – 971 (2015)
27. Peroni, S.: A simplified agile methodology for ontology development. In: Dragoni, M., Poveda-Villalón, M., Jimenez-Ruiz, E. (eds.) OWLED/ORE -2016. LNCS, vol. 10161, pp. 55–69. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54627-8_5
28. Sherif, M.A., Ngonga Ngomo, A.-C., Lehmann, J.: Wombat – a generalization approach for automatic link discovery. In: Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., Hartig, O. (eds.) ESWC 2017. LNCS, vol. 10249, pp. 103–119. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58068-5_7
29. Siegel, M.D., Madnick, S.E.: A metadata approach to resolving semantic conflicts (1991)
30. UniProt Consortium: UniProt: the universal protein knowledgebase. Nucl. Acids Res. **46**(5), 2699 (2018)
31. Whetzel, P.L., Noy, N.F., Shah, N.H., et al.: BioPortal: enhanced functionality via new web services from the national center for biomedical ontology to access and use ontologies in software applications. Nucl. Acids Res. **39**(Web Server issue), W541–5 (2011)
32. Wimalaratne, S.M., Bolleman, J., Juty, N., et al.: SPARQL-enabled identifier conversion with identifiers.org. Bioinformatics **31**(11), 1875–1877 (2015)
33. Živanovic, M.: SpecINT: a framework for data integration over cheminformatics and bioinformatics RDF repositories (2019). semantic-web-journal.net