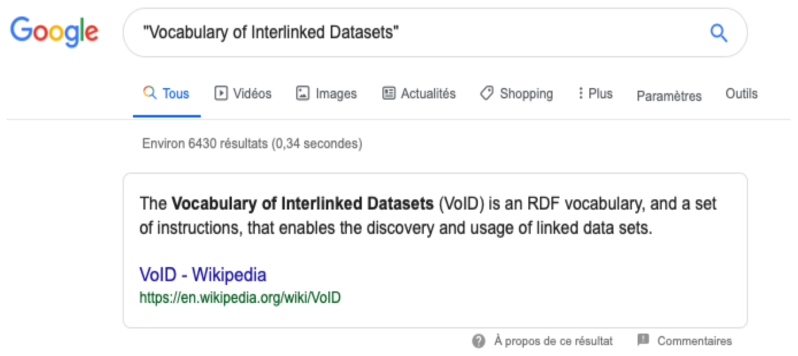
Supplementary material - enhancing interoperable datasets with virtual links

Tarcisio Mendes de Farias[1,2,3,4] (✉) [0000−0002−3175−5372] et al.

[1] SIB Swiss Institute of Bioinformatics, Lausanne, Switzerland
[2] Department of Computational Biology, University of Lausanne, Switzerland
[3] Department of Ecology and Evolution, University of Lausanne, Switzerland
[4] Center for Integrative Genomics, University of Lausanne, Switzerland
{tarcisio.mendesdefarias}@unil.ch

1 Our aim and VoID vocabulary popularity

To enhance semantic interoperability at the data level 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. At the time of writing, more than 600 scientific articles mention VoID[5], which shows the high relevance and usefulness of this vocabulary to the scientific community. Likewise, searching for exact matches of "Vocabulary of Interlinked Datasets" in web search engines such as Google and Bing return more than 6,400 results[6] (see Fig. 1).


"Vocabulary of Interlinked Datasets"

Environ 6430 résultats (0,34 secondes)

The **Vocabulary of Interlinked Datasets** (VoID) is an RDF vocabulary, and a set of instructions, that enables the discovery and usage of linked data sets.

VoID - Wikipedia
https://en.wikipedia.org/wiki/VoID

Fig. 1. Searching for exact matches of "Vocabulary of Interlinked Datasets". It retrieves 6,430 results.

[5] https://scholar.google.com/scholar?q=%22Vocabulary+of+Interlinked+Datasets%22&oq=voca
[6] https://www.google.com/search?q=%22Vocabulary+of+Interlinked+Datasets%22

2 Background - Simplified Agile Methodology for Ontology Development (SAMOD)

Among the agile methodologies for ontology development, we can mention the eXtreme Design methodology (XD) [5] and Simplified Agile Methodology for Ontology Development (SAMOD) [12]. SAMOD is an iterative process that is inspired by the XD methodology. We chose SAMOD to extend VoID and we justify this choice in Section 3. The key concepts of SAMOD are: *the current model*, an initial, intermediary or final TBox (Terminological Box), if any; the *modelet*, a stand-alone model describing a particular domain; and a set of test cases T_n where n is the current iteration counter value. T_n is a sextuple $(MS_n, CQ_n, GoT_n, TBox_n, ABox_n, SQ_n)$ where MS_n is a motivating scenario; CQ_n is scenario-related informal competency questions; GoT_n a glossary of terms related to MS_n and CQ_n; $TBox_n$ a modelet or current model implementing the description introduced in MS_n; $ABox_n$ is the implemented exemplar dataset based on MS_n and according to the $TBox_n$ (ABox - Assertion Box); SQ_n is a set of queries written in a formal language that implements the CQ_n questions. The SAMOD methodology is depicted in Fig. 2.

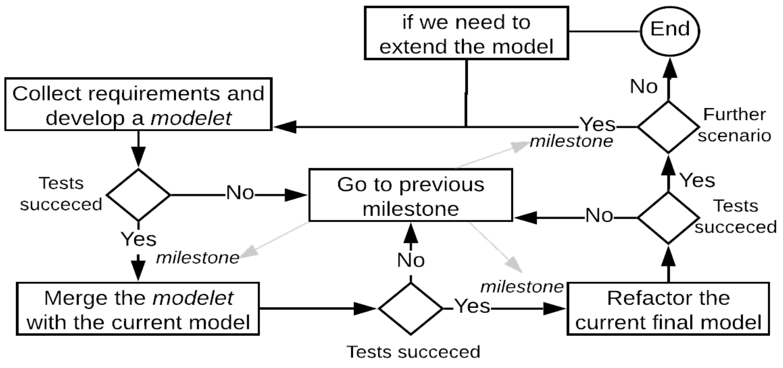

if we need to extend the model
End
No
Collect requirements and develop a *modelet*
Yes
Further scenario
milestone
Tests succeced
No
Go to previous milestone
Yes
Tests succeced
No
Yes
milestone
No
milestone
Merge the *modelet* with the current model
Yes
Refactor the current final model
Tests succeced

Fig. 2. A summary of SAMOD, starting with the "collect..." step (modified from [12]).

3 Building VoIDext

To overcome these impediments to semantic interoperability, we introduce a vocabulary and patterns to describe the semantic relations among datasets. This vocabulary so-called VoIDext is actually an extension and an adaptation of the well-accepted VoID RDF schema vocabulary. In this Supp. Material, the words vocabulary and ontology are interchangeably used. The methodology applied to develop VoIDext was inspired by the simplified agile methodology for ontology development (SAMOD) [12] (see further explanations in Section 2). We mainly chose SAMOD because it is a methodology designed to quickly develop small- and medium-size ontologies and does not require "pair programming"—it usually involves only one ontology engineer. In principle, SAMOD states the involvement of two persona profiles, namely a domain expert and an ontology engineer. The domain expert is mostly required when developing domain ontologies. In the context of VoIDext, a domain expert was not involved, because it is not a domain ontology. Indeed, the proposed VoID extension is a meta-ontology that describes semantic links between RDF datasets — virtual links.

Therefore, we define the test case $T_1 = (MS_1, CQ_1, GoT_1, TBox_1, ABox_1, SQ_1)$ by following the SAMOD methodology. MS_1, CQ_1, and GoT_1 are described in the next subsections. For the sake of simplicity and due to the limit of pages, we do not discuss the intermediary exemplar datasets (ABox) and query sets (SQ) resulted from the two first steps of SAMOD. Indeed, they are quite similar to the ABox and SQ created in the end of the SAMOD process, namely $ABox_1$ and SQ_1. Both of them are available in [7]. $ABox_1$ is built based on a real case study involving several life sciences RDF stores, thus it is not an examplar dataset solely created for test purposes. Moreover, the VoID vocabulary is the current model (i.e. $TBox_0$) since we are extending it. Thus, VoID is an input of the SAMOD iterative process. Finally, VoIDext RDF schema vocabulary is the SAMOD process output, namely $TBox_1$ that is further documented in [8].

3.1 The motivating scenario (MS_1)

Name: Virtual links between RDF datasets.

Description: an RDF dataset is a set of RDF triples that are published, maintained or aggregated by a single provider [2]. RDF datasets are often independent and distributed on the Web. A virtual link is a connection between common resources such as literals and instances from two different RDF datasets. Semantic relaxation is also considered when identifying common resources between datasets. For example, the datasets might use different prefix namespaces for similar instances in a given domain scope.

A virtual link can be interpreted as an intersection data point between two datasets. In addition, a virtual link is not explicitly and concretely stored. The link may be physically established, for example, during a conjunctive federated query execution. Virtual links can simply be predicates (i.e. relations) in one dataset where their subjects and objects are present in disparate datasets. The types of resources involved in a virtual link can be also a valuable information. Another kind of virtual link can be stated when two datasets share the same instances of a given type. These instances usually do not have exactly the same description or context in each dataset.

Moreover, distinct datasets can state different predicates with the same or similar subjects and objects. By considering these subjects and objects which are in common, the RDF graphs from these datasets may establish two types of intersection nodes (i.e. matched nodes): object-object, subject-subject, and subject-object. For example, given the following RDF triples $t_A = (S_A, P_A, O_A)$ in dataset A and $t_B = (S_B, P_B, O_B)$ in dataset B; if $O_A \equiv O_B$ then we can state a virtual link between O_A and O_B; or else, if $S_A \equiv S_B$ then we can state a virtual link between S_A and S_B; otherwise if $S_A \equiv O_B$ then we can state a virtual link between S_A and O_B. The $\equiv$ symbol represents equivalent subjects or objects.

One could also imagine the case of O_A being similar to O_B (i.e. $O_A \sim O_B$). In this case, a resource mapping function $f_m(r)$ is required to establish a virtual link between O_A and O_B — i.e. either $O_A \equiv f_m(O_B)$ or $f_m(O_A) \equiv O_B$. This is due to the fact of existing heterogeneities in the representation of a resource (e.g. "id:1234" or "id_1234" strings) and the resource description, for example, different set of predicates to contextualize a similar resource in different datasets and domain scopes.

Finally, we can also mention the case of two separate datasets sharing the same instance IRIs but from different types (i.e. classes). Yet, a virtual link can be stated between these instances. This fact might be because of distinct domain scopes and constraints such as legacy information systems.

Examples: the following motivating scenario examples are based on existing life sciences RDF datasets on the Web. The RDF data may be accessible through SPARQL endpoints. The considered datasets in this Supp. Material are as follows: OMA (Orthologous MAtrix) [3], UniProtKB (UniProt Knowledgebase) [13], Drugbank [1], Bgee (dataBase for Gene Expression Evolution) [4], and EBI (European Bioinformatics Institute) RDF platform [11]. Table 1 lists the SPARQL endpoints of these datasets and Table 2 depicts the prefixes and the corresponding IRI namespaces used in this Supp. Material.

Table 1. SPARQL endpoints considered in this Supp. Material.

RDF Dataset	**SPARQL endpoint**
DBpedia [**?**]	http://dbpedia.org/sparql
LINDAS**??**	https://lindas-data.ch/sparql
MusicBrainz [10]	http://dbtune.org/musicbrainz/sparql
OMA [3]	https://sparql.omabrowser.org/sparql
UniProtKB[13]	https://sparql.uniprot.org/sparql
Bgee[4]	http://biosoda.expasy.org:8080/rdf4j-server/repositories/bgeelight_mysql
Drugbank[1]	http://wifo5-04.informatik.uni-mannheim.de/drugbank/sparql
EBI RDF[11]	https://www.ebi.ac.uk/rdf/services/sparql

Table 2. In this Supp. Material, 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#
orth:	http://purl.org/net/orth#
up:	http://purl.uniprot.org/core/
oboowl:	http://www.geneontology.org/formats/oboInOwl#
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#
mo:	http://purl.org/ontology/mo/
lscr:	http://purl.org/lscr#
void:	http://rdfs.org/ns/void#
voidext:	http://purl.org/query/voidext#
bioquery:	http://purl.org/query/bioquery#

Example 1: directed link predicates between datasets.

A dataset can contain property assertions involving IRIs which are also present in another dataset. These IRIs in common, that are referred as instances of a given class, often do not necessary have the same set of properties or context in different datasets, however they are frequently complementary. For example, let us consider the OMA and UniProt RDF datasets. Fig. 3 illustrates a triple in OMA where the predicate is a cross-reference property (i.e. *lscr:xrefUniprot*) that assigns as value an IRI in common with Uniprot. By considering this, a direct virtual link such as the one shown in Fig. 3 can be stated where the subject of *lscr:xrefUniprot* is described in OMA and the object is further depicted in UniProt. To find out if the virtual link exists, we need to know which dataset contains the triples with the *lscr:xrefUniprot* predicate and what is the another one with the triples' objects. In this example, the objects are further described in the UniProt RDF dataset. Yet, it is suitable to know what is the context of the *lscr:xrefUniprot* predicate's object in UniProt. In this case, the UniProt IRI that is illustrated as a black circle in Fig. 3 indeed refers to an instance of the *up:Protein* class in UniProt. Therefore, *up:Protein* must be asserted as the range of the virtual link. Although in this example the virtual link's range and the *lscr:xrefUniprot* property's range in OMA are the same, it is not always the case when establishing directed virtual links.

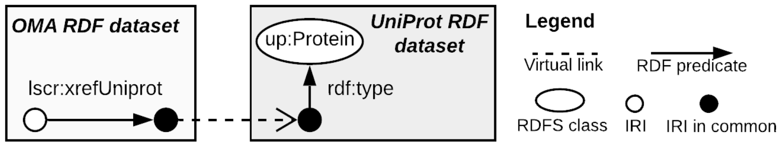


Fig. 3. A directed virtual link predicate between OMA and UniProt datasets.

Example 2: similar or common instances between datasets.
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. This link is illustrated in Fig. 4. To establish this link, we need to define an IRI mapping function (i.e. $f_m(r)$) either to the EBI or UniProt species-related instances — either f_m(*<http://identifiers.org/taxonomy/9606>*) ≡*<http://purl.uniprot.org/taxonomy/9606>* or f_m(*<http://purl.uniprot.org/taxonomy/9606>*) ≡*<http://identifiers.org/taxonomy/9606>*.

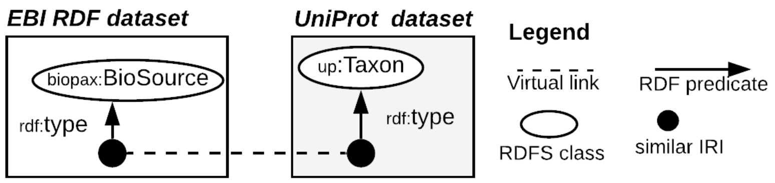


Fig. 4. A virtual link between datasets based on similar IRIs of different types.

Example 3: complex links between datasets. Since RDF does not impose any requirements besides the ones defined in the RDF abstract syntax, RDF triples can be published without a complete or any data schema. This makes more challenge to interoperate already published RDF datasets where semantic relations cannot be defined at the data schema level. For example, the *drugbank:swissprotId* is an RDF predicate but it is not explicitly stated as an RDF property and its object IRIs are not typed (i.e. missing *rdf:type* statement). Nevertheless, we can still establish a virtual link with the EBI RDF dataset as illustrated in Fig. 5. The EBI dataset contains subsets with *oboowl:hasDbXref* property assertions. In this context, several string values asserted to the *oboowl:hasDbXref* property may correspond to IRIs assigned with the *drugbank:swissprotId* predicate. By considering a semantic relaxation, we can state a virtual link between these two datasets. To establish this link, we have to define a mapping function (i.e. $f_m(r)$) either to the *oboowl:hasDbXref* string values or *drugbank:swissprotId* object IRIs — e.g. either f_m(*<http://bio2rdf.org/uniprot:P04275>*) ≡ *"SwissProt:P04275"* or f_m(*"SwissProt:P04275"*) ≡*<http://bio2rdf.org/uniprot:P04275>*. In doing so, we do not need to rely on external services to process the Compact URLs "SwissProt:P04275" and the query can be performed by only using the SPARQL endpoints of the two resources that we want to interlink. Moreover, the compact URL (cURL) "SwissProt:P04275" cannot be solved by the resolution service (identifiers.org – state-of-the-art for solving cURL in life sciences) — see Fig. 6, since it is expecting "uniprot:P04275" rather than "SwissProt:P04275".

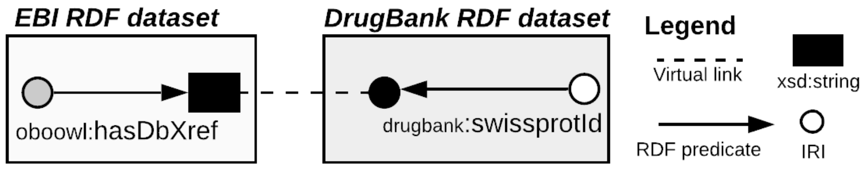
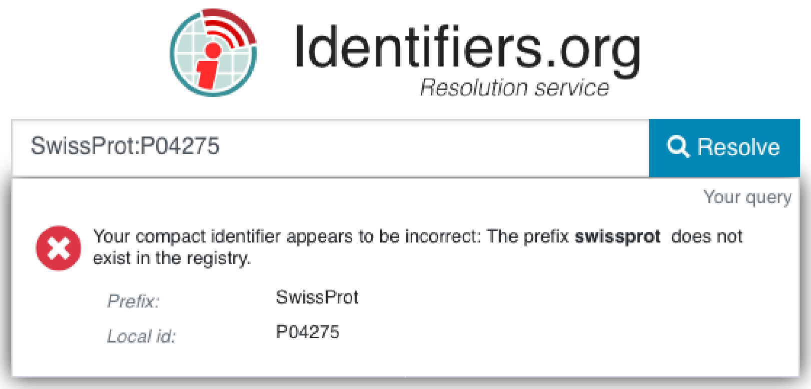


Fig. 5. A complex virtual link between datasets involving two different RDF predicates (excluding *rdf:type*).

3.2 Informal competency questions (CQ_1).

We describe in Tables 3, 4, 5 and 6 the most relevant competency questions (CQ) raised during VoIDext development. Tables 3, 5 and 6 refer to retrieve *virtual links* such as those illustrated in Figures 3, 4 and 5, respectively.

A *virtual link set* (VL) can be classified as simple or complex. A simple VL set must contain in its definition, either exactly one link predicate such as exemplified in Table 3 or exactly one shared instance type (i.e. class) such as depicted in Table 4. The simple VL set defined with exactly one link predicate (i.e. T_1 in Table 3) is also described with information about the source dataset S. S contains the predicate statements, and the type of either the link predicate' subject or object that is stated in a dataset different from S. For example, Fig. 3 illustrates the corresponding object of *lscr:xrefUniprot* link predicate in UniProt dataset where *lscr:xrefUniprot* is asserted and stored in OMA dataset. This is also exemplified in Table 5.



SwissProt:P04275

Your compact identifier appears to be incorrect: The prefix **swissprot** does not exist in the registry.

Prefix:	SwissProt
Local id:	P04275

Fig. 6. "SwissProt:P04275" cannot be solved by the resolution service (identifiers.org – state-of-the-art for solving cURL in life sciences)

Table 3. Competency question about directed virtual links. ϕ represents an empty value and "xor" is the "exclusive or".

Question (1): which are the directed virtual link sets between two RDF datasets? In other words, the simple link sets composed of exactly one link predicate (non rdf:type).
Outcome: A set of tuples $T_1 = (V_L, S, p_o, s_o, s_t, p, o_t, o_o, f_m)$ where V_L is the direct virtual link set IRI; S is the dataset source name where the link predicates are asserted; p_o is the access method to the link predicate assertions (e.g. an SPARQL endpoint); s_o is the access method to the link predicate' subject; s_t is the subject type (DL-based class expression); p is the link predicate; o_t is the object type (DL-based class expression); o_o is the access method to the link predicate's object; f_m is the resource mapping procedure, if any. f_m must be applied to the p predicate' object in the dataset S if $p_o \neq o_o$, otherwise the mapping is applied to the p predicate' subject in S. In addition, the s_t xor o_t statements and the p predicate are in different datasets (e.g. SPARQL enpoints).
Example: T_1^1 = (bioquery:OMA_UNIPROT_1, "Orthologous Matrix (OMA)", <https://sparql.omabrowser.org/sparql>, <https://sparql.omabrowser.org/sparql>, orth:Protein, lscr:xrefUniprot, up:Protein, <https://sparql.uniprot.org/sparql/>, ϕ)

In addition, a more complex description to define *virtual link sets* is required whenever a simple VL set between two resources cannot be established. A complex VL set is composed of exactly either two link predicates or two shared instance types. Complex *virtual links* are illustrated in Figures 4 and 5.

Fig. 4 and Table 5 depict the undirected complex VL between two instance IRIs with some commonality but typed with unlike classes in different datasets. If the commonality means equivalent IRIs, the mapping function f_m is not required (i.e. $f_m = \phi$, where ϕ is an empty value). Nonetheless, the example presented in Table 5 specifies a mapping function f_m^3 to homogenize corresponding instance IRIs between EBI and UniProt RDF datasets. In principle, we can define two mapping functions: a function to map IRIs from EBI to UniProt dataset and another one to perform the inverse mapping (i.e. from Uniprot to EBI dataset). Because of this, when describing a complex VL set that requires a mapping function such as the one in Table 5, we should state what is the recommended mapping to be considered.

Furthermore, it is often preferable or only possible to define a VL set between two datasets if we know the two link predicates from each dataset — i.e. a complex VL set. This is because a complex VL set is required whenever the

Table 4. Competency question about bidirectional virtual links between common instances. ϕ represents an empty value.

Question (2): which are the set of virtual links between two shared instances of the same class in different RDF datasets? In other words, the simple link sets that are also a Shared instance set
Outcome: A set of tuples $T_2 = (V_L, ds_1, ds_2, I_t, A_{ds_1}, A_{ds_2})$ where V_L is the virtual link IRI; ds_1 and ds_2 are the name of the datasets that contain the instance IRIs in common; I_t is the type (DL-based class expression) of the instances in common; A_{ds_1} and A_{ds_2} are the access methods such as SPARQL endpoints to the ds_1 and ds_2 datasets, respectively.
Example: T_2^1 = (bioquery:OMA_BGEE_2, "Bgee - a database of gene expression", "Orthologous Matrix (OMA)", up:Taxon, <http://biosoda.cloudlab.zhaw.ch:8080/rdf4j-server/repositories/bgeelight>, <https://sparql.omabrowser.org/sparql>)

Table 5. Competency question about undirected virtual links between instances of different types.

Question (3): which are the set of virtual links between two similar or equivalent instances of unlike types in different RDF datasets? In other words, complex link sets composed of two Shared instance sets
Outcome: A set of tuples $T_3 = (V_L, ds_1, ds_2, I_{t1}, I_{t2}, A_{ds_1}, A_{ds_2}, f_m)$ where V_L is the virtual link set IRI; ds_1 and ds_2 are the name 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: T_3^1 = (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/>, $f_m^3(i)$) where i is any instance of biopax:BioSource type and $f_m^3(i) \equiv$ "'?i a <http://www.biopax.org/release/biopax-level3.owl#BioSource>. BIND(IRI(CONCAT("http://purl.uniprot.org/taxonomy/", STRAFTER(STR(?i), "http://identifiers.org/taxonomy/"))) as ?NEW_IRI) FILTER(STRSTARTS(STR(?i), "http://identifiers.org/taxonomy/"))"'

matched subject or object of predicates in different datasets are not typed (i.e. missing *rdf:type* assertions) or the matched object is a literal. Fig. 3 along with Table 6 exemplify both cases when establishing a complex VL set between the DrugBank and EBI RDF datasets.

3.3 Glossary of terms (GoT_1)

The complete list of VoIDext terms is available on the documentation of VoIDext (i.e. $TBox_1$) in [8].

3.4 A brief overview of the resulted SQ_1, $ABox_1$ and $TBox_1$

VoID instances (ABox) are fully backward compatible with the VoIDext schema since we add new terms without modifying the original VoID TBox. The only performed modification concerns the void:target property domain. In VoIDext, this domain is the union of 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.

Table 6. Competency question about complex virtual links between subjects and objects of different predicates in distinct datasets. "xor" is the "exclusive or" and ϕ is an empty value.

Question (4): Which are the sets of complex virtual links between two similar or equivalent resources that are either subject or object of different predicates in distinct RDF datasets? In other words, Which are the complex virtual link sets composed of two link predicates stored in distinct datasets?
Outcome: A set of $T_4 = (T_4^1, T_4^2)$ tuples where T_4^1 and T_4^2 are also tuples and defined as follows: $T_4^1 = (V_L, ds_1, ds_2, p^1, I_{type}, s_t^1, o_t^1, A_{ds_1}, A_{ds_2}, f_m^1)$; $T_4^2 = (V_L, ds_2, ds_1, p^2, I_{type}, s_t^2, o_t^2, A_{ds_1}, A_{ds_2}, f_m^2)$; where V_L is the virtual link IRI; ds_1 is the dataset that contains p^1 predicate and its subject-object resources such as IRIs or literals that are equivalent or similar to either the p^2 predicate's object or p^2 subject; ds_2 is the dataset that contains p^2 and its subject-object resources that are equivalent or similar to either the p^1 predicate's object or p^1 subject; I_{type} is the type of the intersection/matching (e.g. object-object); s_t^1 is the subject's type of p^1 in ds_1; o_t^1 is the object's type of p_1 in ds_1; s_t^2 is the subject's type of p^2 in ds_2; o_t^2 is the object's type of p_2 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^1 xor f_m^2 is the recommended resource mapping procedure. If any mapping exists then solely one of them is not equal to ϕ, otherwise both f_m^1 and f_m^2 are equal to ϕ.
Example: T_4^1 = (bioquery:DRUGBANK_EBI-ORDO_VL, "Orphanet Rare Disease Ontology (ORDO)", "Drug Bank RDF", oboowl:hasDbXref, "Object-object intersection", ϕ, xsd:string, <https://www.ebi.ac.uk/rdf/services/sparql> , <http://wifo5-04.informatik.uni-mannheim.de/drugbank/sparql>, $f_m^1(i)$) T_4^2 = (bioquery:DRUGBANK_EBI-ORDO_VL, "Drug Bank RDF", "Orphanet Rare Disease Ontology (ORDO)", drugbank:swissprotId, "Object-object intersection", drugbank:targets, ϕ , <https://www.ebi.ac.uk/rdf/services/sparql> , <http://wifo5-04.informatik.uni-mannheim.de/drugbank/sparql>, ϕ) where i is the oboowl:hasDbXref object. $f_m^1(i) \equiv$ "'?g <http://www.geneontology.org/formats/oboInOwl#hasDbXref> ?i. BIND(IRI(CONCAT("http://bio2rdf.org/uniprot:", STRAFTER(?i,"SwissProt:"))) as ?NEW_IRI) FILTER (CONTAINS(?i, "SwissProt:"))"'

Despite this modification, assertions of *void:target* based on VoID remain compatible with VoIDext. Minor modifications are reported in the end of the VoIDext documentation available in [8].

Listing 1.2 shows the formal query in SQ_1 to answer the informal competency question in Table 6 that is related to the MS_1 motivating scenario described in Subsection 3.1. The tuples T_4^1 and T_4^2 are obtained by considering that each variable in the SPARQL query projection corresponds to an element in these tuples. As a result, we can define the T_4 tuple, in other words, the competency question answer. The other SQ_1 queries to answer the competency questions discussed in this Supp. Material are available in [7]. The SQ_1 queries can be executed on the SPARQL endpoint in [6].

4 Patterns to model simple link sets with VoIDext

Fig. 7 shows the representation of a simple link set based on *owl:sameAs* assertions for musical artists that also includes bands from the MusicBrainz dataset [10] to DBpedia. In Fig. 7, we can easily compare both representations of this simple link set only using VoID terms (left-hand side) with a modelling based on VoIDext (right-hand side). With VoIDext, we explicitly state that

```
bioquery:DRUGBANK_EBI-ORDO a void:Linkset;
             void:subjectsTarget bioquery:DRUGBANK;
             rdfs:label "The link predicate description from drugbank dataset
                  to the ORDO/EBI rare diseases dataset.";
             void:objectsTarget bioquery:EBI-ORDO_DRUGBANK;
             void:linkPredicate drugbank:swissprotId;
             voidext:linkPredicateDomain drugbank:targets;
             voidext:isSubsetOf bioquery:DRUGBANK ;
                     dcterms:issued  "2019-06-30"^^xsd:date .

bioquery:EBI-ORDO_DRUGBANK a void:Linkset;
             rdfs:label "The link predicate description from ORDO/EBI rare
                 diseases dataset to the drugbank dataset.";
             void:objectsTarget bioquery:DRUGBANK_EBI-ORDO;
             void:subjectsTarget bioquery:EBI_ORDO;
             void:linkPredicate oboowl:hasDbXref;
             voidext:linkPredicateRange xsd:string ;
             voidext:resourceMapping '''
                 ?g <http://www.geneontology.org/formats/oboInOwl#hasDbXref>
                      ?i .
                  BIND( IRI(CONCAT("http://bio2rdf.org/uniprot:",STRAFTER(?i
                       ,"SwissProt:"))) as ?NEW_IRI)
                  FILTER (CONTAINS(?i, "SwissProt:" ))''';
             voidext:isSubsetOf bioquery:EBI_ORDO ;
                     dcterms:issued  "2019-06-30"^^xsd:date .

 bioquery:DRUGBANK_EBI-ORDO_VL a voidext:ComplexLinkSet;
            rdfs:label "Virtual links for rare diseases (ORDO/EBI dataset)
                 and drug targets (drugbank dataset).";
                         voidext:intersectAt bioquery:EBI-ORDO_DRUGBANK ;
                         voidext:intersectAt bioquery:DRUGBANK_EBI-ORDO ;
                         voidext:intersectionType voidext:OBJECT_OBJECT ;
                    voidext:recommendedMapping bioquery:EBI-ORDO_DRUGBANK ;
                    dcterms:issued  "2019-06-30"^^xsd:date .


bioquery:EBI a void:Dataset.
bioquery:DRUGBANK a void:Dataset.

bioquery:ORDO sd:name <http://rdf.ebi.ac.uk/dataset/ordo>;
            a sd:Graph.
bioquery:ORDO_2_6 sd:name <http://rdf.ebi.ac.uk/dataset/ordo/2.6>;
            a sd:Graph.

bioquery:EBI_ORDO a sd:Dataset, void:Dataset;
            dcterms:title "Orphanet Rare Disease Ontology (ORDO)";
            sd:namedGraph bioquery:ORDO, bioquery:ORDO_2_6 ;
            sd:defaultGraph [
                        a sd:Graph, void:Dataset;
                        dcterms:title "EBI RDF serialisation" ];
            void:sparqlEndpoint <https://www.ebi.ac.uk/rdf/services/sparql>;
            void:subset bioquery:EBI.
```

Listing 1.1. A portion of the $ABox_1$ serialized in RDF/Turtle syntax that is also partially illustrated in Fig. **??**.

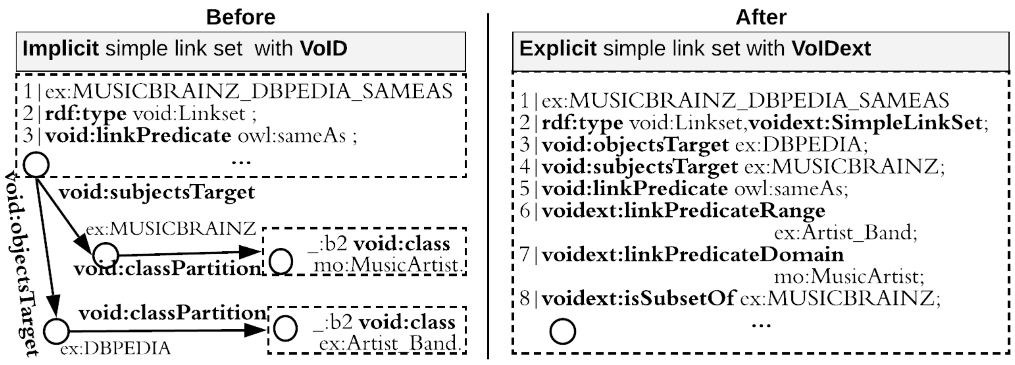
```
prefix void:     <http://rdfs.org/ns/void#>
prefix bioquery: <http://purl.org/query/bioquery#>
prefix voidext:<http://purl.org/query/voidext#>
prefix dcterms: <http://purl.org/dc/terms/>

SELECT distinct ?links ?source_dataset1_name ?target_dataset2_name ?predicate
 ?intersection_type ?subj_type ?obj_type  ?source_endpoint ?target_endpoint
 ?resourceMapping {

 #values(?dataset1){ (bioquery:EBI_ORDO) }
 #values(?dataset2){ (bioquery:DRUGBANK) }

 ?links a voidext:ComplexLinkSet.
 ?links  voidext:intersectAt ?set2;
         voidext:intersectionType/rdfs:label ?intersection_type.
 ?set2   voidext:isSubsetOf ?target_db.
 ?target_db dcterms:title ?target_dataset2_name;
            void:sparqlEndpoint ?target_endpoint.
 ?set2 void:linkPredicate ?predicate2 .


 ?links  voidext:intersectAt ?set1.
 ?set1 void:linkPredicate ?predicate .
 ?set1 voidext:isSubsetOf ?source_db .
 ?source_db dcterms:title ?source_dataset1_name;
            void:sparqlEndpoint ?source_endpoint.
optional{ ?set1 voidext:linkPredicateDomain ?subj_type}
optional{ ?set1 voidext:linkPredicateRange ?obj_type.}
optional{ ?links voidext:recommendedMapping ?set1.
               ?set1 voidext:resourceMapping  ?resourceMapping}
filter(?source_db != ?target_db)

} order by ?links
```

Listing 1.2. The SPARQL query to answer the competency question 4 in Table 6.

this *owl:sameAs*-link set is also a *voidext:SimpleLinkSet* – see line 2 in the Fig. 7 right-hand side. Besides this, it is useful to know which are the subjects' types of *owl:sameAs* to consider since not all *owl:sameAs* assertions are related to DBpedia, and more specifically to musicians and bands. To do so, we can directly assign the *voidext:linkPredicateDomain* property to the link set – see line 7 in the Fig. 7 right-hand side. Similarly, we can explicitly declare the objects' types of the link predicate by asserting the *voidext:linkPredicateRange* property – see line 6. Since it is a simple link set, the range must be defined as an OWL-DL class expression based on the TBox of the dataset where the object is described (i.e. *void:objectsTarget*). In Fig. 7, the *owl:sameAs* range is defined with OWL 2 DL as follows: *ex:Artist_Band* ≡ *dbo:Artist* ⊔ *dbo:Band* ⊔ *<http://dbpedia.org/class/yago/Artist109812338>*.

To explicitly define a simple virtual link set between two datasets that share the same instances of the same class, we can instantiate the class *voidext:SharedInstanceSet* (Def. 1). An example of a shared instance set between Bgee [4] and OMA datasets is depicted in Listing 1.3.

Definition 1 (shared instance set). *A shared instance set between exactly two datasets. For example, two datasets that contain the same OWL/RDFS class instances.*


Before
Implicit simple link set with VoID
void:subjectsTarget
void:objectsTarget
ex:MUSICBRAINZ
void:classPartition
_:b2 void:class mo:MusicArtist
void:classPartition
ex:DBPEDIA
_:b2 void:class ex:Artist_Band
After
Explicit simple link set with VoIDext

```
ex:MUSICBRAINZ_DBPEDIA_SAMEAS
rdf:type void:Linkset ;
void:linkPredicate owl:sameAs ;
...
```

```
ex:MUSICBRAINZ_DBPEDIA_SAMEAS
rdf:type void:Linkset,voidext:SimpleLinkSet;
void:objectsTarget ex:DBPEDIA;
void:subjectsTarget ex:MUSICBRAINZ;
void:linkPredicate owl:sameAs;
voidext:linkPredicateRange
                              ex:Artist_Band;
voidext:linkPredicateDomain
                              mo:MusicArtist;
voidext:isSubsetOf ex:MUSICBRAINZ;
...
```

Fig. 7. Comparing the modelling of a simple link set based on *void:Linkset* between VoID and VoIDext. Circles: different instance IRIs; dashed rectangles: instance's attributes; and edges: RDF predicates.

```
bioquery:OMA_BGEE_3 a voidext:SharedInstanceSet , voidext:SimpleLinkSet;
        void:target bioquery:OMA, bioquery:BGEE;
        voidext:sharedInstanceType up:Taxon.
```

Listing 1.3. A simple link set between OMA and Bgee datasets about organism taxonomy. Tab. 2 contains the IRI prefixes.

Therefore, with VoIDext, we can model in a less verbose and more explicit way the simple link sets. We also avoid multiple representations allowed by VoID to define the link predicate's range/domain. For instance, VoID does not restrict the subject or object targets to a single class partition, thus if multiple class partitions are defined, we are not able to distinguish which classes refers to the domain/range of the link predicate. Further examples of simple link sets involving the datasets in Tab. 1 and defined with VoIDext are available in [7].

5 Virtual link set maintenance

Let us consider the example depicted in Fig. 7. This link set that is a *voidext:SimpleLinkSet* between artists and bands in DBpedia and MusicBrainz. The link set contains fewer links than a complex link set composed of the following link predicates: *foaf:name* in MusicBrainz and *rdfs:label* in DBpedia. Indeed, there are 812 against 530 links in this complex link set. This complex link set is the *ex:DBPEDIA_MUSICBRAINZ_VL* instance and its serialization in RDF/Turtle syntax is available in [7]. This set enables to establish links that were not possible with the simple link set because of missing *owl:sameAs* assertions in the MusicBrainz dataset. For example, this is the case of the "Izaline Calister" artist resource in the MusicBrainz RDF dataset that does not assert the *owl:sameAs* property with *dbr:Izaline_Calister* IRI from DBpedia. Moreover, we can also rely on the performance metrics to choose a virtual link set among different link sets of the same purpose such as linking artists between DBpedia and MusicBrainz. For example, if we want to write a federated query related to artists that requires a better recall than precision, we should choose the *ex:DBPEDIA_-MUSICBRAINZ_VL* based on names/labels rather than the *owl:sameAs* link set depicted in Fig. 7.

6 Figures illustrating patterns to model virtual link sets

Figures 8, 9, 10, and 11 illustrate Listings 3, 4, 5, and 6 in [9], respectively.

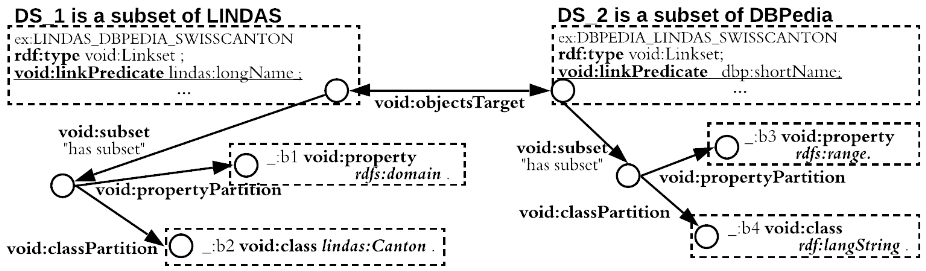

DS_1 is a subset of LINDAS
ex:LINDAS_DBPEDIA_SWISSCANTON
rdf:type void:Linkset ;
void:linkPredicate lindas:longName ;
...
DS_2 is a subset of DBPedia
ex:DBPEDIA_LINDAS_SWISSCANTON
rdf:type void:Linkset;
void:linkPredicate dbp:shortName;
...
void:objectsTarget
void:subset
"has subset"
_:b1 void:property rdfs:domain .
void:propertyPartition
void:classPartition
_:b2 void:class lindas:Canton .
void:subset
"has subset"
_:b3 void:property rdfs:range.
void:propertyPartition
void:classPartition
_:b4 void:class rdf:langString .

Fig. 8. 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*). Circles: different instances; dashed rectangles: instance attributes; and edges: RDF predicates.

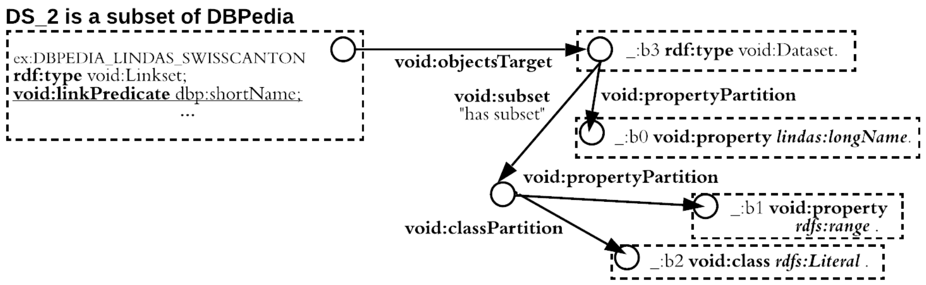
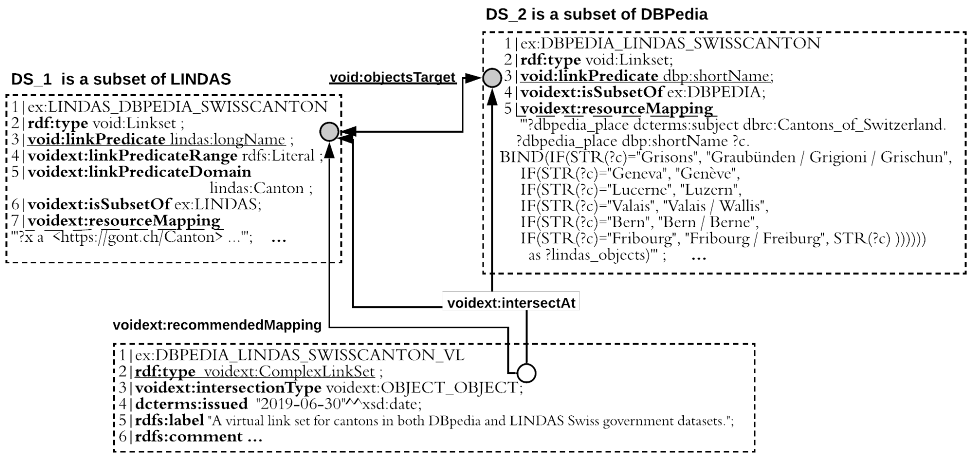

DS_2 is a subset of DBPedia
ex:DBPEDIA_LINDAS_SWISSCANTON
rdf:type void:Linkset;
void:linkPredicate dbp:shortName;
...
void:objectsTarget
_:b3 rdf:type void:Dataset.
void:subset
"has subset"
void:propertyPartition
_:b0 void:property lindas:longName.
void:propertyPartition
_:b1 void:property rdfs:range .
void:classPartition
_:b2 void:class rdfs:Literal .

Fig. 9. 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 illustrated because the link set in LINDAS containing the *lindas:longName* link predicate is similarly modelled as the one in DBpedia. Circles: different instances; dashed rectangles: instance attributes; and edges: RDF predicates.

References

1. D2R server publishing the DrugBank database. `http://wifo5-03.informatik.uni-mannheim.de/drugbank/`, accessed: 2019-4-7
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, April 20, 2009, CEUR Workshop Proceedings (2009), `http://ceur-ws.org/Vol-538/ldow2009_paper20.pdf`

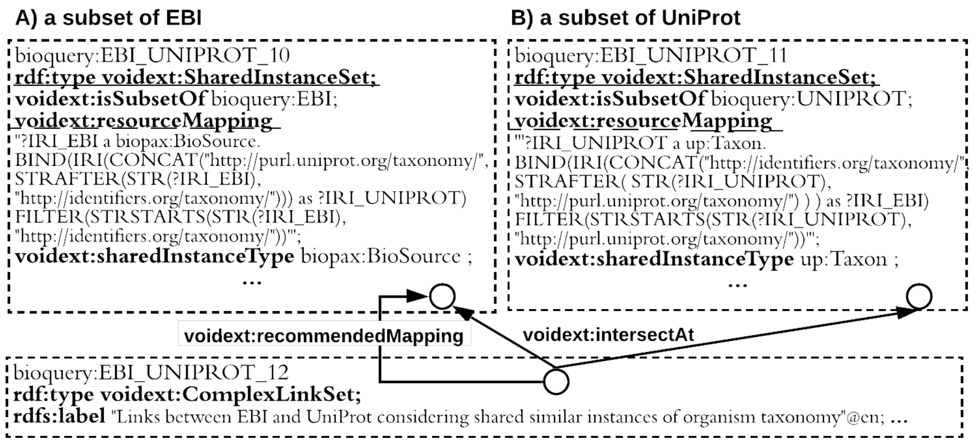

DS_2 is a subset of DBPedia
DS_1 is a subset of LINDAS
void:objectsTarget
voidext:intersectAt
voidext:recommendedMapping

```
ex:LINDAS_DBPEDIA_SWISSCANTON
rdf:type void:Linkset ;
void:linkPredicate lindas:longName ;
voidext:linkPredicateRange rdfs:Literal ;
voidext:linkPredicateDomain
                          lindas:Canton ;
voidext:isSubsetOf ex:LINDAS;
voidext:resourceMapping
'''?x a <https://gont.ch/Canton> ...''';    ...
```

```
ex:DBPEDIA_LINDAS_SWISSCANTON
rdf:type void:Linkset;
void:linkPredicate dbp:shortName;
voidext:isSubsetOf ex:DBPEDIA;
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.";
rdfs:comment ...
```

Fig. 10. VoIDext-based patterns to model a complex virtual link set between the LINDAS Linked Data service and DBpedia relying on link sets as targets. Circles: different instances; dashed rectangles: instance attributes; edges: RDF predicates; dashed underlined: one of the two can be chosen as the *voidext:recommendedMapping*; fully underlined: predicates used to connect the datasets by *void:objectsTarget* predicates.



```
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 ;
                    ...
```

```
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;
rdfs:label "Links between EBI and UniProt considering shared similar instances of organism taxonomy"@en; ...
```

Fig. 11. VoIDext-based patterns to model a complex virtual link set between EBI and UniProt datasets modelled with shared instance sets (see fully underlined assertions). Circles: different instances; dashed rectangles: instance attributes; edges: RDF predicates; dashed underlined: one of the two can be chosen as the *voidext:recommendedMapping.*

3. Altenhoff, A.M., Glover, N.M., Train, C.M., Kaleb, K., Warwick Vesztrocy, A., Dylus, D., de Farias, T.M., Zile, K., Stevenson, C., Long, J., Redestig, H., Gonnet, G.H., Dessimoz, C.: The OMA orthology database in 2018: retrieving evolutionary relationships among all domains of life through richer web and programmatic interfaces. Nucleic Acids Res. **46**(D1), D477–D485 (Jan 2018)
4. Bgee team: Bgee data sources. `https://bgee.org/?page=source` (2019), accessed: 2019-04-09

5. Blomqvist, E., Presutti, V., Daga, E., Gangemi, A.: Experimenting with extreme design. In: Cimiano, P., Pinto, H.S. (eds.) Knowledge Engineering and Management by the Masses. pp. 120–134. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
6. Farias, T.M.: Sparql endpoint for accessing virtual links among bgee, uniprotkb, oma, ebi, and drugbank. http://biosoda.expasy.org:8890/sparql, accessed: 2019-04-09
7. Farias, T.M.: VoIDext git project. https://github.com/biosoda/voidext, accessed: 2019-04-09
8. Farias, T.M.: VoIDext vocabulary specification draft. https://biosoda.github.io/voidext/, accessed: 2019-04-09
9. de Farias, T.M., Stockinger, K., Dessimoz, C.: VoIDext: Vocabulary and patterns for enhancing interoperable datasets with virtual links. arXiv preprint arXiv:1906.01950v2 (2019)
10. Jacobson, K., Dixon, S., Sandler, M.: Linked-brainz: providing the musicbrainz next generation schema as linked data. In: Late-Breaking Demo Session at the 11th International Society for Music Information Retrieval Conference (2010)
11. Jupp, S., Malone, J., Bolleman, J., Brandizi, M., Davies, M., Garcia, L., Gaulton, A., Gehant, S., Laibe, C., Redaschi, N., Wimalaratne, S.M., Martin, M., Le Novère, N., Parkinson, H., Birney, E., Jenkinson, A.M.: The EBI RDF platform: linked open data for the life sciences. Bioinformatics **30**(9), 1338–1339 (May 2014)
12. Peroni, S.: A simplified agile methodology for ontology development. In: Dragoni, M., Poveda-Villalón, M., Jimenez-Ruiz, E. (eds.) OWL: Experiences and Directions – Reasoner Evaluation. pp. 55–69. Springer International Publishing, Cham (2017)
13. UniProt Consortium: UniProt: the universal protein knowledgebase. Nucleic Acids Res. **46**(5), 2699 (Mar 2018)