KNOWLEDGE GRAPHS

Semantic Data Management

Authors
Enric Reverter López
Pol Gerdt Basullas

Contents

1 Introduction

1.1 Data

The data used for this assignment is the one that one of the group members curated for the first assignment of the course. The source of the data is DBLP, from which the following files have been processed: *authors.csv*, *volumes.csv*, *editions.csv*, *journals.csv*, *conferences.csv*, *reviewers.csv*, *reviews.csv*, and *keywords.csv*. Therefore, most values were already available, but some have been synthetically generated to fulfill the problem statement. That is, paper types (*DemoPaper*, *ShortPaper FullPaper*, and *Poster*) have been randomly allocated to existing articles. The same is true for conference types (*Workshop*, *Symposium*, and *ExpertGroup*). Finally, dummy values for *Chair*, *Editor*, and *Venue* have also been allocated. Moreover, dummy attributes have been generated to meet the constraint of at least one attribute per class.

2 Ontology creation

2.1 TBOX definition

It is assumed that every *Paper* is published in some Venue so it has at least two reviews from reviewers assigned by a *Chair* or *Editor*. This does not restrict the possibility that the two reviews are from the same *Reviewer* so this should be checked when creating the ABOX. It is also considered to relax this condition and allow papers that are not published by having additional subclasses of *Paper* like *PublishedPaper* and *UnpublishedPaper*, where *PublishedPaper* would have the relations to *Review*, *Venues*, and so on. However, this would add unnecessary complexity that was not required for the specific TBOX definition in this assignment.

2.2 ABOX definition

The ABOX has been generated using the `rdflib` library available in *Python*. To do so, the non-semantic data described in Section 1.1 is iterated and parsed into N-Triples format. Since the TBOX definition is known, the naming convention while generating the ABOX is set the same, so the subsequent linking can be properly achieved. It is worth mentioning that the files are structured in a relational style. That is, *papers.csv* contain information for each article and are related to either a volume (journal) or edition (conference) through a foreign key. The same is true for the rest of features.

First, the `namespace` for the ontology and RDF graph are initialized. Then, each file is loaded and iterated. Regarding the explicit definition, it is assumed that most super classes will be inferred by the entailment regime. For example, when loading the papers, only the subclass is initialized as the relation to the class `Paper` should be automatically implied. Likewise, `Conference`, `Venue` `ConferencePaper`, `GeneralPaper`, and `Person` should be defined by the chosen ruleset.

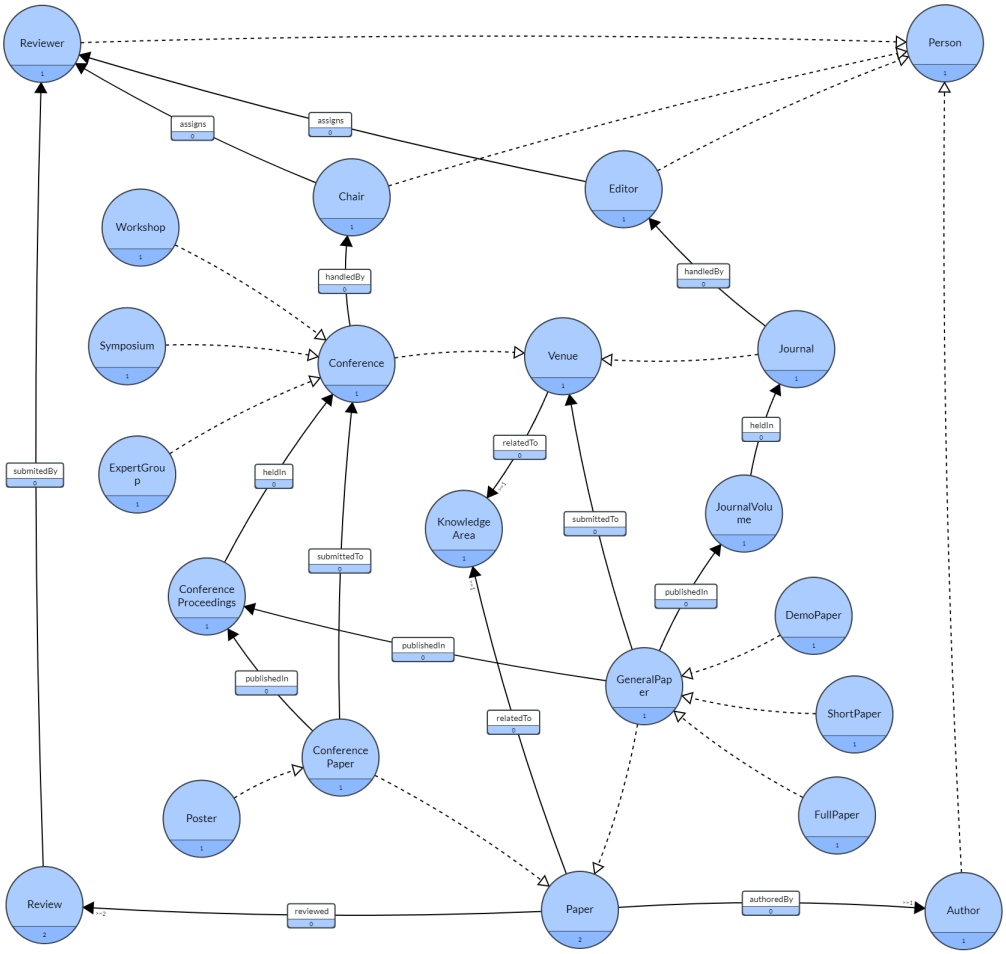

Figure 1: TBOX graphical representation

Main classes	Count
Distinct classes	**25**
Paper	998
Person	7474
Venue	8
Proceeding	66
Volume	158
Area	50
Review	2994

Table 1: Count for some of the main classes. Only 22 of out the 25 unique properties are from the custom `sdm` ontology.

2.3 Final ontology

Once the TBOX and ABOX have been curated, the linking is made by the same `rdflib` library by means of the `serialize` method. Both sources are then parsed and stored in a single N-Triples file. Such file is then uploaded to *graphDB*, where the inference takes place.

The chosen entailment regime for the article is owl2-ql, which allows the utilization of the OWL2 specification, such as `owl:minQualifiedCardinality`. A few of the OWL2 capabilities are being used, and for the current simple ontology, a simpler ruleset like owl-max or even only rdfs (with some modifications) could be used. However, the additional computational cost may be justified in the future if there is a need to express more complex semantics.

Thanks to reasoning with `rdfs:subClassOf`, there is no need to explicitly specify the parent classes, such as `Paper`, for all the types of papers like `DemoPaper`, `ShortPaper`, and so on. Similarly, the `Conference` type does not need to be defined for all conference sub-classes like `Symposium` and `Workshop`. In our case, defining a `Poster` automatically assigns it the type `ConferencePaper`, with the constraint of being published to a `Conference`, as well as the general type `Paper`. The same applies to `Reviewer`, `Chair`, `Editor`, and `Author`, which are automatically classified as `Person` and share common properties like `name`. Finally, `Journal` and `Conference` are designated as `Venue` and are automatically required to be associated with a `KnowledgeArea`.

The original number of statements is 61,349. After inference, the number goes up to 95,459 (i.e., 34,110 are being inferred). This represents an expansion ratio of 1.56. Tables 1 and 2 depict the count of the main classes and properties defined by the TBOX and ABOX. It is worth mentioning that the counts are coherent with the source data. For instance, the average number of authors and areas per article are approximately 4 and 5, respectively, which is on line with the counts.

Main properties	Count
Distinct properties	**51**
authoredBy	4356
assigns	7472
heldIn	224
publishedIn	998
relatedTo	5049
reviewed	2994

Table 2: Count for some of the main properties. Only 28 of out the 51 unique properties are from the custom `sdm` ontology.

2.4 Querying the ontology

The prefix required to query the defined ontology is named `sdm` throughout the section.

2.4.1 Find all Authors

The query simply returns those instances where the object of a given type is from the `Author` class. It does so by using the ABOX definition.

```
PREFIX sdm: <http://www.sdm.com/ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?Authors WHERE {
    ?Authors rdf:type sdm:Author
}
```

2.4.2 Find all properties whose domain is Author

The query first returns all the properties which domain is `Author` by means of the TBOX definition. Then adds to it those properties that might be indirectly linked. Again, making use of the TBOX.

```
PREFIX sdm: <http://www.sdm.com/ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?properties WHERE {
    {
            ?properties rdfs:domain sdm:Author
    }
    UNION
    {
        sdm:Author rdfs:subClassOf* ?sup_class .
        ?properties rdfs:domain ?sup_class
    }
}
```

2.4.3 Find all properties whose domain is either Conference or Journal

Same idea as in the previous query. Now the query traverses through subclasses instead.

```
PREFIX sdm: <http://www.sdm.com/ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?properties WHERE {
    {
            ?properties rdfs:domain sdm:Venue
    }
    UNION
    {
        ?sub_class rdfs:subClassOf* sdm:Venue .
        ?properties rdfs:domain ?sub_class
    }
}
```

2.4.4 Find all the papers written by a given author that where published in database conferences

The query first fetches those instances where the `name` property from `Author`, which is actually defined in the super class `Person`, conforms with the desired author (e.g., *Hung Bui*). That is, both TBOX and ABOX are being used. Then, a similar pattern is applied to fetch the papers published in conferences that are mostly related to the desired knowledge area (e.g., *Databases*).

```
PREFIX sdm: <http://www.sdm.com/ontology#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?author (group_concat(distinct ?paper;separator="\n") as ?papers) ?conference
WHERE {
    ?paper sdm:authoredBy ?author .
    ?author sdm:name "Hung Bui" .
    ?paper sdm:publishedIn ?edition .
    ?edition sdm:heldIn ?conference .
    ?conference rdf:type sdm:Conference ;
                sdm:relatedTo ?area .
    ?area sdm:name "Databases" .
}
GROUP BY ?author ?conference ?area
```