Knowledge Graphs
Open Data

Arnau Abella
Lluís Grífols

May 25, 2021

1 Introduction

The project is publicly available at Github.

2 TBOX definition

In order to create our ontology we decided to use *OWL* since our ontology required restrictions which cannot be expressed with *RDFS*.

To model our ontology, we represented the problem statement as a UML diagram (see fig. 2) and then we translated this UML to OWL.

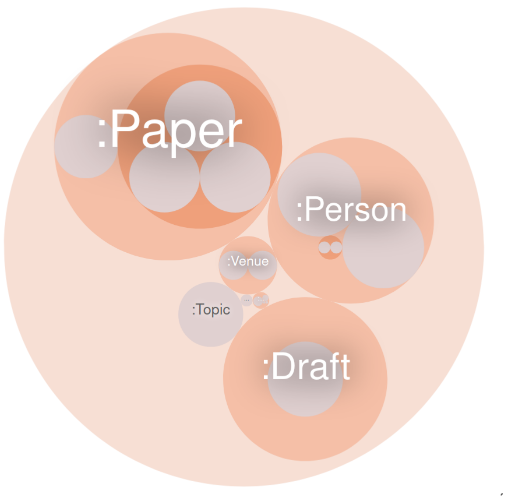

Figure 1: TBOX Graph Representation

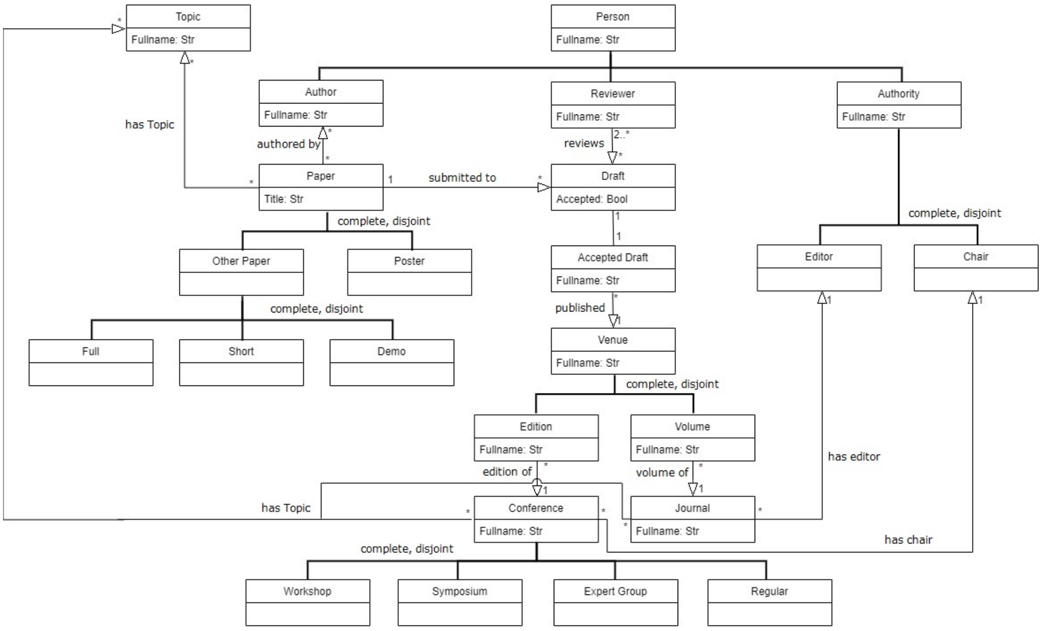

Figure 2: UML Class Diagram

For this project, we have used the programming language *Python* and Python's library *Owlready2* [1]. Owlready2 is an eDSL for Python which translates Python's syntax to OWL's definitions using the following rules:

- `owl:Class` and `rdf:Property` is defined using a python's class definition.
- `rdfs:subClassOf` and `rdfs:subPropertyOf` is defined using Python's inheritance.
- `owl:ObjectProperty`, `owl:DataProperty` and `owl:FunctionalProperty` are defined using class inheritance from Python's classes `ObjectProperty`, `DataProperty` and `FunctionalProperty`, respectively.
- `rdfs:domain` and `rdfs:range` is defined using a class attribute.
- `owl:Restriction` is also defined using a class attribute.

As an example of our ontology, the class *AcceptedDraft* is defined as follows

```
<owl:Class rdf:about="#AcceptedDraft">
  <rdfs:subClassOf rdf:resource="#Draft"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#published"/>
      <owl:onClass rdf:resource="#Venue"/>
      <owl:qualifiedCardinality
        rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">1</owl:qualifiedCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="#Draft"/>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#accepted"/>
          <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean">true</owl:hasValue>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
```

Listing 1: AcceptedDraft Class definition

The code to generate the ontology and the ontology itself can be found at `OD-E_B1_AbellaGrifols.py` and `OD-E_B1_AbellaGrifols.rdf`, respectively.

3 ABOX definition

For the ontology's ABOX, we have used data extracted from DBLP [2]. The missing classes such as paper and workshop subtypes were assigned following a normal distribution from a synthetic dataset.

For this part, we have also used Owlready2. Owlready2 has a unique approach to ABOX generation. Once an ontology is loaded into Python's runtime, all elements from the ontology are available as Python's constructs such as classes or class attributes. The user only has to instantiate those classes and attributes in order to generate the corresponding ABOx. For example, a `publications:Draft` is instantiated as follows:

```
draft = Draft("draft_1")
draft.isReviewedBy = [reviewer1, reviewer2]
draft.supervisedBy = [chair]
draft.accepted = True
draft.published = [volume]
```

Listing 2: ABOX programatically definition

The code and ontology instances for this section can be found at `OD-E_B2_AbellaGrifols.py` and `OD-E_B2_AbellaGrifols.rdf`, respectively.

4 Create the final ontology

An OWL DL reasoner was capable of infering the following knowledge from our ontology:

- A Person is infered to be an Author/Reviewer/Editor/Chair depending on the relationship domain/range.
- A Venue is infered to be a Conference/Journal depending on the relationship domain/range.
- A Draft is infered to be AcceptedDraft depending on the Datatype property 'accepted'.
- A Paper is infered to be a Poster if it is presented in a Conference.
- Inverse properties such as AuthorOf/AuthoredBy are infered.

As an extra, Owlready2 is equipped with a powerful reasoner engine with allowed us to check the correctness of our ontology and to verify if our instances were following the restrictions of our ontology. Owlready2 also provides an SPARQL engine, which combined with the provided reason, the user can test if the ontology is sound and the inference is working as expected.

Number of Classes	115
Number of Properties	90

Table 1: Ontology statistics

We provide a summary table of the resulting knowledge graph (see tables 1, 2, 3)

Paper	OtherPaper	Full	4335
		Demo	4198
		Short	4292
	Poster		1815
			14640
Person	Authority	Chair	3
		Editor	3
	Author		20317
	Poster		15846
			25614
Venue	Edition		30
	Volume		30
		Total Venue	60
Draft	AcceptedDraft		7283
			14640
Conference	Regular Conference		2
	Expert Group		1
	Symposium		0
	Workshop		0
			3
Journal			3
Topic			2139

Table 2: Classes breakdown

authorOf	38765
chairOf	5
editorOf	4
draftOf	14640
responsibleFor	21960
editionOf	30
volumeOf	30
hasTopic	36629
published	7283
reviews	24808

Table 3: Properties breakdown

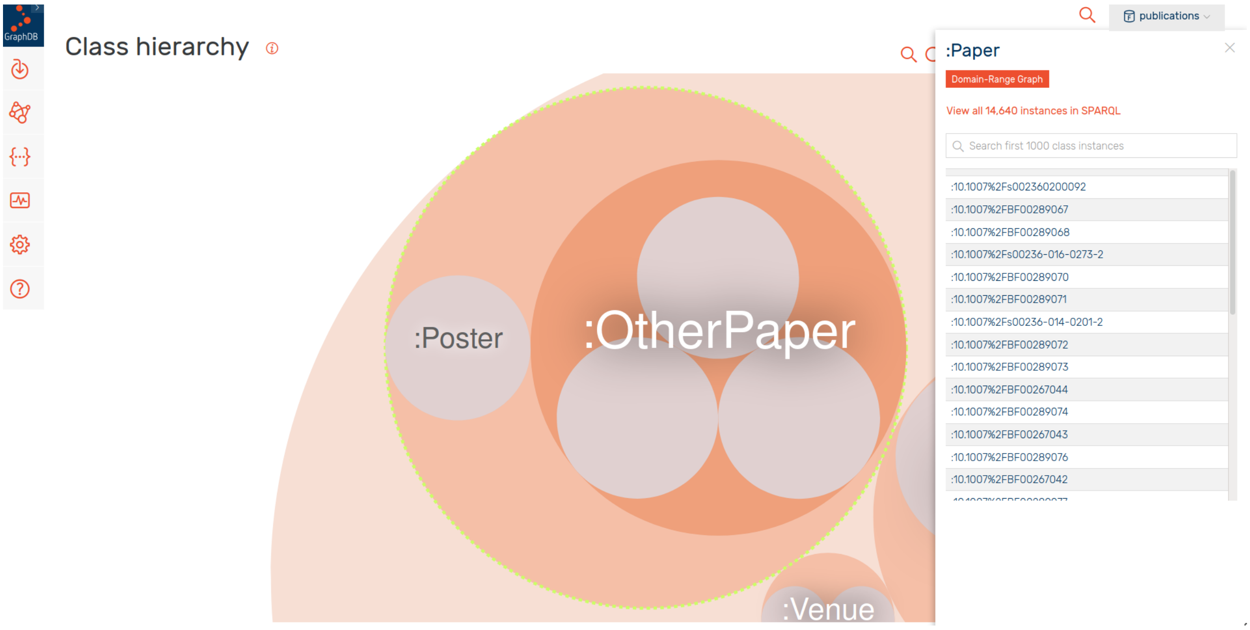

Figure 3: GraphDB instances

5 Querying the ontology

5.1 Find all Authors

```
PREFIX : <http://localhost:7200/publications#>

select ?name where {
  ?author a :Author .
  ?author :fullname ?name
} limit 100
```

5.2 Find all properties whose domain is Author

```
PREFIX : <http://localhost:7200/publications#>
PREFIX rdf: <http://www.w3.org/2000/01/rdf-schema#>

select ?p where {
  ?p rdf:domain :Author .
} limit 100
```

5.3 Find all properties whose domain is either Conference or Journal

```
PREFIX : <http://localhost:7200/publications#>
PREFIX rdf: <http://www.w3.org/2000/01/rdf-schema#>

select distinct ?p where {
  {?p rdf:domain :Conference}
  UNION
  {?p rdf:domain :Journal}
} limit 100
```

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

```
PREFIX : <http://localhost:7200/publications#>
PREFIX rdf: <http://www.w3.org/2000/01/rdf-schema#>

select ?author (group_concat(?title) as ?titles) where {
    ?author :authorOf ?paper .
    ?paper :title ?title .
    ?paper :submittedTo ?draft .
    ?draft a :AcceptedDraft .
    ?draft :published ?venue .
    ?venue a :Edition .
    ?venue :hasTopic ?topic .
    ?topic :fullname ?label .
    filter contains(?label, "big data")
}
group by ?author
limit 100
```

Notice, we changed *"database"* for *"big data"* because our dataset does not include *"database"* as a topic.

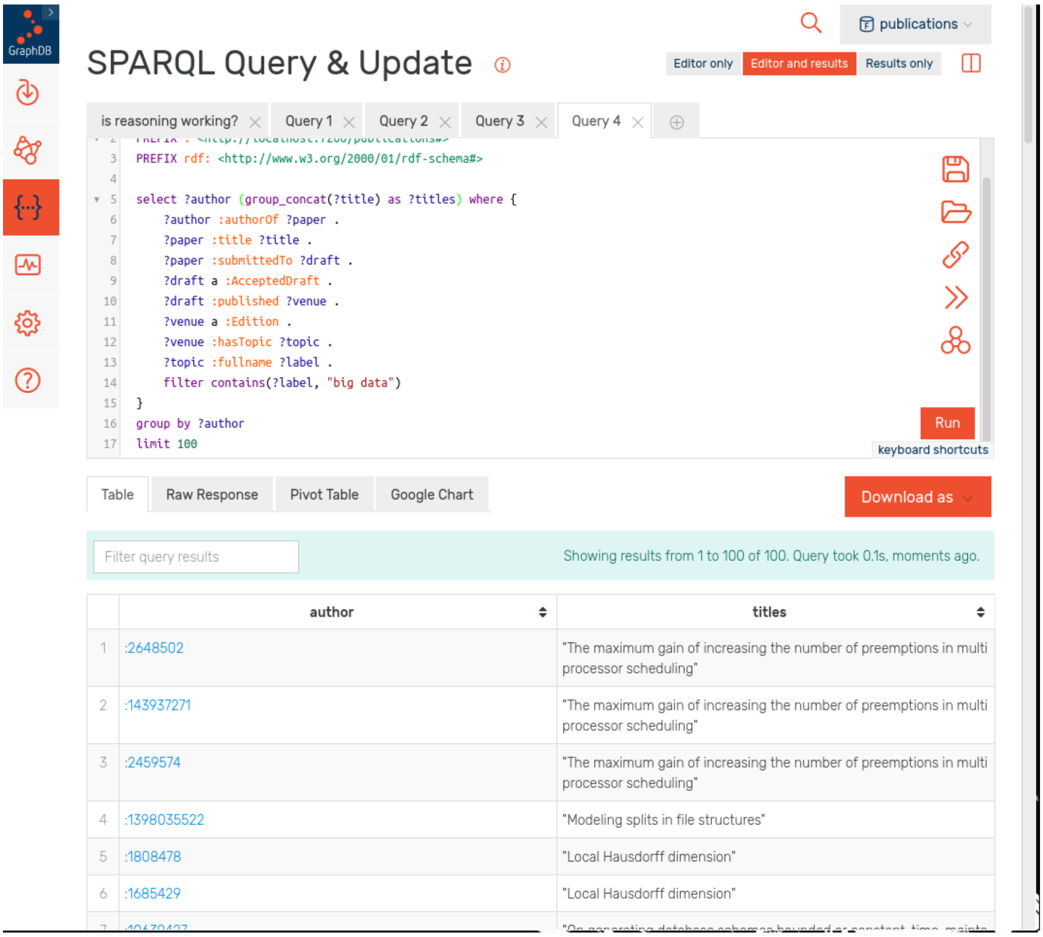

Figure 4: SPARQL: Query 4

References

[1] Lamy JB. *Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies.* **Artificial Intelligence In Medicine 2017**;80:11-28

[2] The dblp team: dblp computer science bibliography. Monthly snapshot release of March 2021. https://dblp.org/xml/release/dblp-2021-03-01.xml.gz