Knowledge Graphs Lab

Semantic Data Management

Lab Report

Víctor Diví

Pietro Ferrazzi

May, 2022

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

B.1: TBOX Definition

The TBOX has been generated programmatically using the Jena API, the code can be found in the file "`Pietro Victor-B1-FerrazziDivi.java`" or inside the executable project as the `TBox` class.

The TBOX generated uses several OWL elements and concepts, some of which may be out of scope for the project. Since they had been already implemented by the time that fact came to knowledge, they are included in the TBOX and in this report. However, the generating code is split in two parts, one that builds the base model, and another that extends it with OWL restrictions. In particular, the base model uses the following OWL elements:

- `owl:DatatypeProperty`
- `owl:ObjectProperty`
- `owl:inverseOf`

Any other OWL element used in the TBOX is added in the second part.

Two Turtle files are included in the project regarding this section:

- Base TBOX: "`Pietro Victor-B1-FerrazziDivi-Base.ttl`"
- Extended TBOX: "`Pietro Victor-B1-FerrazziDivi-Extended.ttl`"

Figure 1 below shows a simplified version of the class diagram implemented (in particular, Datatype Properties are not being shown).

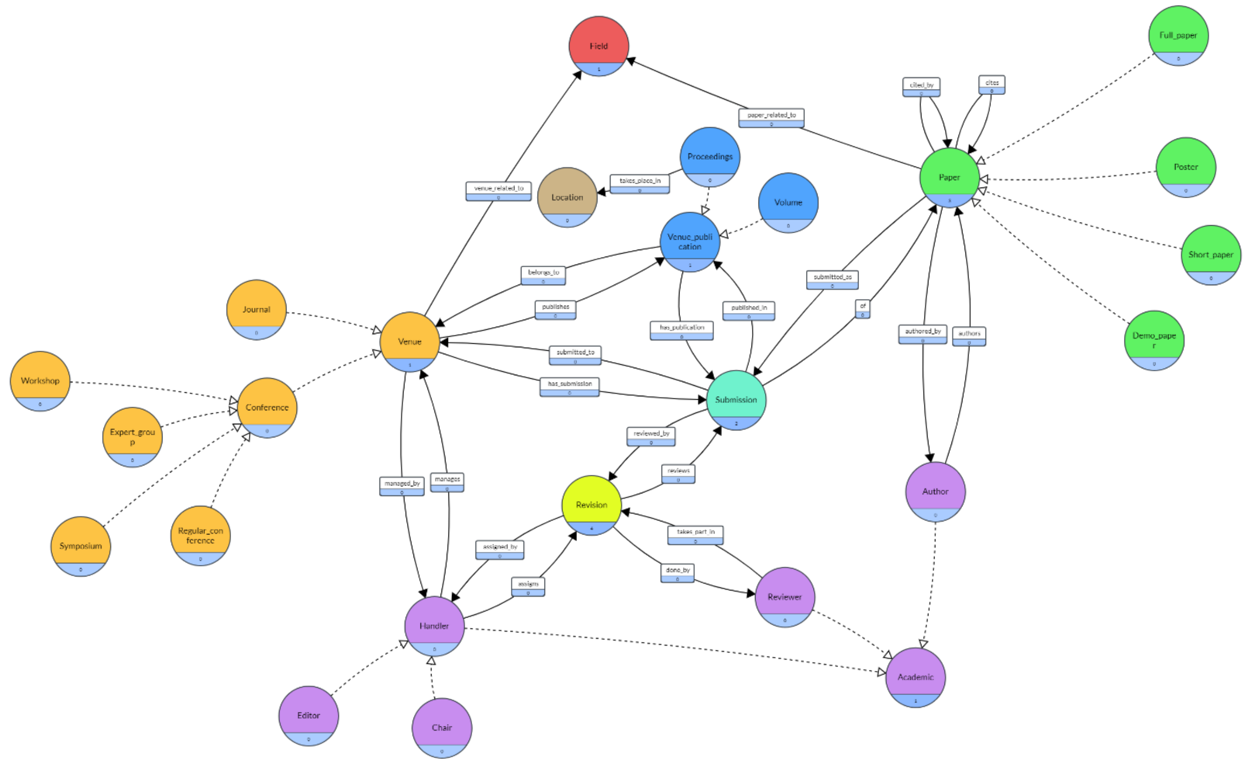

Figure 1 TBOX class diagram (Classes and Object Properties only)

As can be seen by the colors, we have 4 hierarchies: Paper, Academic, Venue, and Venue Publication; and 4 single classes: Revision, Submission, Field and Location. All classes and properties are local (prefix https://ferrazzi.divi/#), except for Location, which is from the dbpedia ontology (http://dbpedia.org/Ontology/Location).

Table 1 lists all the Object Properties shown previously in the diagram, showing also the relationship between them (inverse properties), as well as the cardinality of each one (the slash separates the cardinality of the property and the cardinality of the inverse).

Property	Domain	Range	inverseOf	Cardinality
authors	Author	Paper	authored_by	≥1 / ≥1
cites	Paper	Paper	cited_by	- / -
submitted_as	Paper	Submission	of_paper	- / =1
submitted_to	Submission	Venue	has_submission	=1 / -
published_in	Submission	VenuePublication	has_publication	≤1 / -
belongs_to	VenuePublication	Venue	publishes	=1 / -
manages	Handler	Venue	managed_by	≥1 / ≥1
assings	Handler	Revision	assigned_by	- / ≥1
done_by	Revision	Reviewer	takes_part_in	≥2 / ≥1
reviews	Revision	Submission	reviewed_by	≥1 / =1
paper_related_to	Paper	Field	-	≥1
venue_related_to	Venue	Field	-	≥1
takes_place_in	Proceedings	Location	-	≥1

Table 1 List of Object Properties

Table 2 lists all the Datatype Properties of the TBOX that are missing in Figure 1 (following the same color code). The last column shows the cardinality of each property. Note that we accept having multiple values of most textual properties to be able to have translations.

Domain	Property Name	Range (XSD)	SubPropertyOf	Cardinality
Academic	name	String	rdfs:label	≥1
Paper	doi	String	-	≤1
Paper	title	String	rdfs:label	≥1
Paper	abstract	String	rdfs:comment	≥1
Submission	submission_date	Date	-	=1
Submission	submission_accepted_date	Date	-	≤1
Revision	accepted	Boolean	-	≤1
Revision	review_text	String	rdfs:comment	-
Revision	revision_date_start	Date	-	=1
Revision	revision_date_end	Date	-	≤1
Field	keyword	String	rdfs:label	≥1
Venue	venue_name	String	rdfs:label	≥1
VenuePublication	year	Year	-	=1
Volume	volume_number	Integer	-	=1

Table 2 List of Datatype Properties

The following assumptions were made in the TBOX design:

- In a revision of a submission, only one review text and final decision is made. I.e., there is one review text and decision per revision, not per reviewer.
- At least one attribute has been added per hierarchy, meaning that each concept is in the domain of at least one attribute, although it may not be directly (e.g., Handler has no attributes, but Academic does).
- Conferences may take place in different locations each year, and therefore that location is related to the Proceedings of that Conference.

In addition to the restrictions defined by the cardinalities specified in Table 1 and Table 2, the following restrictions are implemented as OWL restrictions, and therefore, the violation of any of them would make the Ontology inconsistent:

- The seven main classes (Academic, Venue, VenuePublication, Paper, Submission, Revision and Field) are pairwise disjoint.
- Academic subclasses are complete (i.e., Academic is equivalent to the union of Author, Reviewer and Handler).
- Handler subclasses are complete.
- Paper subclasses are complete and pairwise disjoint.
- Venue subclasses are complete and pairwise disjoint.
- Conference subclasses are complete and pairwise disjoint.
- VenuePublication subclasses are complete and pairwise disjoint.

- Journals (Conferences) can only publish Volumes (Proceedings). Conversely, Volumes (Proceedings) can only belong to Journals (Conferences).

The following restrictions were identified as necessary for a semantically correct Ontology, but they are not implemented, so violating any of them would not generate an inconsistency:

- Posters can only be submitted/published in Conferences
- A Submission can only be published if it is accepted by the Revision
- A Paper can only be published once
- The dates related to Submission and Revision must be coherent (Submission date before revision start, revision start before revision end, and revision end before submission acceptance date)

It's possible that more restrictions could be found, but since they have not been identified, they are considered out of scope for this project.

B.2: ABOX Definition

The ABOX has been generated programmatically with Jena API (like the TBOX), using files originally created for the 1st Lab of SDM (Property Graphs Lab). We refer to "Annex 1: Extract from Lab 1", for an explanation on how these files were created, although a brief explanation of the files used and their relevant fields can be found in this section.

Since the ABOX has been created with the same tool as the TBOX, the links between them have been created along with the individuals. In particular, we use `OntClass::createIndividual` to create all the individuals, which sets the type of the individual created to the class used.

The code for the ABOX creation can be found in `"Pietro Victor-B2-FerrazziDivi.java"` or inside the executable project as the `ABox` class. Note that these files contain the code for both B.2 and B.3 sections of the lab. One Turtle file is added containing the output of the program (using the extended TBOX) named `"Pietro Victor-B2-FerrazziDivi.ttl"`.

The following subsections explain the 4 different files used.

Data

The main CSV file, named `data.csv`, contains the major part of the information, and is used to create the Papers, Authors, Venues, VenuePublications, Submissions and Fields. The following table shows the fields used and the information they contain.

Field	Meaning
Authors	Names of the authors
Author(s) ID	Ids of the authors
Title	Title of the paper

Year	Year of publication of the paper
Source title	Name of the venue that published the paper
Volume	Volume of the venue in which the paper is published
DOI	DOI of the published paper
Abstract	Abstract of the paper
Index keywords	Keywords of the paper
Document Type	Indicates whether the venue is a journal or a conference

For each row in the file, the following individuals and datatype properties are created:

- 1 Paper, with an autogenerated URN, and a title, an abstract and a DOI. If the paper is published in a conference, then one paper subclass will be randomly selected as the type of the paper, the same happens if it is published in a journal, but without Poster as an option.
- At most 1 Author per item in “Authors”/“Author(s) ID” fields, using the ids in “Author(s) ID” as URNs, to be able to reuse them when they appear again. If a new name appears with the same id, it is added again (they are normally the same name with minor changes in spelling or abbreviations).
- At most 1 Venue, with an autogenerated URN and a name. Like the authors, venues are reused. If the venue is a journal, then the class Journal is used to create the instance, otherwise, one of the Conference subclasses is randomly selected and used. When the venue is created, three Handlers of the corresponding subclass (Chair for Conferences, Editor for Journals) are created with an autogenerated URN.
- At most 1 VenuePublication, with a URN that depends on the Venue URN and the Volume/Year. Like the venues, they are reused. If the venue is a journal, then the Volume class will be used to create the instance, and a volume number will be added. If the venue is a conference, the Conference will be used instead, and the instance will be related to a Location instance (:Barcelona is used for every one).
- 1 Submission, with an autogenerated URN, and related to the paper, venue and venue publication through the corresponding properties.
- At most 1 Field per element in “Index Keywords” field, with a keyword. Both the paper and venue are related to each field through the corresponding properties.

Citations

The `citations.csv` file contains pairs of papers, meaning that, for each pair, the first cites the second. Therefore, for each row in the file, a property `:cites` is created between the first and second paper.

Reviews and reviewers

The revision information is contained in two files:

- `reviewers.csv`: Contains pairs of Paper and Authors indicating that said Authors review said Paper (note that in the resulting graph, all Reviewers will also be Authors).
- `reviews.csv`: Contains, for each Paper-Reviewer pair, the decision and the review text.

To create the Revisions, first reviews.csv is iterated and, for each paper (not for each row, only the first row that has a positive outcome for each paper is used), a revision is created, with a review text and an accepted value (which is always true), and it is related to the submission of the paper and to a randomly selected handler of the venue to which the paper is submitted through the corresponding properties.

The, for each row in `reviews.csv`, the revision is related to the reviewers through the `:done_by` property.

B.3: Ontology Overview and Statistics

For the Ontology generated, an OWL-QL entailment regime is considered, which provides the expressive power to represent the cardinality constraints and the other restrictions used, and allows us to skip the explicit generation of the following properties:

- `rdfs:label` of Academic, Venue, Field and Paper instances, via the `rdfs:subPropertyOf` from `:name`, `:venue_name`, `:keyword` and `:title` respectively.
- `rdfs:comment` of Paper and Revision instances, via the `rdfs:subPropertyOf` from `:abstract` and `:review_text` respectively.
- `rdfs:type` via `rdfs:subClassOf` of the following instances:

Explicit class	Inferred classes
Chair, Editor	Handler, Academic
Author, Reviewer	Academic
Volume, Proceedings	VenuePublication
Journal	Venue
Workshop, Symposium, Expert Group, Regular Conference	Conference, Venue
Full Paper, Short Paper, Demo Paper, Poster	Paper

Table 3 Inferred classes via the rdfs:subClassOf property

- `rdfs:type` of Reviewer is inferred when creating a `:done_by` property between the individual (which at the moment were only of type Author) and a Revision instance.

- All properties that have an inverse (see Table 1 above) are just created in one of the directions. The inverse property is inferred thanks to the `owl:inverseOf` property.

Table 4 shows for each class, its number of instances. We can see that for disjoint hierarchies, all the numbers add up (e.g., in Venue Publications: 332 + 463 = 795).

Class	Number of Instances
Academic	2652
Author	1743
Reviewer	1519
Handler	909
Chair	315
Editor	594
Paper	2660
Full Paper	803
Short Paper	766
Demo Paper	828
Poster	263
Submission	2660
Revision	2660
Venue	303
Journal	198
Conference	105
Expert Group	33
Symposium	22
Workshop	29
Regular Conference	21
Venue Publication	795
Proceedings	463
Volume	332
Field	6555
Location	1

Table 4 Classes and their number of instances

Table 5 and Table 6 show the same concept, but with object properties and datatype properties respectively. In addition, the average number of properties per Individual of their domain is also shown.

Property	Inverse	Times used	Average	
paper_related_to	-	41804	15.72	
cites	cited_by	32969	13.21	14.1

authors	authored_by	12172	6.98	4.57
venue_related_to	-	9435	31.14	
takes_part_in	done_by	7980	5.25	3
assigns	assigned_by	2660	3.33	1
submitted_as	of_paper	2660	1	
published_in	has_publication	2660	1	3.34
reviews	reviewed_by	2660	1	
submitted_to	has_submission	2660	1	8.79
manages	managed_by	909	1	3
publishes	belongs_to	795	2.62	1
takes_place_in	-	463	1	

Table 5 Object Properties and the number of times they are used

Property	Times used	Average
keyword	6555	1
abstract	2660	1
title	2660	1
doi	2660	1
accepted	2660	1
submission_date	2660	1
submission_accepted_date	2660	1
takes_place_in	463	1
name	1848	1.06
year	1791	1
revision_date_start	2660	1
revision_date_end	2660	1

Table 6 Datatype Properties and the number of times they are used

B.4: Ontology Querying

All following queries can be found in "`Pietro Victor-B4-FerrazziDivi.sparql`" and make use of the following prefixes:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX fd: <https://ferrazzi.divi/#>
```

1. Find all Authors.

```
SELECT ?person WHERE {
      ?person rdf:type fd:Author
}
```

2. Find all properties whose domain is Author.

```
SELECT ?property WHERE {
      ?property rdfs:domain fd:Author
}
```

3. Find all properties whose domain is either Conference or Journal.

```
SELECT ?property WHERE {
      {?property rdfs:domain fd:Conference}
      UNION
      {?property rdfs:domain fd:Journal}
}
```

4. Find all the papers written by a given author that were published in database conferences

```
SELECT ?paper WHERE {
      ?paper rdf:type fd:Paper ; #*
             fd:authored_by fd:<insert authorId here> ;
             fd:submitted_as ?submission .
      ?submission rdf:type fd:Submission ; #*
                  fd:published_in ?proceedings .
      ?proceedings fd:belongs_to ?conference .
      ?conference rdf:type fd:Conference ;
```

```
                    fd:venue_related_to ?field .
            ?field fd:keyword “databases” .
        }
```

In this query, the lines marked with an `*` are not really needed, since they are redundant with the properties also queried (`fd:Paper` is the domain of both `fd:authored_by and fd:submitted_as`, and `fd:Submission` is the range of `fd:submitted_as` and domain of `fd:published_in`).

Annex 1: Extract from Lab 1

We obtained the base for our data from the BYU Engineering Publications in Scopus 2017-21 Kaggle dataset[1] as a csv file, which presents data about publications and already provided real data regarding authors, publication title, publication year, volume, DOI, access link, author's affiliations, keywords, document type (if article, conference paper or other), and publication stage. The Python script generate_data.py processes this file with the following steps:

1. We only used articles and conference papers.
2. We discarded data that does not have one of the fields needed.
3. We removed the edition information such as year and edition number from the conference names
4. We parsed the author's affiliations and generated a separate csv file with them.
5. Because we lacked sufficient papers in the same conference but in four different years (for one of the queries requested), we also synthetically created extra papers for the previous and/or next years.
6. To have conferences/journals belonging to the database community (for section D), we selected the 15 conferences/journals with more publications, and added one of the community keywords to each of them.
7. We synthetically created the citation links between articles and stored them in a separate csv file. For creating such citations, we made sure that the papers being cited were from a previous year and had at least one keyword in common with the paper citing them. A paper could have from 0 to 20 citations.
8. Since we didn't have reviewer data, we chose up to 3 authors that have at least one keyword in common with the paper and assigned them as reviewers, saving this information in a separate csv file. We have also generated some text for the reviews themselves and stored them as a separate csv. Since we assumed all publications would be accepted in the end, there can be up to 1 reviewer rejecting each of them.

[1] https://www.kaggle.com/dpixton/byu-engineering-publications-in-scopus-201721/version/1