Knowledge Graphs Lab

Semantic Data Management

Monday, 23 May 2022

Luiz Fonseca

Zyrako Musaj

Margarita Hernández Casas

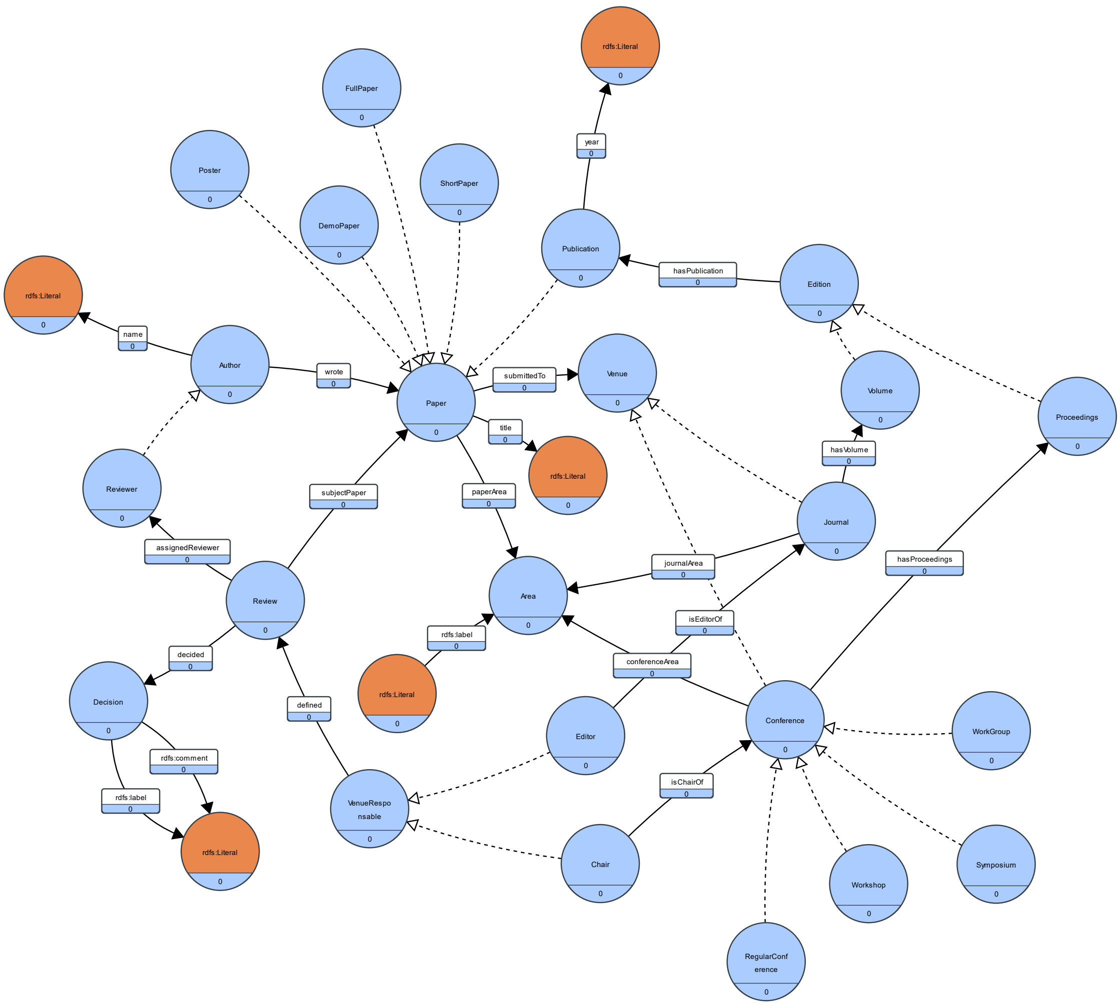
Table of Contents

B. Ontology Creation

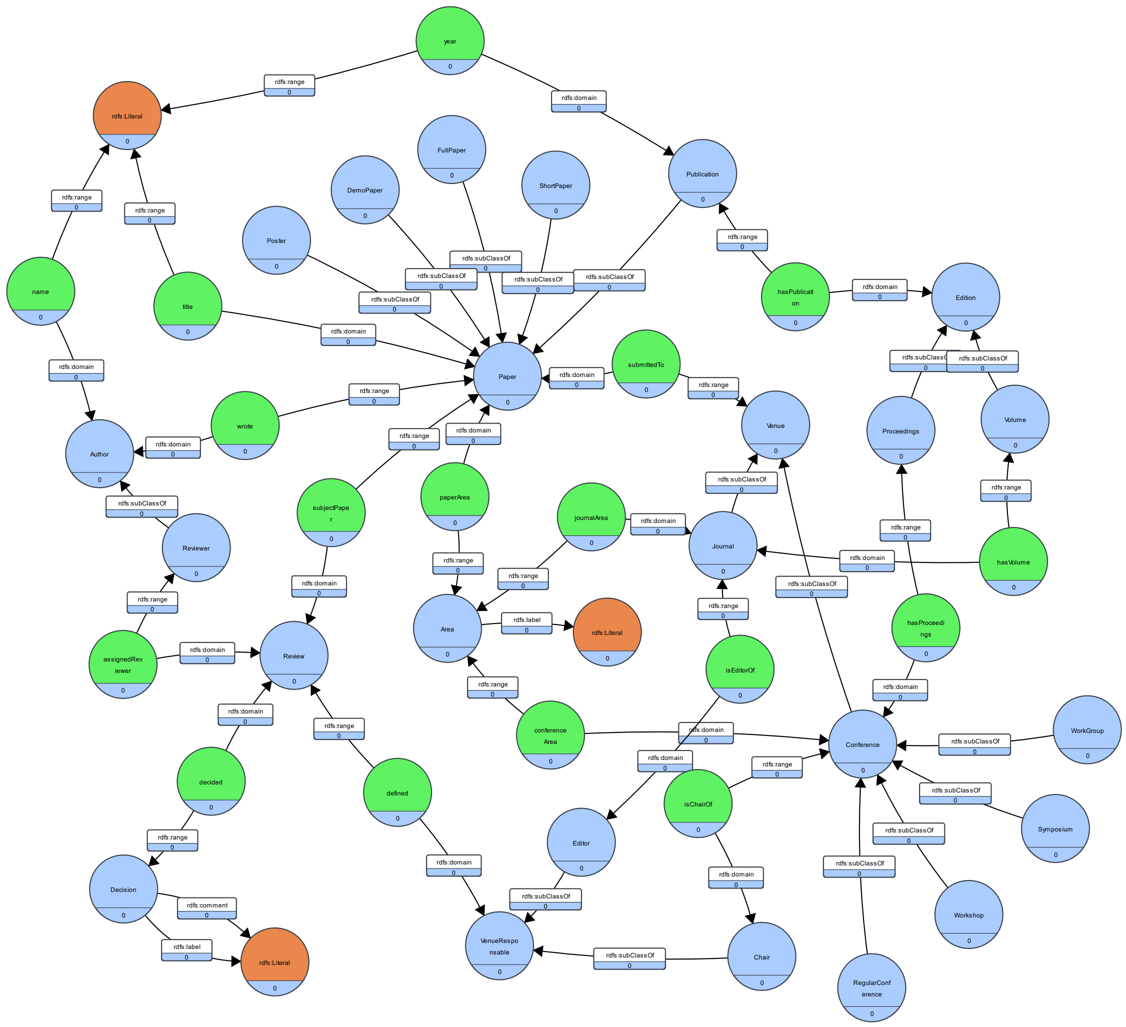
B.1. TBOX Definition

Find below our model. We have two versions of the same model. The first version is simplified and, thus, easier to visualize. The second version is more descriptive and specifies all the RDFS statements used in the TBox like rdfs:range, rdfs:domain and rdfs:subClassOf.

1. Version 1 (simplified)



2. Version 2



Legend:

Blue nodes are of type rdfs:Class

Green nodes are of type rdfs:Property

Red nodes are of type rdfs:Literal

The images are available in better quality in the links below:

[Click here to see version 1.]()
[Click here to see version 2.]()

3. Clarifications about the model

- The class **Venue** condenses common properties of both **Conference** and **Journal**. For instance, a **Paper** may be submitted to either a **Conference** or a **Journal.** So, to avoid creating 2 different properties, we created only 1 (submittedTo) with the object of that property being **Venue.**
- The same logic above applies to **VenueResponsible** and **Edition**. Since both **Chair** and **Editor** can define a **Review**, we merged this common behaviour into a single node called **VenueResponsible**, making good use of inheritance. The same happens with **Volume** and **Proceedings**. Both have publications, so we condensed this common behaviour in

the class **Edition**, considering that a **Volume** is like an edition of a **Journal** and a **Proceedings** is like an edition of a **Conference**.

- We added **Publication** as a subclass of **Paper** since it has all characteristics of a **Paper** plus the attribute **Year** (the year when the paper was published). Since in RDFS one instance may belong to more than 1 class, there's no problem in a paper being a **ShortPaper** and a **Publication** at the same time, for instance.
- To model the non-binary relationship between **Paper**, **Reviewer** and **VenueResponsible**, we created the class **Review**. The **Review** also has as outcome a **Decision.**
- We explicitly depicted some obvious RDFS statements like rdfs:label and rdfs:comment to make clear that some nodes like **Decision** will have those properties defined since we need a description and a label for some of the nodes.
- All the nodes labelled as rdfs:Literal belong to the same class (rdfs:Literal) even though they're depicted as several distinct nodes. We did this to make it easier to visualize the model.

4. Generating the TBox programmatically

We generated the Tbox using the Java library RDF4J. We have 1 single java project for both Tbox and Abox. All the code is in the zipped file named **BDMAG12C-FonsecaHernandezMusaj.zip** and you can also access the code in this github repository. You can install the required libraries via Maven using the pom file provided with the code.

The code is divided into three classes:

- Main (Main.java)
- Tbox (Tbox.Java)
- Abox (Abox.Java)

The Tbox and Abox classes have all the methods and attributes to create respectively the Tbox and Abox.

The Main class is responsible for connecting to GraphDB, calling the methods to create the TBox and ABox from the respective classes, writing the models in the database and finally writing the rdf files (.trix) as an output.

If you run the Java code, it automatically creates the statements in the graph for you. You wouldn't need to export the rdf files and import them in GraphDB. If you simply change the variables `GRAPHDB_SERVER` and `REPOSITORY_ID` to your own configuration, you can save the model automatically in your GraphDB repository.

The output rdf files have the extension .trix to maintain the internal subgraph structure that we created. We created one subgraph for the Tbox and one for the Abox just to facilitate the distinction. Note that they are different contexts inside the same graph and not two different graphs. The output file for this part is named **BDMAG12C-B1-FonsecaHernandezMusaj.trix**.

We used the namespace **"http://example.org/sdm#"** since we don't have a domain of our own and this KG will not be published publicly. It is just for pedagogical purposes.

B.2. ABOX Definition

1. Methodology to define ABOX from non-semantic data

For the creation of the csv files, we have used as a starting point the data extracted from the DBLP source that was used in Lab 1 for this same course. However, some changes have been made to conform to the provided statement and the requirements of RDFS.

A csv has been created for each node, except those of type **rdfs:literal**, as well as those that are inferable through the created schema (e.g. superclasses, by defining their child nodes as subclasses, it is not necessary to create a csv of their own as they will be inferred as superclasses).

- Subclasses in **Paper** and **Conference** were included as a column in the superclass csv, specifying which of their four subclasses they belong to.
- **Rdfs:literals** linked with properties such as **rdfs:label** or **rdfs:range** were also included as a column in the main class, avoiding the creation of new files. E.g. **Area.csv** consists of area id, as well as area keyword.
- Properties were created using a csv for each. In these files, only the identifiers of the classes that are part of the relationship are present.
- Information that was not available from the original source, were synthetically created. This is the case for some node identifiers, and also for **Decision** (accepted and rejected, which directly influences which papers are considered as publications), **Review Text** (supporting such decision), **Conference Type**, **Paper Type**. Furthermore, some relationships were randomly defined: Chairs and Editors responsible for Conferences and Journals, these venue responsible assigning reviewers, etc.

A total of 19 csv files were used in order to define the ABox, and can be found HERE.

2. Generating the ABox programmatically

As explained above in the section "Generating the TBox programmatically", the class Abox.java provides all the methods necessary to create the Abox. The class basically reads the csv files and create the RDFS statements for each of them.

The output rdf file (.trix) contains the Abox definition ready to be imported in GraphDB or you can simply run the code as explained before and it will be automatically saved to your repository.

B.3. Create the final ontology

We connected TBox and ABox when creating the ABox instances. The code for this part is also available in the class Abox.java. The reason for that was to avoid having to read all the csv files more than once. If we had separated the ABox creation from the ABox connection we'd have to read the files twice and the code would be longer. By doing both at the same time we gain performance and maintainability in the code.

So, the output file for the connections between ABox and TBox is the same as the one for part B2, which is the one named **BDMAG12C-B2-FonsecaHernandezMusaj.trix**.

1. Inference regime entailment

When creating the repository in GraphDB we chose the ruleset **RDFS (Optimized)**.

ATTENTION! According to the community there's a bug in the GraphDB ruleset **RDFS-Plus (Optimized)**. When using this ruleset, GraphDB wasn't able to infer the properties that could be derived from rdfs:range and rdfs:domain. This type of inference only worked when we used the ruleset **RDFS (Optimized)**. Thus, it is recommended to use this ruleset to have the fully functional inference for rdfs:domain and rdfs:range.

By using this inference regime entailment, we avoided having to write two main types of statements that would be inferred given the TBox, which are:

1. Statements that could be inferred from **rdfs:subClassOf**
2. Statements that could be inferred from **rdfs:domain** and **rdfs:range.**

To exemplify, every time we defined something as a **Symposium** using rdf:type, this instance would automatically be defined as being of type **Conference**, since **Symposium** is a subclass of **Conference.** As we can see in the image below, the resource with id **53e1815b20f7dfbc07e8b8e9** is in a statement such as rdf:type sdm:Symposium (line 5 - context http://example.org/sdm#abox) that was defined in the context of ABox. From that, the reasoning tool inferred that this instance is also from type Conference (line 4) and since Conference is subclass of Venue, it is also inferred to be of type Venue (line 6). We can see that the context is http://www.ontotext.com/implicit, which means that those statements were inferred.

Source: http://example.org/sdm#53e1815b20f7dfbc07e8b8e9

subject | predicate | object | context | all — Explicit and Implicit — Show Blank Nodes — Download as — Visual graph

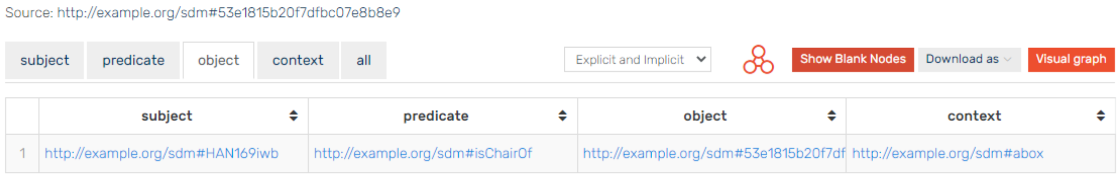
	subject	predicate	object	context
1	http://example.org/sdm#53e1815b20f7df	http://example.org/sdm#conferenceArea	http://example.org/sdm#26	http://example.org/sdm#abox
2	http://example.org/sdm#53e1815b20f7df	http://example.org/sdm#hasProceedings	http://example.org/sdm#19ab28nk	http://example.org/sdm#abox
3	http://example.org/sdm#53e1815b20f7df	http://example.org/sdm#hasProceedings	http://example.org/sdm#99ag18if	http://example.org/sdm#abox
4	http://example.org/sdm#53e1815b20f7df	rdf:type	http://example.org/sdm#Conference	http://www.ontotext.com/implicit
5	http://example.org/sdm#53e1815b20f7df	rdf:type	http://example.org/sdm#Symposium	http://example.org/sdm#abox
6	http://example.org/sdm#53e1815b20f7df	rdf:type	http://example.org/sdm#Venue	http://www.ontotext.com/implicit

An example for rdfs:range and rdfs:domain can be seen in the tables below.

Source: http://example.org/sdm#53f45728dabfaec09f209538

subject | predicate | object | context | all — Explicit and Implicit — Show Blank Nodes — Download as — Visual graph

	subject	predicate	object	context
1	http://example.org/sdm#53f45728dabfae	http://example.org/sdm#name	"Peijuan Wang"	http://example.org/sdm#abox
2	http://example.org/sdm#53f45728dabfae	http://example.org/sdm#wrote	http://example.org/sdm#53e99784b7602	http://example.org/sdm#abox
3	http://example.org/sdm#53f45728dabfae	rdf:type	http://example.org/sdm#Author	http://www.ontotext.com/implicit
4	http://example.org/sdm#53f45728dabfae	rdf:type	http://example.org/sdm#Reviewer	http://www.ontotext.com/implicit



Source: http://example.org/sdm#53e1815b20f7dfbc07e8b8e9

subject | predicate | object | context | all — Explicit and Implicit — Show Blank Nodes — Download as — Visual graph

	subject	predicate	object	context
1	http://example.org/sdm#HAN169iwb	http://example.org/sdm#isChairOf	http://example.org/sdm#53e1815b20f7df	http://example.org/sdm#abox

The instance with ID **53f45728dabfaec09f209538** was defined as the subject of the predicate **sdm:wrote**, and since the domain of **sdm:wrote** is an **sdm:Author**, the instance was inferred to be of type **sdm:Author** (line 3). This instance is also the object of a property **sdm:AssignedReviewer**, which automatically makes it a **sdm:Reviewer** (line 4). We can see that the context is http://www.ontotext.com/implicit, which means that those statements were inferred.

So, to summarize, we didn't explicitly write in the code the rdf:type of most of the nodes because the types would be inferred either by rdfs:domain, rdfs:range or rdfs:subClassOf. When you see that the context is http://www.ontotext.com/implicit, it means that the statement was inferred and not explicitly written.

2. Computing statistics about the graph

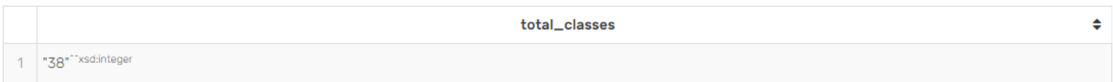
To compute basic statistics about our graph, we used SPARQL queries. Find below the queries used in each case along with a screenshot of the result.

Compute total number of classes

This includes the RDFS classes used in our model.

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

SELECT (COUNT(DISTINCT ?class) as ?total_classes)
WHERE {?class a rdfs:Class}
```



	total_classes
1	"38"^^xsd:integer

Compute top 10 classes with most instances

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

SELECT ?class (COUNT(DISTINCT ?subject) as ?count)
WHERE { ?subject rdf:type ?class }
GROUP BY ?class
ORDER BY DESC(?count)
LIMIT 10
```

	class	count
1	http://example.org/sdm#Decision	"13578"^^xsd:integer
2	http://example.org/sdm#Review	"13578"^^xsd:integer
3	http://example.org/sdm#Author	"10252"^^xsd:integer
4	http://example.org/sdm#Reviewer	"7508"^^xsd:integer
5	http://example.org/sdm#Paper	"6789"^^xsd:integer
6	http://example.org/sdm#Publication	"5150"^^xsd:integer
7	http://example.org/sdm#Edition	"3604"^^xsd:integer
8	http://example.org/sdm#VenueResponsible	"2489"^^xsd:integer
9	http://example.org/sdm#Venue	"2443"^^xsd:integer
10	http://example.org/sdm#Proceedings	"2353"^^xsd:integer

See all classes

This is not a statistic but is a summary that might be useful.

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

SELECT DISTINCT ?class
WHERE{?class a rdfs:Class}
```

sdm-lab3

15	http://example.org/sdm#Reviewer
16	http://example.org/sdm#Author
17	http://example.org/sdm#Poster
18	http://example.org/sdm#Paper
19	http://example.org/sdm#DemoPaper
20	http://example.org/sdm#FullPaper
21	http://example.org/sdm#ShortPaper
22	http://example.org/sdm#Publication

Compute total number of properties

```
SELECT (COUNT(DISTINCT ?p) AS ?total_properties)
WHERE { ?s ?p ?o}
```

	total_properties
1	"23"^^xsd:integer

Compute top 10 properties most used

```
SELECT ?p (COUNT(*) AS ?count)
WHERE { ?s ?p ?o }
GROUP BY ?p
ORDER BY DESC(?count)
LIMIT 10
```

	p	count
1	rdf:type	"82071"^^xsd:integer
2	rdfs:label	"13608"^^xsd:integer
3	http://example.org/sdm#assignedReviewer	"13578"^^xsd:integer
4	http://example.org/sdm#decided	"13578"^^xsd:integer
5	http://example.org/sdm#defined	"13578"^^xsd:integer
6	http://example.org/sdm#subjectPaper	"13578"^^xsd:integer
7	rdfs:comment	"13578"^^xsd:integer
8	http://example.org/sdm#name	"10252"^^xsd:integer
9	http://example.org/sdm#wrote	"10252"^^xsd:integer
10	http://example.org/sdm#title	"6789"^^xsd:integer

See all properties

This is not a statistic but is a summary that might be useful.

```
SELECT  DISTINCT ?p
WHERE
   { ?s ?p ?o}
```

4	rdfs:domain
5	rdfs:range
6	rdfs:comment
7	rdfs:label
8	http://example.org/sdm#name
9	http://example.org/sdm#wrote
10	http://example.org/sdm#submittedTo
11	http://example.org/sdm#title
12	http://example.org/sdm#paperArea
13	http://example.org/sdm#journalArea
14	http://example.org/sdm#conferenceArea
15	http://example.org/sdm#isEditorOf
16	http://example.org/sdm#isChairOf
17	http://example.org/sdm#defined
18	http://example.org/sdm#decided
19	http://example.org/sdm#assignedReviewer

B.4. Querying the ontology

1. Find all authors

This query obtains the names of all the authors. Since some authors are found more than once in our data, we look only for the distinct values.

```
PREFIX sdm: <http://example.org/sdm#>

SELECT DISTINCT ?name
WHERE {
        ?author sdm:name ?name .
}
```

The output, found in the figure below, is just 5 out of 10.019 different authors

	name
1	"Peijuan Wang"
2	"Jiahua Zhang"
3	"Donghui Xie"
4	"Yanyan Xu"
5	"Yun Xu"

2. Find all properties whose domain is Author.

The query:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX sdm: <http://example.org/sdm#>
SELECT ?predicate
WHERE {
        ?predicate rdfs:domain sdm:Author .
}
```

The result:

	predicate
1	http://example.org/sdm#name
2	http://example.org/sdm#wrote

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

The query:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX sdm: <http://example.org/sdm#>

SELECT ?predicate
WHERE {
    {?predicate rdfs:domain sdm:Conference} UNION
    {?predicate rdfs:domain sdm:Journal}
}
```

The result:

	predicate
1	http://example.org/sdm#conferenceArea
2	http://example.org/sdm#hasProceedings
3	http://example.org/sdm#journalArea
4	http://example.org/sdm#hasVolume

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

Since our data doesn't contain 'database' as an area for conferences, we chose another one called 'classifier'. This area had 72 different papers, out of which we filtered the ones written by the author 'Robert Fox'.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sdm: <http://example.org/sdm#>
SELECT ?title
WHERE {
    ?author sdm:wrote ?paper ;
            sdm:name ?name .
    ?paper sdm:title ?title ;
           sdm:submittedTo ?venue .
    ?venue rdf:type sdm:Conference ;
           sdm:conferenceArea ?area .
    ?area rdfs:label ?areaname .
    FILTER (?name="Robert Fox") .
    FILTER (?areaname = "classifier") .
}
```

The result of the query:

	title
1	"Digital viability."
2	"Data Emancipation."
3	"Being responsive."