UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Semantic Data Management

Lab 2: Knowledge Graph

Elnara Yerbolatova
Stephanie Silva Vieira Gomes

Gerard Pons
Oscar Romero

June 2025

TBOX Definition

1.0.1 TBOX Design

Figure 1.1 shows the design chosen to model all the requirements of the research publications graph as a knowledge graph. The image in a better quality, and all the code for this project can be found at this GitHub repository. Classes are represented in green and properties/literals in blue.

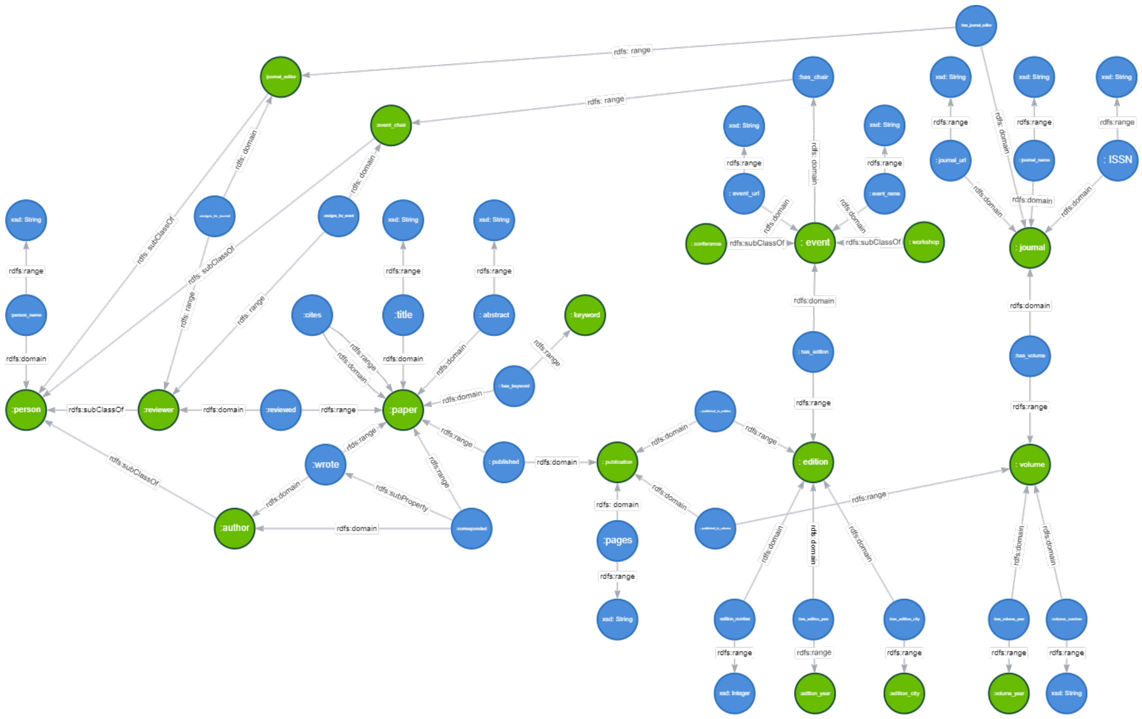

Figure 1.1: TBOX Design - Research Publication

To define the TBOX, we used the RDFLib library to apply the RDF and RDFS concepts. The following steps were taken to generate the TBOX in Python:

- **Namespace Definition**: created a personalized namespace 'http://www.example.edu/research-publication/' to uniquely identify and organize all ontology classes, properties, and instances related to the research publication domain.
- **Classes Definition**: defined all the key entities of the domain to represent as a class. Classes: person, author, reviewer, journal_editor, event_chair, paper, keyword, publication, journal, event, edition, volume, conference, workshop, edition_year, edition_city, volume_year.
- **Sub-Classes Definition**: represents entities that are specialized representations of the main class. Therefore, the subclass is used to model a more specific category within the broader concept defined by the parent class.
  Sub-classes: author, reviewer, journal_editor, event_chair -> subclass of person;
  conference, workshop -> subclass of event
- **Properties Definition (Domain and Range)**: defined properties to represent relationships and attributes in the ontology, specifying their domain and range.
  Example: Represent the property 'person_name' from a Person: `person_name` (property); `person` (domain); `xsd:string` (range).
- **Sub-Property Definition**: defined sub-properties to capture more specific relationships within the ontology. In this TBOX, we only have one sub-property: 'corresponded' is a sub-property of 'wrote'.

- **Serialization**: converted the ontology into a machine-readable format (Turtle format).

Some of the assumptions made for the TBOX definition were:

1. The 'publication' class was defined as a reification to capture the property 'pages' that are common to both editions and volumes. Therefore, we decided to centralize common properties, such as `pages`, within the `publication` class, allowing both edition and volume entities to associate with publications in a consistent manner. Moreover, having 'publication' as a separate reified node allows us to extend our knowledge graph in the future. Namely, it becomes possible to attach additional metadata to the publication such as the open_access status, DOI, exact paper's acceptance and publication dates, which are properties of the event of publication, i.e. not just properties of the paper or the publication venue.
2. edition_year, edition_city, volume_year, and keyword were defined as classes instead of literals since in this way we can allow for the same class to be referenced by more than one relationship. For example, the city `Barcelona` as an instance of `edition_city` allows it to be associated with multiple editions. This approach promotes data consistency and facilitates queries such as retrieving all events held in the same city.
3. All properties are defined with explicit domains and ranges to enforce data integrity and clarify the expected types of relationships.
4. Properties such as `wrote`, `corresponded`, and `reviewed` are used to model specific roles of persons in relation to papers, with sub-property hierarchies capturing more specialized relationships.
5. The ontology distinguishes between events (conferences, workshops) reflecting real-world publication structures and supporting hierarchical organization.

ABOX Definition

To define the ABOX, we used the RDFLib library to attach instances to classes and properties defined in the TBOX model. We read the CSV files with pandas, constructed URIs, and generated triples according to our ontology.

2.1 Data Sources

Our non-semantic data consisted of multiple CSV files, each representing entities or relationships such as:

- author.csv (authors)
- paper.csv (papers)
- review_relations.csv (reviewers and their reviews)
- paper_publishedIn_volume.csv and paper_publishedIn_edition.csv (publication events)
- journal.csv, volume.csv, edition.csv, event.csv, etc.

2.2 Entity URI Construction

For each entity (e.g., author, paper, journal), we constructed a unique URI using a base namespace and a sanitized identifier from the CSV. This ensures global uniqueness and consistency across the graph. Some examples include:

```
rp = Namespace('http://www.example.edu/research-publication/')
author_uri = URIRef(rp + 'author/'+str(row['authorId']))
keyword_uri = URIRef(rp + 'keyword/'+sanitize_uri_string(row['keyword']))
```

where sanitize_uri_string is a function that replaces spaces and special characters with underscores while keeping alphanumeric characters. As shown, unique identifiers from the sources were used (such as authorId, paperId) and, where appropriate, the strings were attached as URN (for entities, which can be shared among multiple instances, such as keyword, city).

2.3 Mapping Properties and Relationships

We mapped columns in the CSVs to ontology properties:

- **Data properties** (e.g., person_name, title, abstract, pages) became RDF literals attached to the entity URIs.
- **Object properties** (e.g., wrote, reviewed, published, published_in_volume) became RDF triples linking two entity URIs.

For example, from author_wrote_paper.csv, each row links an author to a paper via the wrote property:

```
<author/123> rp:wrote <paper/456>
```

2.4 Count of Instances and Triples

We counted triples per property and per class to verify coverage and completeness, ensuring the ABOX represents the source data as expected and aligns with the ontology. Below, we provide a summary table of the main statistics and tables highlighting some statistics for main classes and properties:

Number of Classes	17
Number of Properties	28
Number of Instances for Main Classes	439,184
Number of Triples for Main Properties	577,608
Total Number of Triples	814,349

Table 2.1: Statistics on the Knowledge Graph

Class	Count
Papers	325,397
Authors	77,614
Reviewers	24,307
Journals	2,667
Volumes	7,711
Editions	969
Conferences	504
Workshops	15
Events (total)	519

Table 2.2: Counts of individuals per class for main classes

Property	Count
wrote	94,403
corresponded	15,166
reviewed	35,633
cites	397,702
published	13,012
published_in_edition	1,475
published_in_volume	11,537
has_volume	7,711
has_edition	969

Table 2.3: Counts of main property assertions

2.5 Saving rdf:type Links

Thanks to our well-defined TBOX and RDFS inference regime entailment, it was generally unnecessary to explicitly use rdf:type to connect instances to appropriate classes or properties. For example, if an instance appears as the subject of the wrote property (where wrote is defined in the TBox with domain Author and range Paper), RDFS inference allows us to deduce that the subject is an Author and the object is a Paper.

The only exception was the need to define this link for instances of class 'conference' and of class 'workshop'. We found it unnecessary to complicate the knowledge graph with subproperties specific to each. Moreover, the classes often overlap in practice (e.g., a conference may include workshops, and the terms are sometimes used interchangeably).

Querying the Ontology

For both queries, we wanted to leverage SPARQL syntax and reasoning in knowledge graphs. Therefore, the queries were designed to explore the TBOX class hierarchy and inference.

3.0.1 Query 1: Cross-Domain Research Impact Analysis

This query identifies influential individuals who may hold multiple roles within the publication ecosystem. It counts the papers that a person wrote, corresponded or reviewed and calculates, based on a score, the most influential individuals. Furthermore, for each person, the keywords associated with their papers are retrieved to demonstrate the domain they work. The score was created giving more relevance to reviewers, the corresponding author, and the author, in this order. This was defined based on the fact that to review a paper, an individual has to be qualified within the academic world, and a corresponding author is the main contributor of a given paper.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rp: <http://www.example.edu/research-publication/>
PREFIX rdfs: <http://www.rdfs.org/2000/01/rdf-schema#>

SELECT ?personName ?authorPapers ?correspondedPapers ?reviewedPapers ?totalInfluence
    (GROUP_CONCAT(DISTINCT STRAFTER(STR(?keyword), "/keyword/"); separator=", ") AS ?researchAreas)
WHERE {
    # Find persons who are both authors and reviewers (multi-role academics)
    ?person a rp:person ;
            rp:person_name ?personName .

    # Count papers they authored (includes corresponded papers via subproperty reasoning)
    {
        SELECT ?person (COUNT(DISTINCT ?authoredPaper) AS ?authorPapers) WHERE {
            ?person rp:wrote ?authoredPaper .
        }
        GROUP BY ?person
    }

    # Count papers they corresponded for (demonstrates subproperty reasoning)
    OPTIONAL {
        SELECT ?person (COUNT(DISTINCT ?correspondedPaper) AS ?correspondedPapers) WHERE {
            ?person rp:corresponded ?correspondedPaper .
        }
        GROUP BY ?person
    }

    # Count papers they reviewed (cross-role analysis)
    OPTIONAL {
        SELECT ?person (COUNT(DISTINCT ?reviewedPaper) AS ?reviewedPapers) WHERE {
            ?person rp:reviewed ?reviewedPaper .
        }
        GROUP BY ?person
    }

    # Get research areas from authored papers
    OPTIONAL {
        ?person rp:wrote ?paper .
        ?paper rp:has_keyword ?keywordURI .
```

```
        # Extract keyword from URI (assumes keyword is in URI fragment)
        BIND(REPLACE(STR(?keywordURI), ".[/#]([^/#])$", "$1") AS ?keyword)
    }

    # Calculate total influence score
    BIND(
            (COALESCE(?authorPapers, 0) * 1) +
            (COALESCE(?correspondedPapers, 0) * 2) +
            (COALESCE(?reviewedPapers, 0) * 3)
        AS ?totalInfluence)

    # Filter for influential multi-role academics
    FILTER(?totalInfluence > 2)
}
GROUP BY ?personName ?authorPapers ?correspondedPapers ?reviewedPapers ?
    totalInfluence
ORDER BY DESC(?totalInfluence) ?personName
LIMIT 10
```

Listing 3.1: Cross-Domain Research Impact Analysis with Reasoning

Figure 3.1 is the output for the top 10 most influential individuals.

	personName	authorPapers	correspondedPapers	reviewedPapers	totalInfluence	researchAreas
1	"Jun Lyu"	"75"^^xsd:integer			"75"^^xsd:integer	"nomogram, concordan...
2	"R. Cacabelos"	"21"^^xsd:integer	"19"^^xsd:integer		"59"^^xsd:integer	"personalized_medicine, ...
3	"T. Awad"	"18"^^xsd:integer	"11"^^xsd:integer		"40"^^xsd:integer	"fragmentation, isobaric...
4	"Fengshuo Xu"	"27"^^xsd:integer	"4"^^xsd:integer		"35"^^xsd:integer	"concordance, nomogra...
5	"L. Jones"	"21"^^xsd:integer	"7"^^xsd:integer		"35"^^xsd:integer	"atherosclerotic_cardiov...
6	"O. Mutlu"	"25"^^xsd:integer	"5"^^xsd:integer		"35"^^xsd:integer	"xeon, initialization, dra...
7	"Z. Stark"	"23"^^xsd:integer	"6"^^xsd:integer		"35"^^xsd:integer	"data_sharing, genomic, ...
8	"Zhihan Lv"	"11"^^xsd:integer	"11"^^xsd:integer		"33"^^xsd:integer	"packet_loss, nosql, big_...
9	"A. Sturm"	"28"^^xsd:integer	"2"^^xsd:integer		"32"^^xsd:integer	"kexin, pcsk9, atheroscle...
10	"C. Leung"	"19"^^xsd:integer	"6"^^xsd:integer		"31"^^xsd:integer	"microdata__statistics_, ...

Figure 3.1: Top 10 Most Influential Individuals

3.0.2 Query 2: Multi-Venue Cross-Citation Analysis

In this query, we analyse the citation patterns across different types of publication venues. Here we identify authors who publish in a venue type and cite others' work that has been published in another venue type (e.g., conference-to-journal or journal-to-conference). With this query, we can identify authors with a high count of cross-citations, indicating that their research is not siloed within a single type of publication venue but instead bridges different communities.

```
PREFIX rp: <http://www.example.edu/research-publication/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?author_name ?citing_venue_type ?cited_venue_type
       (COUNT(DISTINCT ?citing_paper) AS ?papers_citing)
       (COUNT(DISTINCT ?cited_paper) AS ?papers_cited)
       (COUNT(DISTINCT ?citation_link) AS ?total_citations)
WHERE {
  # Find authors who wrote papers
  ?author rp:wrote ?citing_paper .
  ?author rp:person_name ?author_name .

  # These papers cite other papers
  ?citing_paper rp:cites ?cited_paper .
  BIND(CONCAT(STR(?citing_paper), "-", STR(?cited_paper)) AS ?citation_link)

```

```
  # Find venue types for citing papers (using reasoning but filtering out
      parent class)
  {
    ?pub1 rp:published ?citing_paper .
    ?pub1 rp:published_in_edition ?edition1 .
    ?venue1 rp:has_edition ?edition1 .
    ?venue1 rdf:type ?venue_type1 .
    ?venue_type1 rdfs:subClassOf* rp:event .
    # Filter out the parent 'event' class to get only specific subtypes
    FILTER(?venue_type1 != rp:event)
    BIND(STRAFTER(STR(?venue_type1), STR(rp:)) AS ?citing_venue_type)
  }
  UNION
  {
    ?pub1 rp:published ?citing_paper .
    ?pub1 rp:published_in_volume ?volume1 .
    BIND("journal" AS ?citing_venue_type)
  }

  # Find venue types for cited papers (using reasoning but filtering out parent
      class)
  {
    ?pub2 rp:published ?cited_paper .
    ?pub2 rp:published_in_edition ?edition2 .
    ?venue2 rp:has_edition ?edition2 .
    ?venue2 rdf:type ?venue_type2 .
    ?venue_type2 rdfs:subClassOf* rp:event .
    # Filter out the parent 'event' class to get only specific subtypes
    FILTER(?venue_type2 != rp:event)
    BIND(STRAFTER(STR(?venue_type2), STR(rp:)) AS ?cited_venue_type)
  }
  UNION
  {
    ?pub2 rp:published ?cited_paper .
    ?pub2 rp:published_in_volume ?volume2 .
    BIND("journal" AS ?cited_venue_type)
  }

  # Filter for cross-venue citations (different venue types)
  FILTER(?citing_venue_type != ?cited_venue_type)
}
GROUP BY ?author_name ?citing_venue_type ?cited_venue_type
HAVING (COUNT(DISTINCT ?citation_link) >= 2)
ORDER BY DESC(?total_citations)?author_name
```

Listing 3.2: Multi-Venue Cross-Citation Analysis with Subclass Reasoning

Figure 3.2 shows the output for the query.

	author_name	citing_venue_type	cited_venue_type	papers_citing	papers_cited	total_citations
1	"O. Mutlu"	"journal"	"conference"	"18"^^xsd:integer	"12"^^xsd:integer	"42"^^xsd:integer
2	"Yu Zheng"	"journal"	"conference"	"8"^^xsd:integer	"9"^^xsd:integer	"27"^^xsd:integer
3	"O. Mutlu"	"conference"	"journal"	"6"^^xsd:integer	"14"^^xsd:integer	"24"^^xsd:integer
4	"Andrea Bartolini"	"journal"	"conference"	"14"^^xsd:integer	"6"^^xsd:integer	"23"^^xsd:integer
5	"Ruiyuan Li"	"journal"	"conference"	"7"^^xsd:integer	"6"^^xsd:integer	"23"^^xsd:integer
6	"Chao Chen"	"journal"	"conference"	"5"^^xsd:integer	"9"^^xsd:integer	"21"^^xsd:integer
7	"Yuejie Chi"	"conference"	"journal"	"3"^^xsd:integer	"10"^^xsd:integer	"18"^^xsd:integer
8	"Juan G'omez-Luna"	"journal"	"conference"	"4"^^xsd:integer	"6"^^xsd:integer	"15"^^xsd:integer
9	"R. Schlosser"	"journal"	"conference"	"4"^^xsd:integer	"5"^^xsd:integer	"15"^^xsd:integer
10	"Saugata Ghose"	"journal"	"conference"	"9"^^xsd:integer	"7"^^xsd:integer	"15"^^xsd:integer

Figure 3.2: Top 10 Authors with Highest Cross-Citation Between Different Venues