Knowledge Graph Assignment Report

# 1. TBOX Design Methodology

We designed a TBOX ontology that models the academic publishing domain by defining key concepts such as Paper, Author, Journal, Conference, Edition, Review, and Topic as classes. Each concept is linked via properties like hasAuthor, correspondingAuthor, publishedInJournal, and hasKeyword. Properties were typed with rdfs:domain and rdfs:range. For example, assignedReviewer has domain Review and range Reviewer, allowing inference engines to deduce rdf:type relationships without explicitly declaring them.

# 2. ABOX Generation Methodology

We extracted 100 papers from the Semantic Scholar API using the query 'data science'. To enrich the dataset, we synthetically generated 25 additional papers authored by a simulated conference community spanning five years. The synthetic data includes conference editions and simulated reviewers. All papers, authors, topics, and venues are modeled as RDF resources and linked using properties defined in the TBOX. RDFLib was used to construct the ABOX triples programmatically.

# 3. Inference Regime

The inference regime used is RDFS entailment. Due to domain and range definitions, we omitted explicit rdf:type triples for inferred types such as Reviewer. These types can be inferred automatically during reasoning based on property usage (e.g., assignedReviewer infers the target is a Reviewer).

# 4. Knowledge Graph Summary

|  |  |
| --- | --- |
| Metric | Value |
| Number of Classes | 5 |
| Number of Properties | 8 |
| Number of Triples | See RDF file |

# 5. Querying the Ontology

To demonstrate the value of our ontology and reasoning capabilities in GraphDB, we designed two SPARQL queries. These queries leverage the structure of the TBOX and utilize GraphDB’s inferencing to extract useful insights from the ABOX.

## 5.1 Query 1: Reviewers of Each Paper

This query returns papers and their associated reviewers using the relationships defined in the ontology: ex:hasReview and ex:assignedReviewer. The rdf:type ex:Reviewer is not manually asserted but is inferred by GraphDB thanks to the property range specified in the TBOX. This confirms that RDFS reasoning works properly.

SPARQL Query:

PREFIX ex: <http://example.org/research/>  
SELECT ?paper ?review ?reviewer  
WHERE {  
 ?paper a ex:Paper ;  
 ex:hasReview ?review .  
 ?review ex:assignedReviewer ?reviewer .  
}  
LIMIT 20

## 5.2 Query 2: Most Cited Papers with Topics

This query uses the structure of the knowledge graph to count how many times each paper is cited (using ex:cites), and retrieves associated topics (via ex:hasKeyword). It demonstrates how graph relationships naturally support analytical queries.

SPARQL Query:

PREFIX ex: <http://example.org/research/>  
SELECT ?paper (COUNT(?citation) AS ?citationCount) (GROUP\_CONCAT(DISTINCT ?topic; separator=", ") AS ?topics)  
WHERE {  
 ?paper a ex:Paper .  
 OPTIONAL { ?otherPaper ex:cites ?paper . BIND(?otherPaper AS ?citation) }  
 OPTIONAL { ?paper ex:hasKeyword ?topic }  
}  
GROUP BY ?paper  
ORDER BY DESC(?citationCount)  
LIMIT 10