SPARQL-LD: A SPARQL Extension for Fetching and Querying Linked Data

Pavlos Fafalios and Yannis Tzitzikas
{fafalios,tzitzik}@ics.forth.gr

Institute of Computer Science, Foundation for Research and Technology, GREECE, and Computer Science Department, University of Crete, GREECE

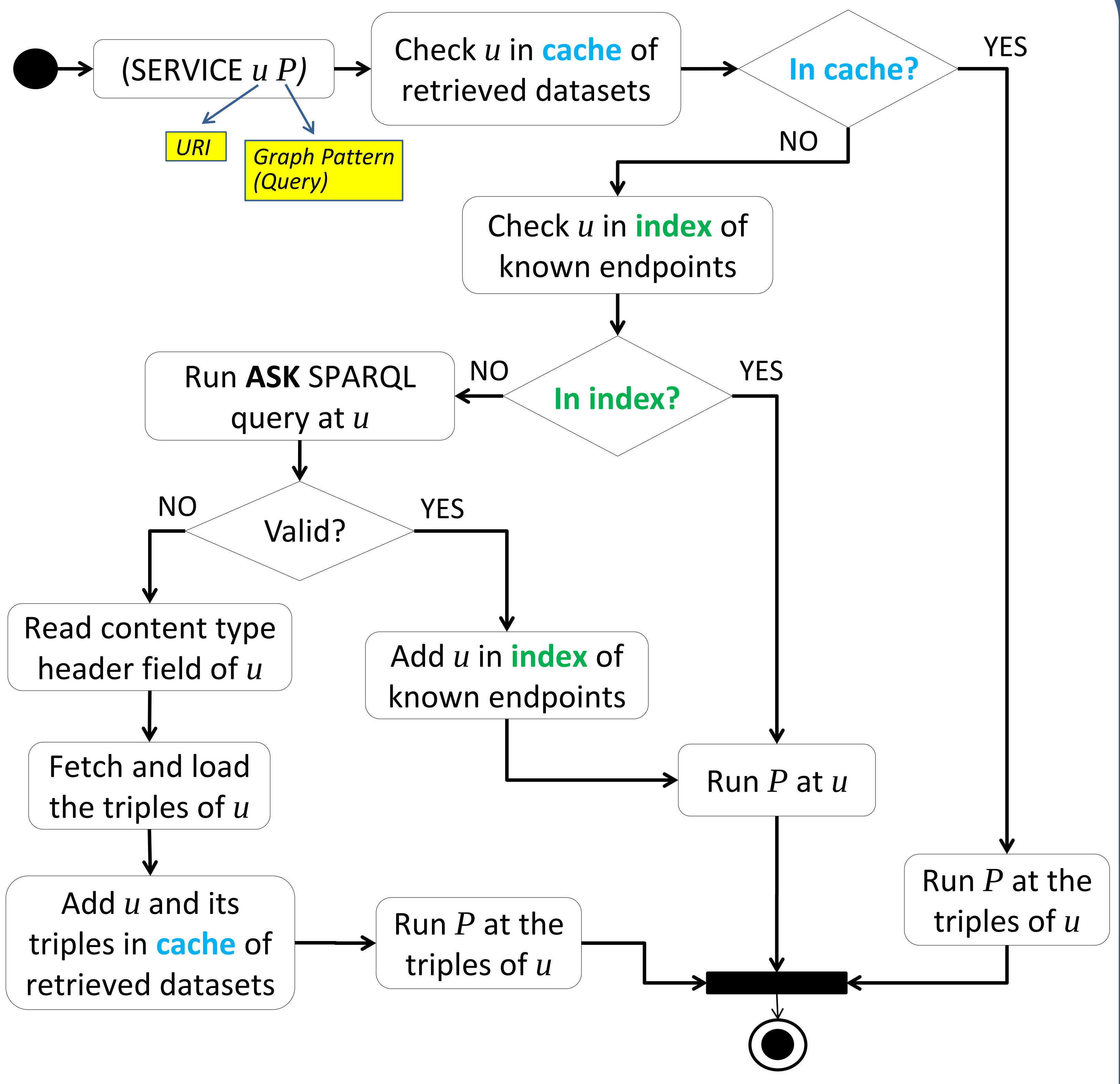
1. WHY?

- ❖ The majority of SPARQL implementations require the data to be available in advance (in main memory or in a repository)
- ❖ The specification of SPARQL allows to directly query an RDF dataset accessible on the Web in a standard format (using FROM/FROM NAMED and GRAPH)
- ❖ However, this has an **important limitation**: it requires knowing **in advance** the URI of the dataset and having declared it in the FROM/FROM NAMED clause
  - ➢ Thus, a URI coming from partial results cannot be used in the GRAPH operator as the dataset to run a portion of the query
- ❖ Furthermore:
  - ➢ Querying **RDFa** and **JSON-LD** is not supported (for the time being) in existing SPARQL implementations although they are both W3C standards
  - ➢ Using the **SERVICE** operator of **SPARQL 1.1 Federated Query** we can invoke a portion of a query against a remote RDF repository
    - ▪ However, SERVICE requires the URI to be the address of a SPARQL endpoint

2. WHAT?

- ❖ SPARQL-LD extends the **applicability** of the SERVICE operator (SERVICE u P) enabling to **fetch** and **query** any Web source containing RDF data
  - ➢ SPARQL-LD can evaluate a graph pattern P not only in a SPARQL endpoint, but generally in an RDF graph specified by a Web resource u
  - ➢ If u is **not** the address of an endpoint, the RDF data that may exist in u are fetched (at real-time) and queried for the graph pattern P
- ❖ SPARQL-LD does **not** require the named graphs to have been declared: one can even query a dataset returned by a portion of the query (derived at query-execution time)
- ❖ SPARQL-LD is a **generalization** of SPARQL: every query that can be answered by the original SPARQL can be also answered by SPARQL-LD
- ❖ Using SPARQL-LD, one can **query and integrate in the same SPARQL query**:
  - ➢ data coming from **dereferenceable URIs**
  - ➢ data embedded in Web pages as **RDFa**
  - ➢ data coming from online **RDF**, **OWL** or **JSON-LD** files
  - ➢ data that is **dynamically** created by **Web Services**
  - ➢ data coming by querying other **endpoints**

3. HOW?


(SERVICE u P)
URI
Graph Pattern (Query)
Check u in cache of retrieved datasets
In cache?
YES
NO
Check u in index of known endpoints
In index?
NO
YES
Run ASK SPARQL query at u
Valid?
NO
YES
Read content type header field of u
Add u in index of known endpoints
Fetch and load the triples of u
Run P at u
Add u and its triples in cache of retrieved datasets
Run P at the triples of u
Run P at the triples of u

- ❖ **Extension** of **Apache Jena** 2.13 **ARQ** component
  - ➢ Available as open source: https://github.com/fafalios/sparql-ld
- ❖ Ability to query RDFa and JSON-LD
- ❖ Optimizations:
  - ➢ ***Index of known endpoints***: avoid submitting an ASK query; the query is directly forwarded to the endpoint
  - ➢ ***Request-scope caching of retrieved dataset(s)***: avoid re-fetching the triples of a Web resource that has been already fetched

4. EXAMPLES

❖ Query **RDFa**, an external **endpoint**, and **dereferenceable URIs** derived at query-execution time:

```
SELECT DISTINCT ?authorURI (count(distinct ?paper) AS ?numOfPapers) (count(distinct ?series) AS ?numOfDiffConfs) WHERE {
    SERVICE <http://users.ics.forth.gr/~fafalios> {  ← Web page containing RDFa
        ?p <http://purl.org/dc/terms/creator> ?authorURI }
    SERVICE <http://dblp.l3s.de/d2r/sparql> {  ← SPARQL endpoint
        ?p2 <http://purl.org/dc/elements/1.1/creator> ?authorURI .
        ?p2 <http://swrc.ontoware.org/ontology#series> ?series }
    SERVICE ?authorURI {  ← Dereferenceable URIs derived at query-execution time
        ?paper <http://purl.org/dc/elements/1.1/creator> ?authorURI }
} GROUP BY ?authorURI ORDER BY DESC(?numOfPapers)
```

The query returns all co-authors of P. Fafalios together with the number of their publications and the number of distinct conferences in which they have a publication.

❖ Parameterize and call a named-entity recognition **Web Service** at query-execution time:

```
SELECT DISTINCT ?detectedEntity ?categoryName (count(?position) as ?NumOfOccurrences) WHERE {
    SERVICE <http://dbpedia.org/resource/Thunnus> {  ← Dereferenceable URI
        dbpedia:Thunnus dbpedia-owl:wikiPageExternalLink ?page }
    VALUES ?templ { <http://139.91.183.72/x-link-marine/api?categories=fish;country&url=PAGE> }
    BIND(REPLACE(str(?templ), "PAGE", str(?page), "i") as ?x) BIND(URI(?x) as ?serv)
    SERVICE ?serv {  ← Named-entity recognition Web Service returning RDF
        ?annot oa:hasBody ?ent .
        ?ent oae:regardsEntityName ?detectedEntity ; oae:position ?position .
        ?ent oae:belongsTo ?category . ?category rdfs:label ?categoryName }
} GROUP BY ?detectedEntity ?categoryName ORDER BY DESC(?NumOfOccurrences)
```

The query first retrieves Web pages related to the fish genus Thunnus (using its dereferenceable IRI), then it calls a named-entity recognition service (X-Link) for identifying (at request time) names of fishes and countries in these Web pages, and for each detected entity the query retrieves (and shows) its name, its category and its number of occurrences in the Web pages (the entities are ordered by the number of occurrences in descending order).

FORTH-ICS-ISL   UOC-CSD

TRY SPARQL-LD:
http://users.ics.forth.gr/~fafalios/sparql-ld-endpoint