The KGRAM Abstract Machine for Knowledge Data Management

Olivier Corby, Catherine Faron Zucker & Alban Gaignard

- Graph based Knowledge Representation
  - Semantic Web, Linked Open Data
  - Conceptual Graph
  - Semantic Network

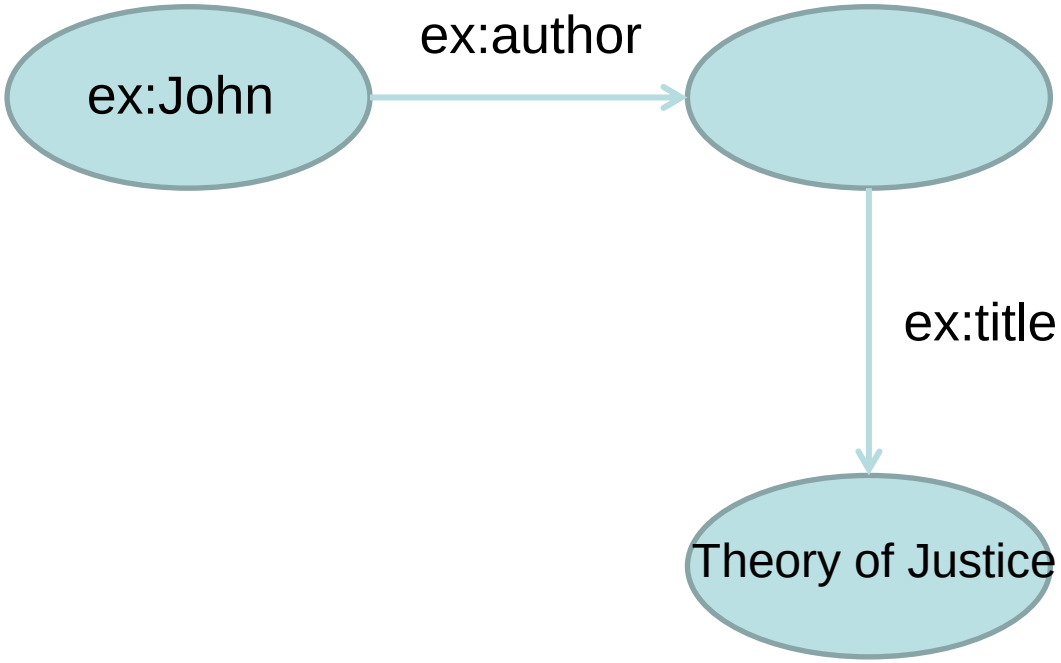
- Graph Query
  - SPARQL Query Language
  - Graph Homomorphism
  - Projection
  - Graph matching

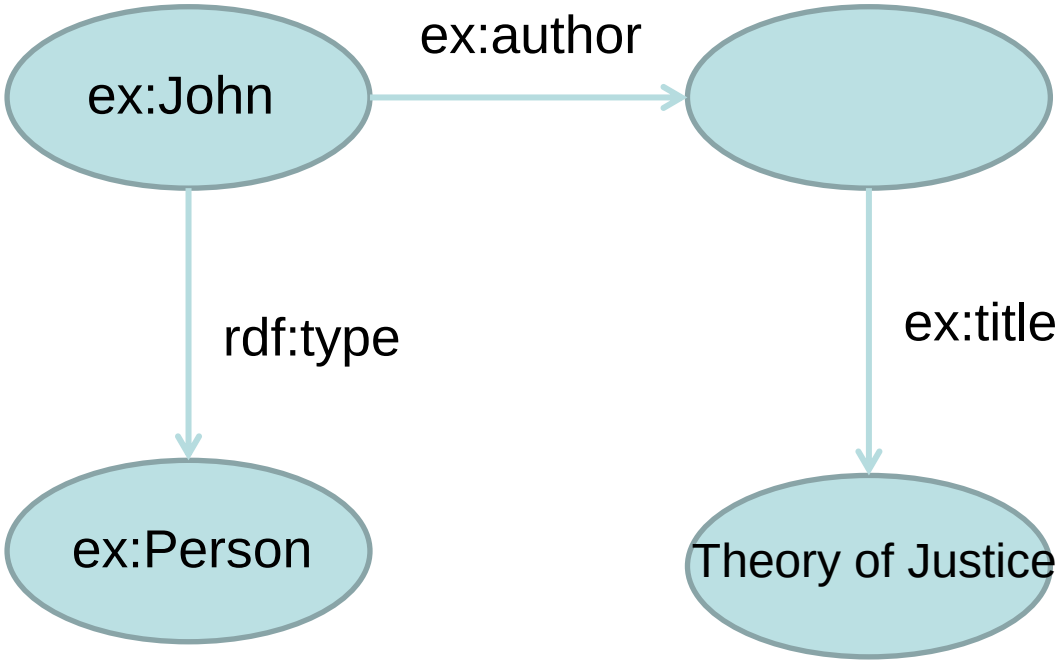
KGRAM:

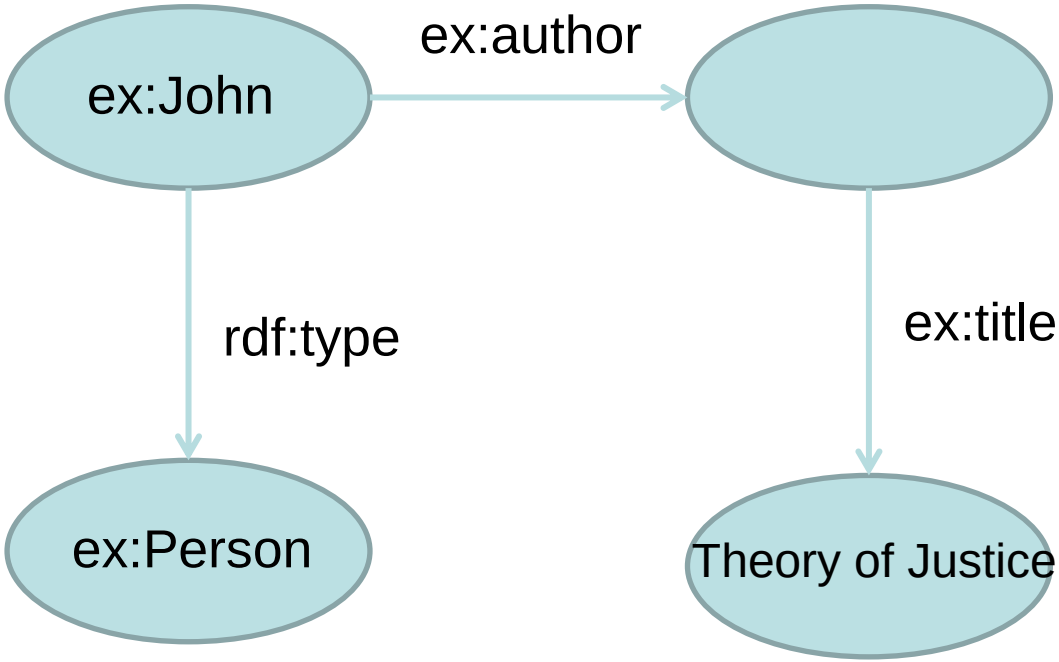
1. Unify and generalize graph structures (RDF, CG)
2. Unify graph matching (homomorphism, projection, SPARQL)
3. Abstract machine for graph matching as a generic and modular software

- KGRAM Principles
  - Generic Extended SPARQL 1.1 Interpreter
- KGRAM Abstract Machine
  - Components
  - Architecture
  - Portability & interoperability
- Conclusion & Ongoing Work

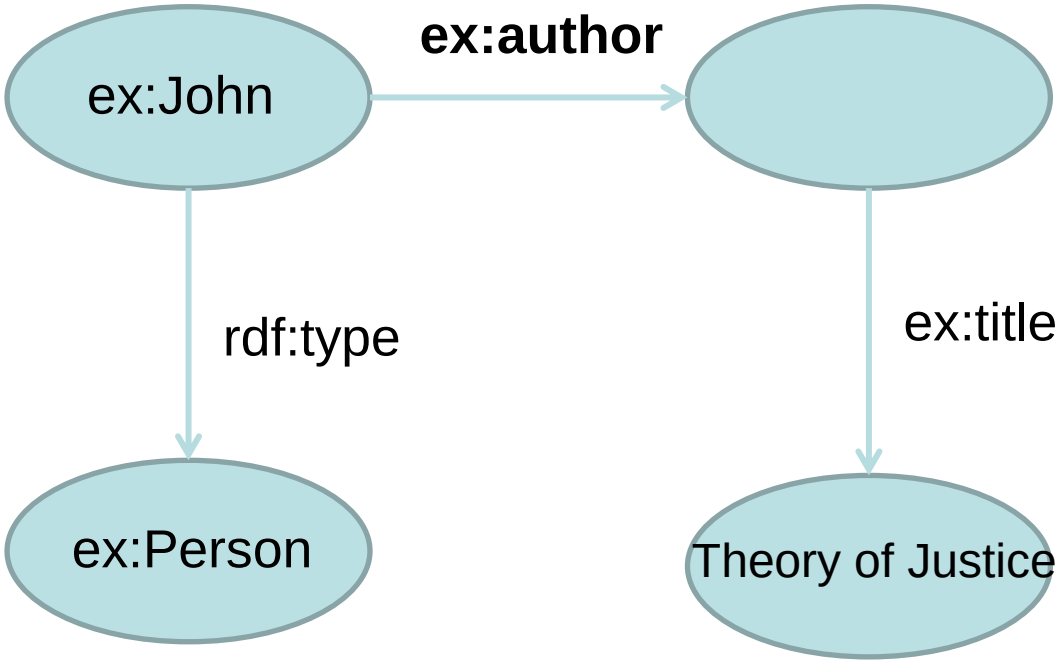
- RDF: Resource Description Framework
- RDFS: RDF Schema (light weight ontology)
- SPARQL 1.1
  - Query
  - Update
- RIF: Rule Interchange Format


ex:John
ex:author
ex:title
Theory of Justice

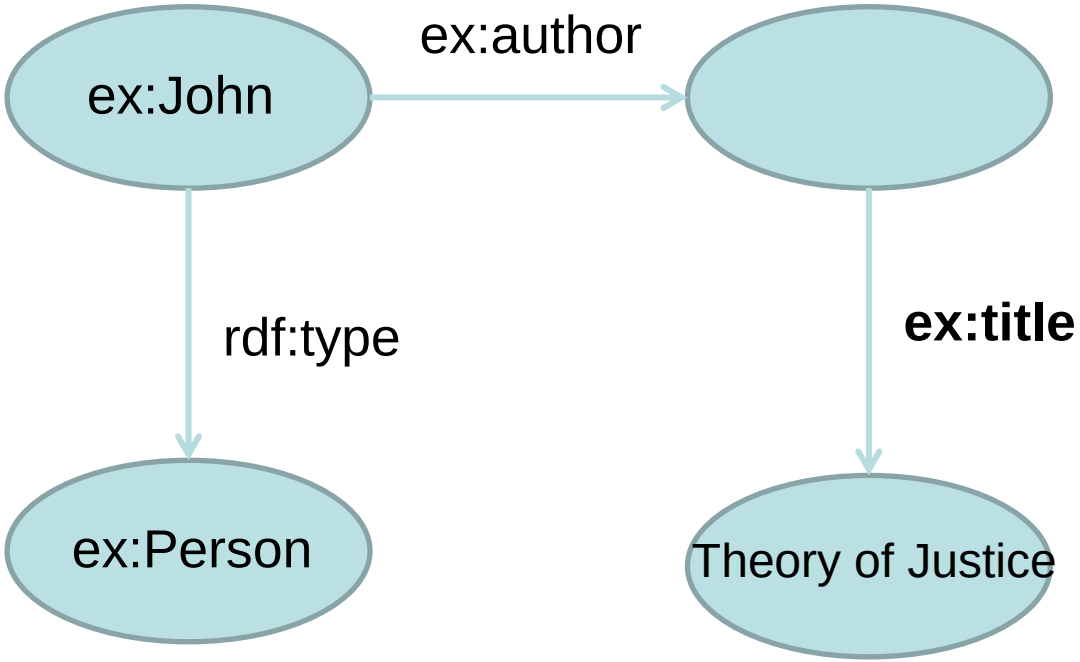

ex:John
ex:author
rdf:type
ex:title
ex:Person
Theory of Justice



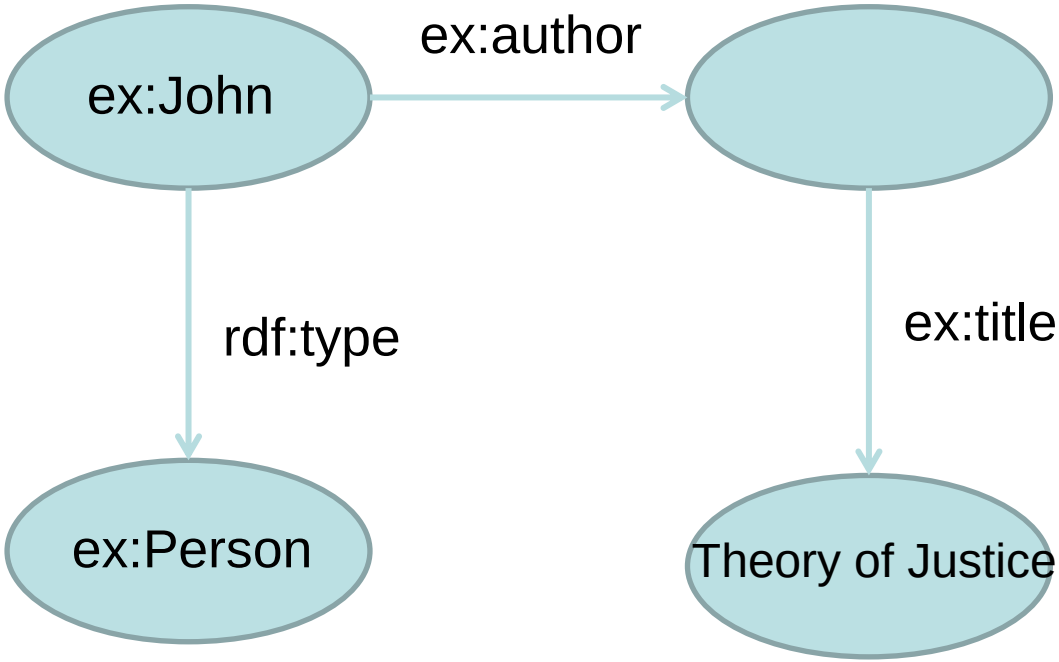

```
prefix ex: <http://www.example.org/>
select ?x ?t where {
  ?x ex:author ?doc .
  ?doc ex:title ?t
}
```




```
prefix ex: <http://www.example.org/>
select ?x ?t where {
  ?x ex:author ?doc .
  ?doc ex:title ?t
}
```

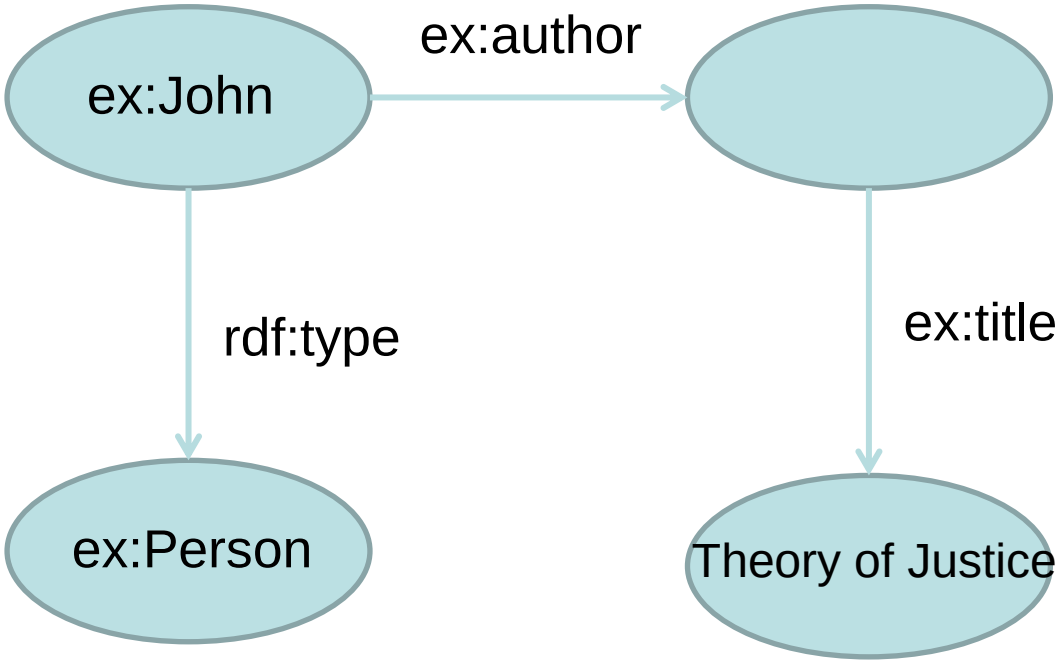



```
prefix ex: <http://www.example.org/>
select ?x ?t where {
  ?x ex:author ?doc .
  ?doc ex:title ?t
}
```

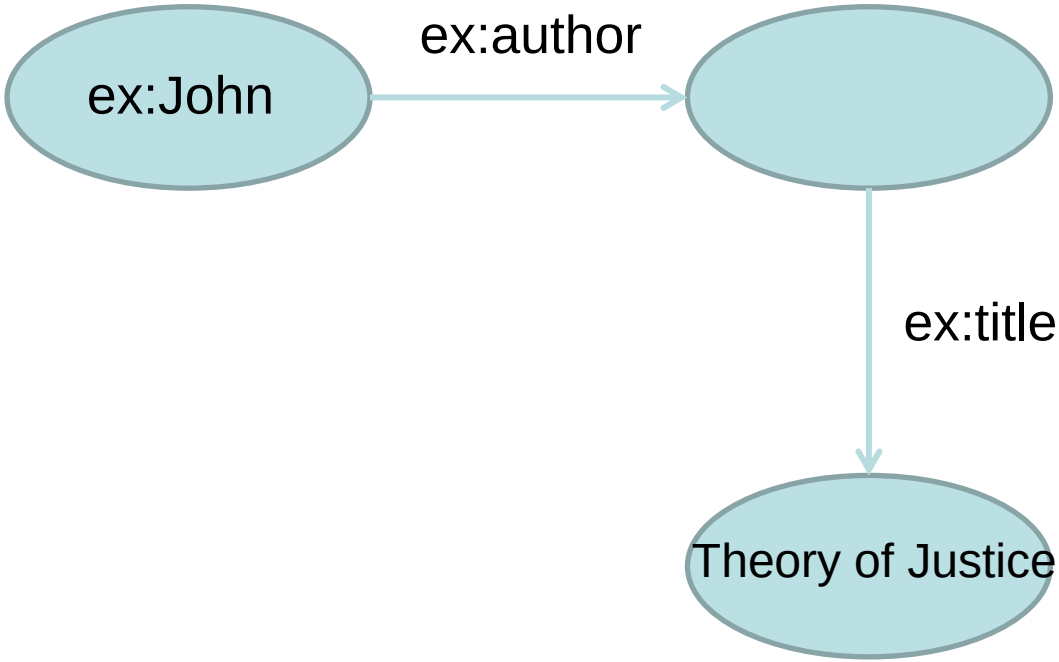



```
prefix ex: <http://www.example.org/>
select ?x ?t where {
  ?x ex:author ?doc .
  ?doc ex:title ?t
}
```

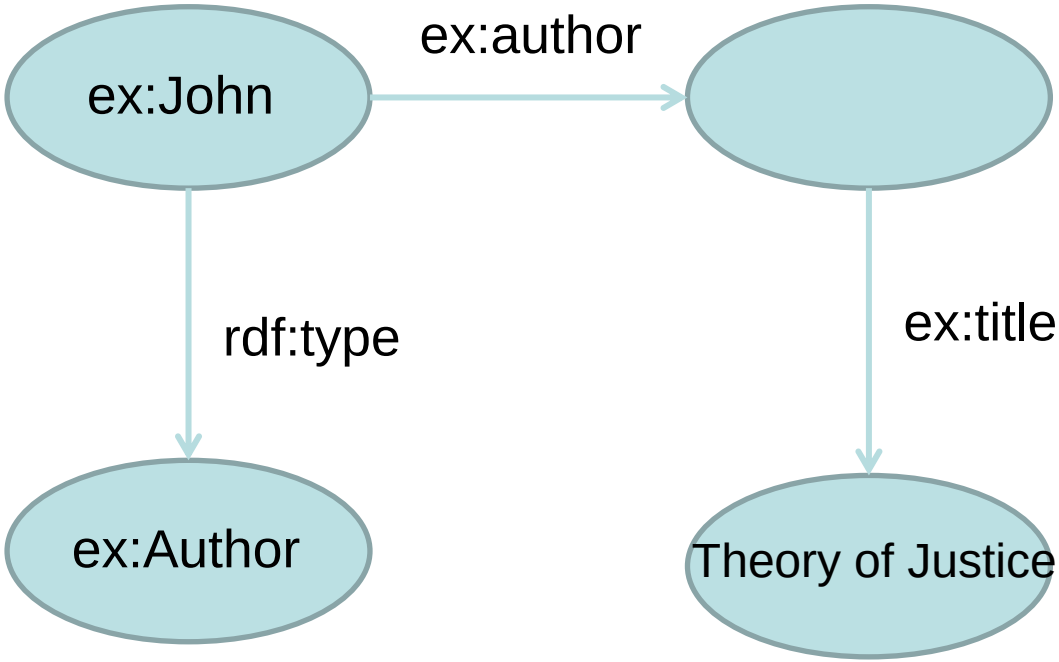
?x = ex:John
?t = « Theory of Justice »




```
prefix ex: <http://www.example.org/>
delete {?x rdf:type ex:Person}
insert  {?x rdf:type ex:Author}
where {?x ex:author ?doc}
```




```
prefix ex: <http://www.example.org/>
delete {?x rdf:type ex:Person}
insert  {?x rdf:type ex:Author}
where {?x ex:author ?doc}
```




```
prefix ex: <http://www.example.org/>
delete {?x rdf:type ex:Person}
insert  {?x rdf:type ex:Author}
where {?x ex:author ?doc}
```

- Matching algorithms
  - SPARQL
  - Simple Graph Homomorphism
  - Several entailment regimes
  - Homomorphism with constraints
  - Node matching

- Data model
  - Not only RDF but labeled graphs
  - Graphs with n-ary edges

1. Database (SQL)
2. XML (XPath)
3. Approximate search wrt types
4. Extended Property Path
5. Rewriting PP into BGP
6. RDF Graph as Query Graph
7. RDF AST SPARQL-based Pretty Printer
8. Rule Engine
9. Pragma (debug mode, PP rewrite, service timeout, …)
10. Event Listener

Database (SQL)

```
insert {?s ?p ?o}
where {
   select (sql(db:name, ‘SELECT FROM WHERE’ ) as (?s ?p ?o))
   where {}
}
```

RDF Graph as Query Graph

Graph g = eval(‘construct {} where {}’)

Mappings map = eval(g)

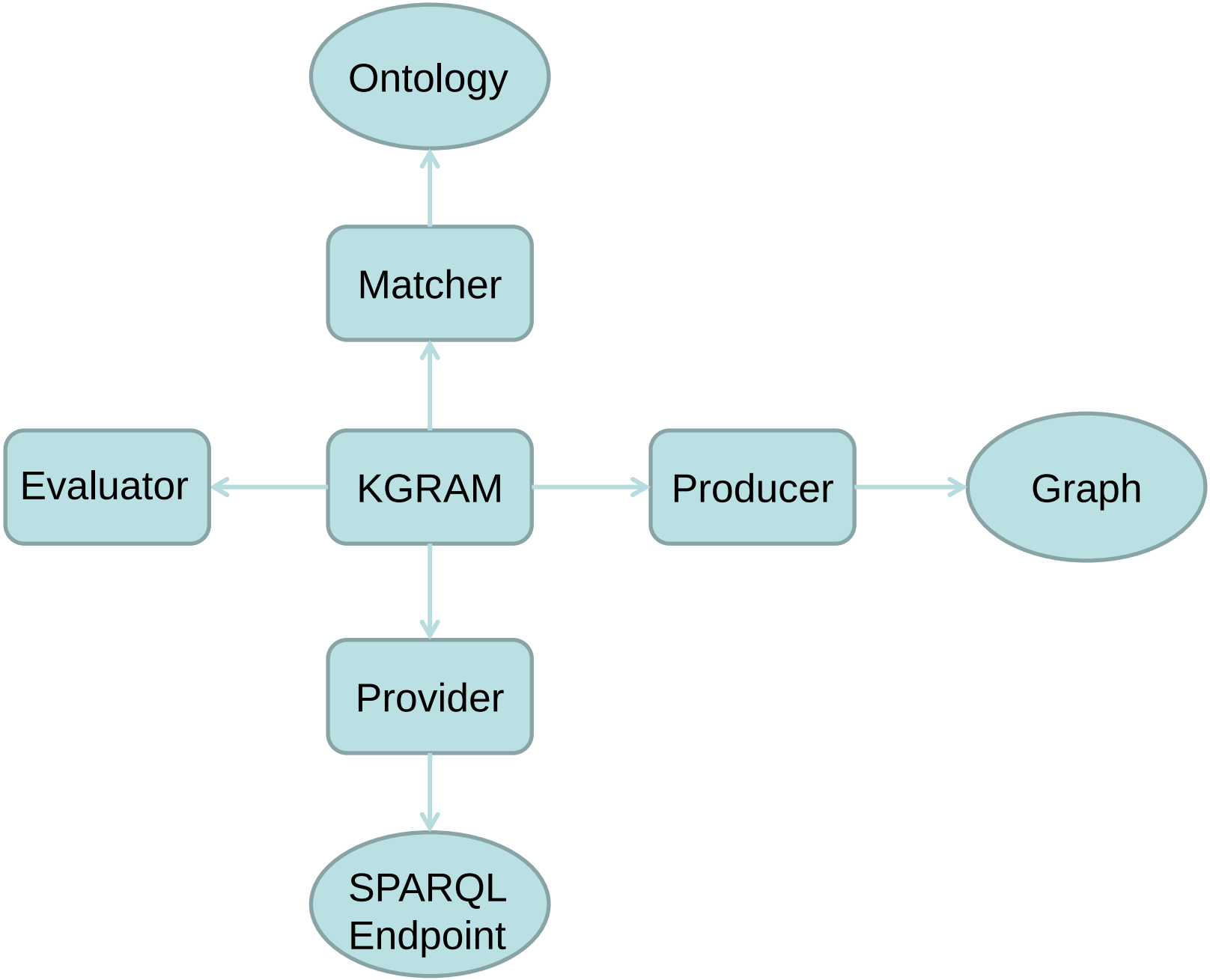
- Interpret a query expression independently of underlying graph implementation
- Operate on abstract query and target graph through APIs and Proxys

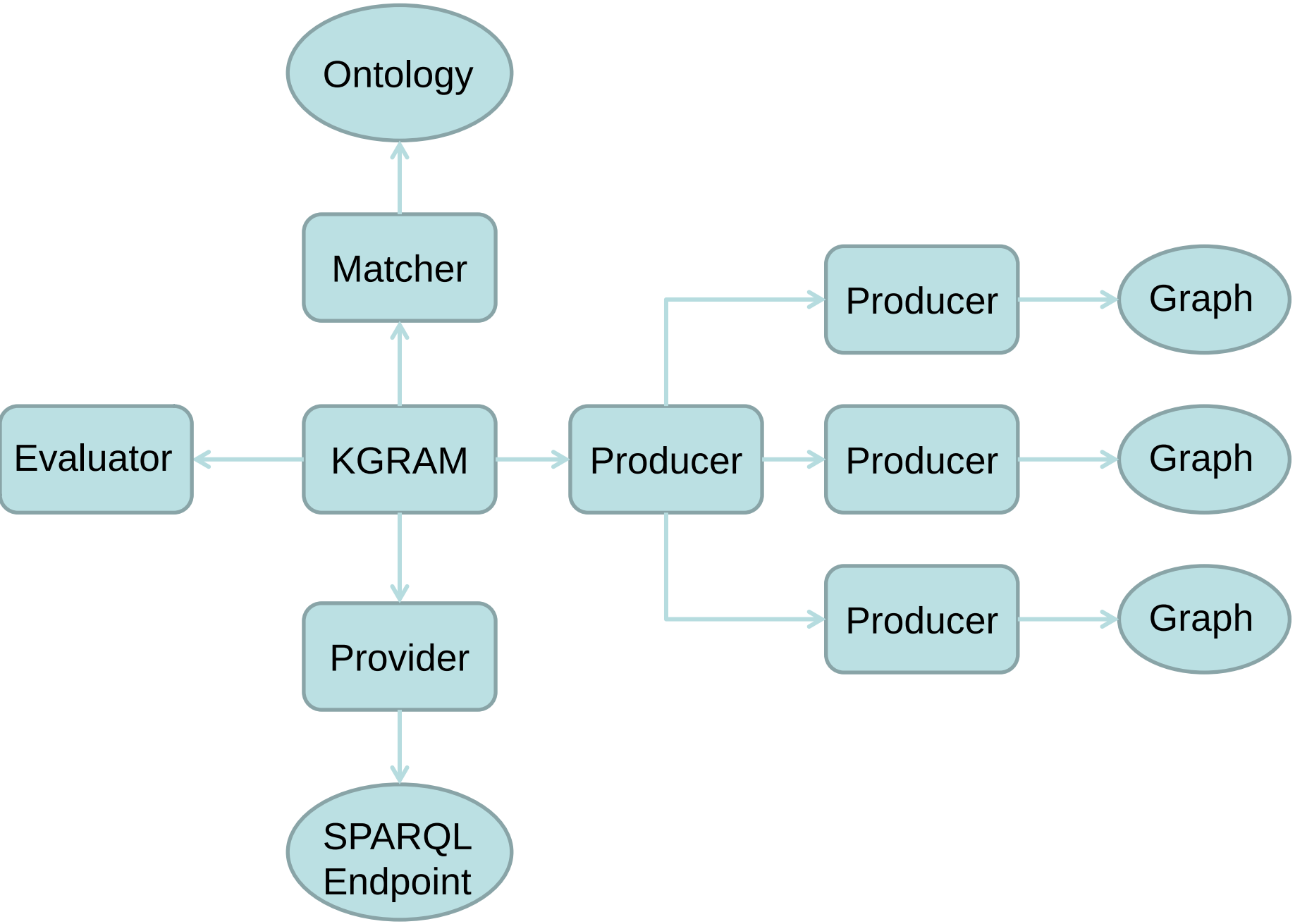
- Query & Target Graph API:
  - Edge,
  - Node

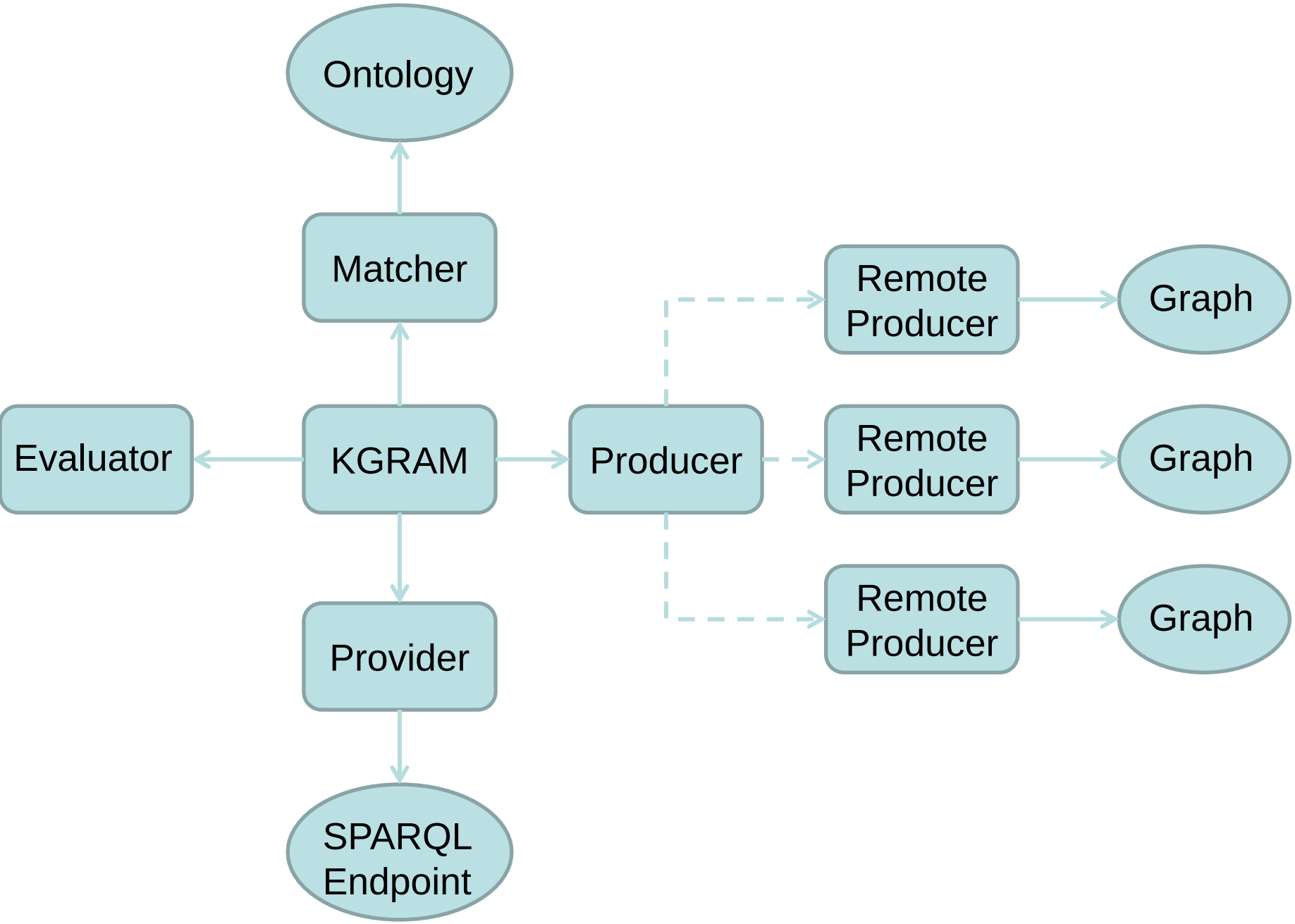
- Portability & Interoperability
  - Corese: CG & RDF/S
  - Jena: RDF/S

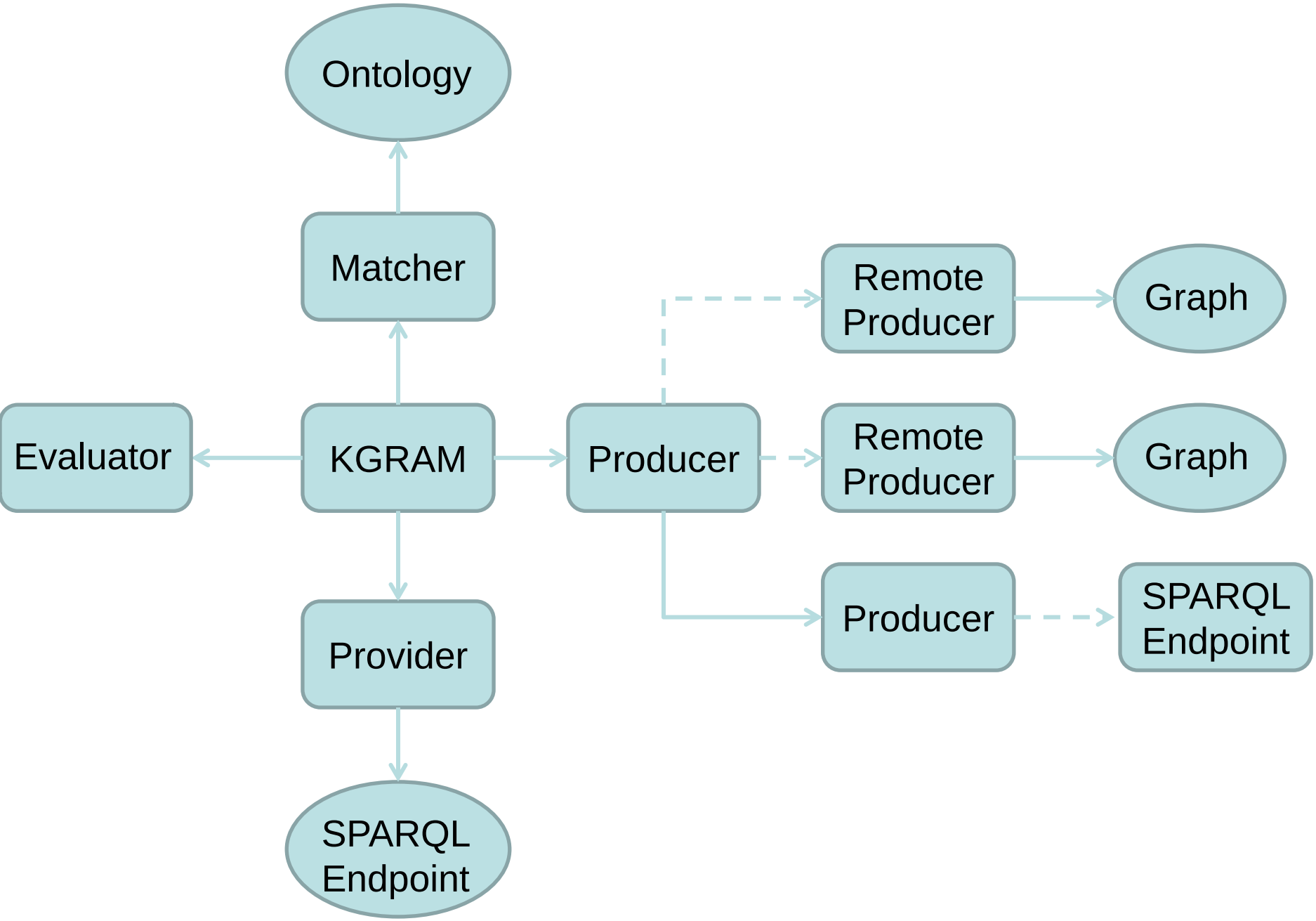
Proxy

1. Producer: enumerate candidate edges
2. Provider: service statement
3. Matcher: check edges wrt ontology
4. Evaluator: filter statement


Ontology
Matcher
Evaluator
KGRAM
Producer
Graph
Provider
SPARQL
Endpoint


Ontology
Matcher
Evaluator
KGRAM
Provider
SPARQL Endpoint
Producer
Producer
Producer
Graph
Graph
Graph


Ontology
Matcher
Evaluator
KGRAM
Producer
Remote Producer
Graph
Remote Producer
Graph
Remote Producer
Graph
Provider
SPARQL Endpoint


Ontology
Matcher
Evaluator
KGRAM
Provider
SPARQL Endpoint
Producer
Remote Producer
Graph
Remote Producer
Graph
Producer
SPARQL Endpoint

A Producer can access :

– One or several local graphs
– One or several remote graphs
– SPARQL endpoints
– A database
– XML documents

- Remote Producers
  - SOAP Web service
  - HTTP

SPARQL Endpoint

```
select * where {
   service <http://fr.dbpedia.org/sparql> {
     ?x rdfs:label ?l
   }
   ?x rdfs:seeAlso ?y
}
```

SPARQL Endpoint

```
select * where {
   service <http://fr.dbpedia.org/sparql> {
     ?x rdfs:label ?l
   }
   ?x rdfs:seeAlso ?y
}
```

SPARQL Endpoint

SPARQL Endpoint

```
select * where {
   service <http://fr.dbpedia.org/sparql> {
     ?x rdfs:label ?l
   }
   ?x rdfs:seeAlso ?y
}
```

SPARQL Endpoint

Local Graph

Rule Engine

Construct where inference rules:

```
construct {?x ex:hasUncle ?z}
where     {?x ex:hasFather ?y . ?y ex:hasBrother ?z}
```

```
ex:John ex:hasFather ex:Jack
ex:Jack ex:hasBrother ex:James
->
ex:John ex:hasUncle ex:James
```

Where clause interpreted as SPARQL query by KGRAM

Distributed reasoning is possible

– Graph Matching Abstract Machine
– SPARQL 1.1 not only for RDF
– Distributed query and reasoning

– Mashup of heterogeneous data
– Distributed reasoning
– Explanation
– Graph Programming for the Semantic Web

– Web site: http://wimmics.inria.fr/corese

Thanx for listening!