Design and Implementation of an Efficient Semantic Web Reasoner

Samir Amir and Hassan Aït-Kaci

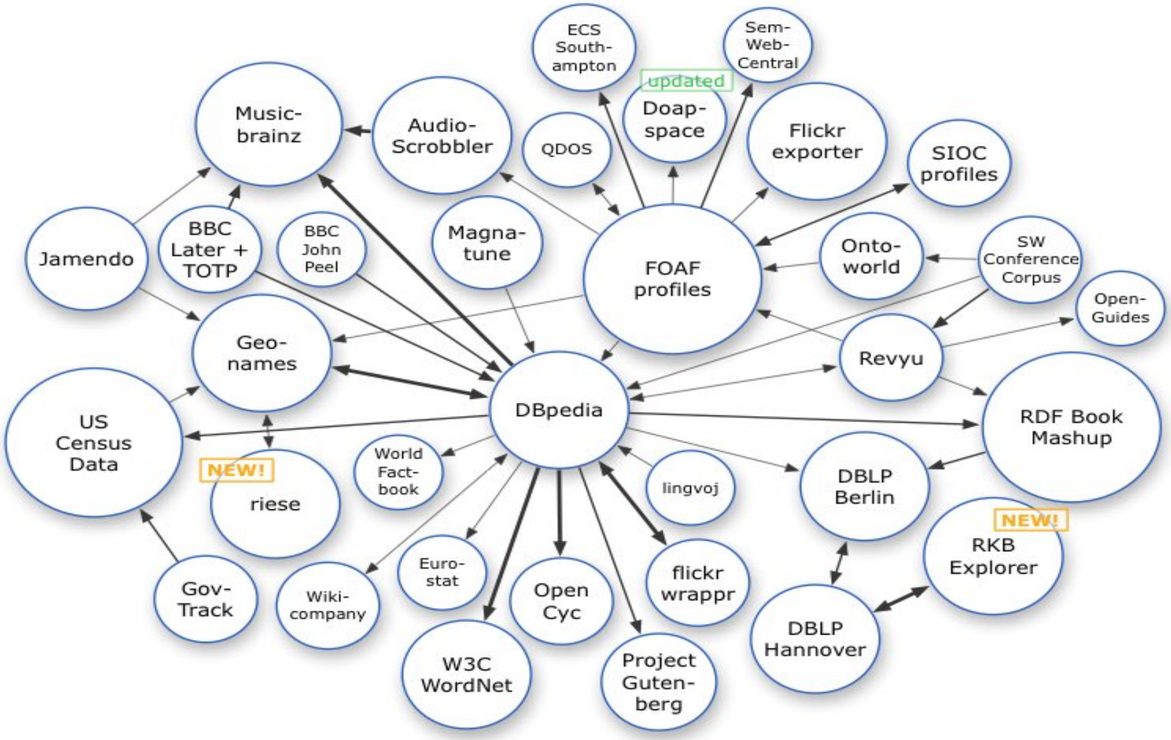
samir.amir@univ-lyon1.fr
hassan.ait-kaci@univ-lyon1.fr

Plan

- Introduction
- Semantic Web reasoning
- Experiments
- CEDAR reasoner building blocks
- Conclusion

how to make the Web Intelligent ?

- **Linked Data** has become *de facto* standard for distributed information—***it does to* RDF *what HTML has done to text: it interconnects knowledge through the Internet***



Challenges are Scalability* and *Distribution

Semantic Web reasoning

(1) Description Logic based reasoners

- Tableau-based reasoning.
- Rules.

HermiT [Rob Shearer2008]
Pellet [Evren Sirin 2007]
FaCT++[Tsarkov2006]
RacerPro[Volker2005]
- ....

(2) Logic programming

- Prolog

- SWORIER [Grosof2003]
- (OLP) Ontological Logic Programming [Sensoy2009]

Large-reasoning is a challenge [Srinivas. 2009] [Dentler2011]

OSF for the Semantic Web

- *OSF* (*Order-sorted feature*)[Aït-Kaci97]

  ➔ *Expressivity*

  ➔ *Scalability*

  ➔ *Graph unification-based reasoning.*

  ➔ *OSF constraints have a graph structure*

  ➔ *RDF compatibility*

- ***CEDAR reasoner is built on the top of OSF***

Syntax

- person (teachesAt => institution, doesResearch => laboratory).

Experiments

CEDAR *vs* OWL reasoners (TBox reasoning)

Benchmarks:

Ontology	sorts	properties
Amphibian [Maglia et al. 2007]	6135	30 (generated)
Molecule Role [Yamamoto et al. 2004]	9127	7
FMA [Rosse and Mejino 2003]	83 283	77
CPO [Hoehndorf et al. 2012]	136 006	55
MESH [of Medicine 2003]	286 380	32
NCBI [Federhen 2012]	903 617	30 (generated)

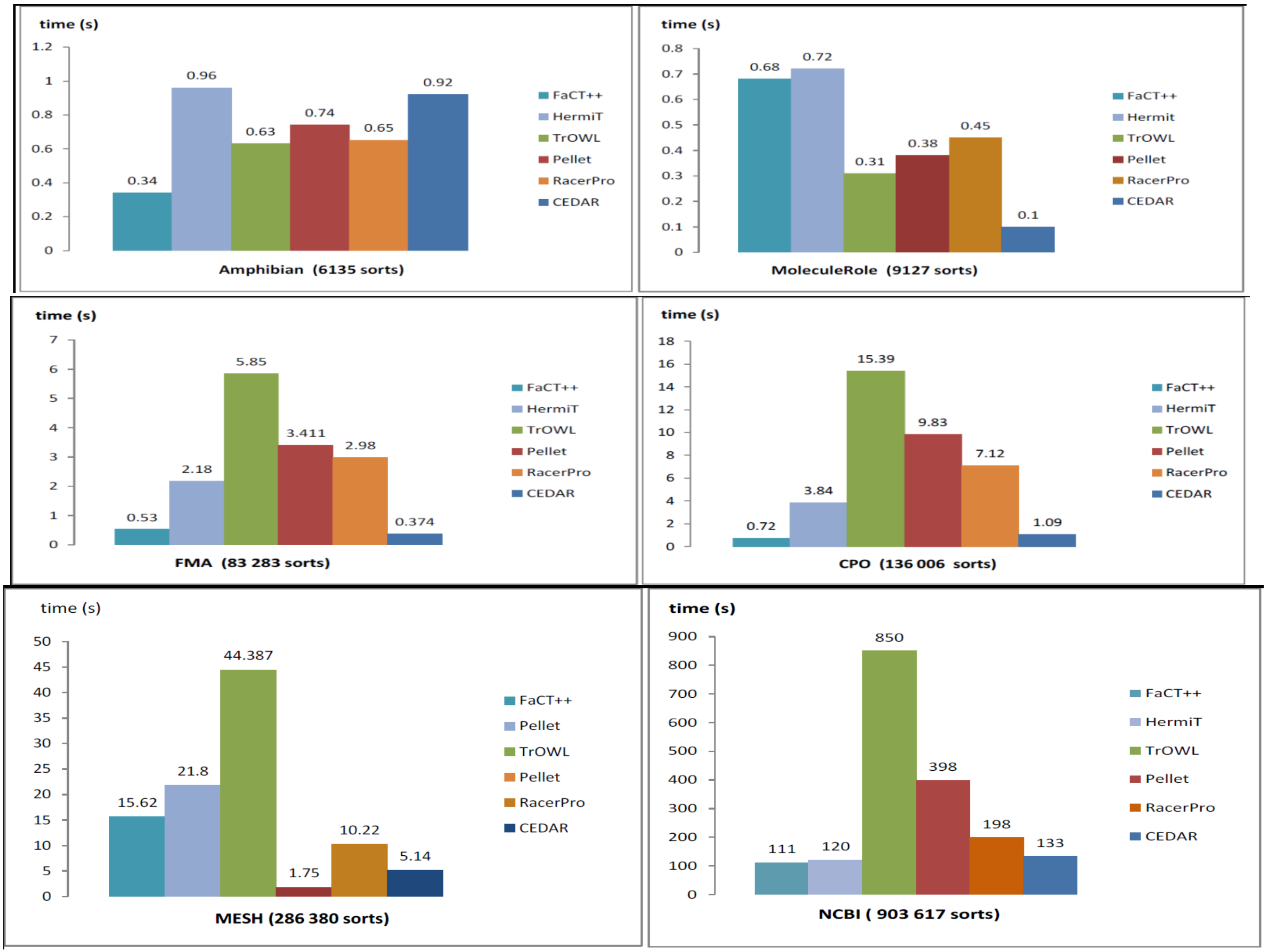
Reasoners:

Pellet, HermiT, RacerPro, TrOWL, FaCT++ and CEDAR

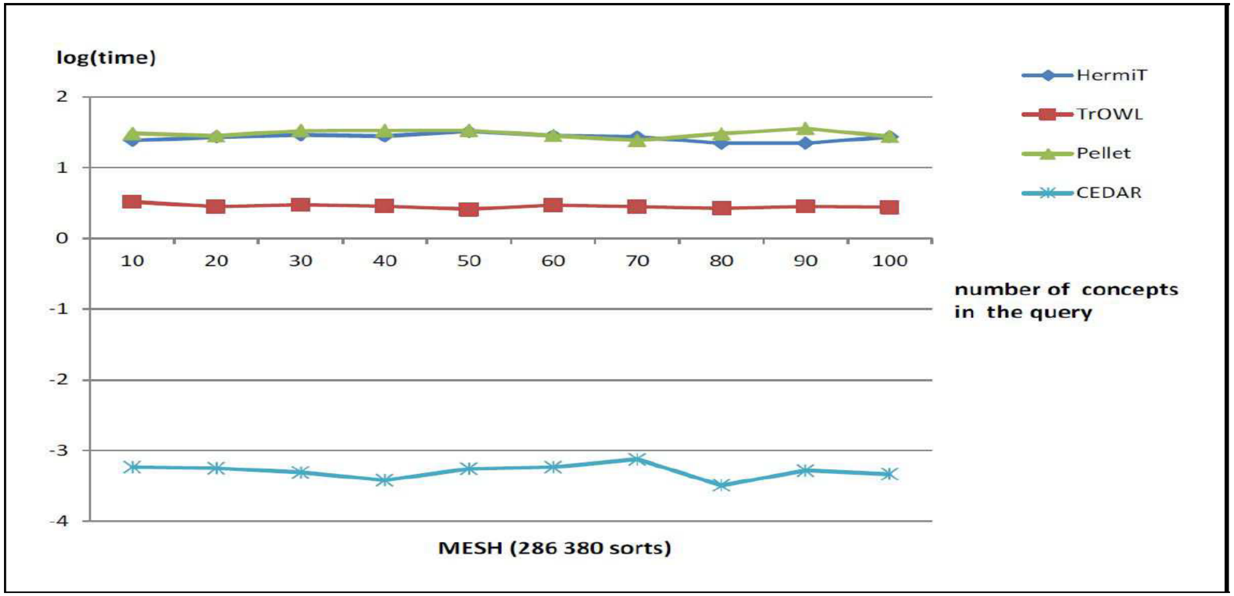
Queries:

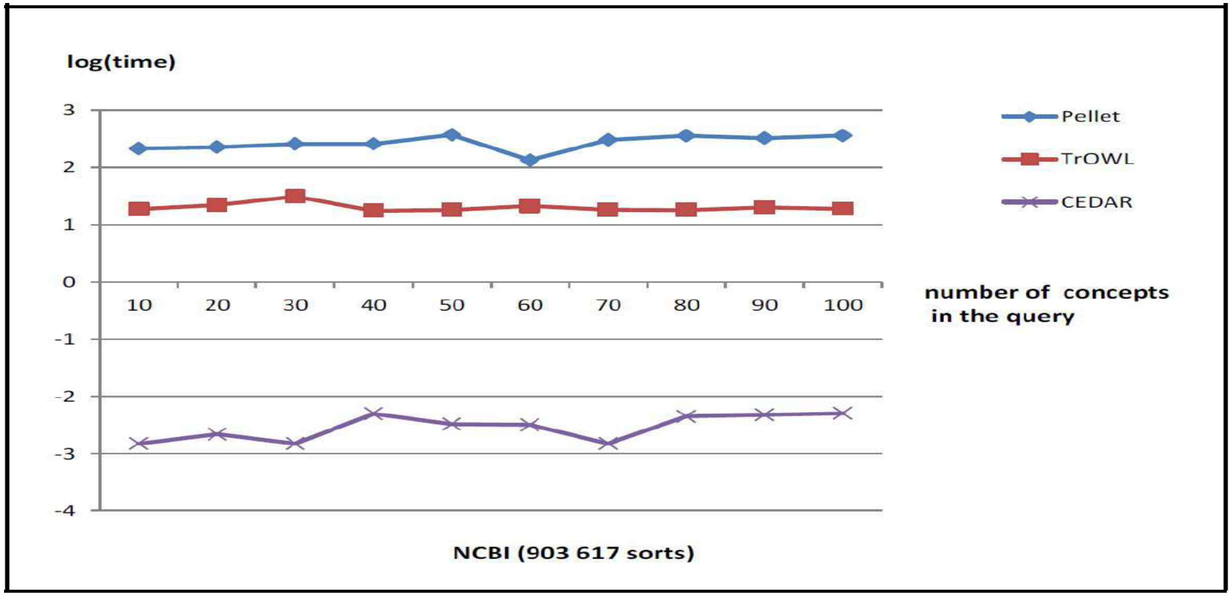
$$\mathbf{X} : \mathbf{s} (\mathbf{f}_1 \Rightarrow \mathbf{s}_1, \ldots, \mathbf{f}_n \Rightarrow \mathbf{s}_n) \text{ for } 10 \leq n \leq 100.$$

Classification



CEDAR *vs* OWL reasoners (TBox reasoning)





HermiT, FaCT++, RacerPro

➔

more than 20 minutes !

CEDAR *vs* OWL reasoners (ABox reasoning)

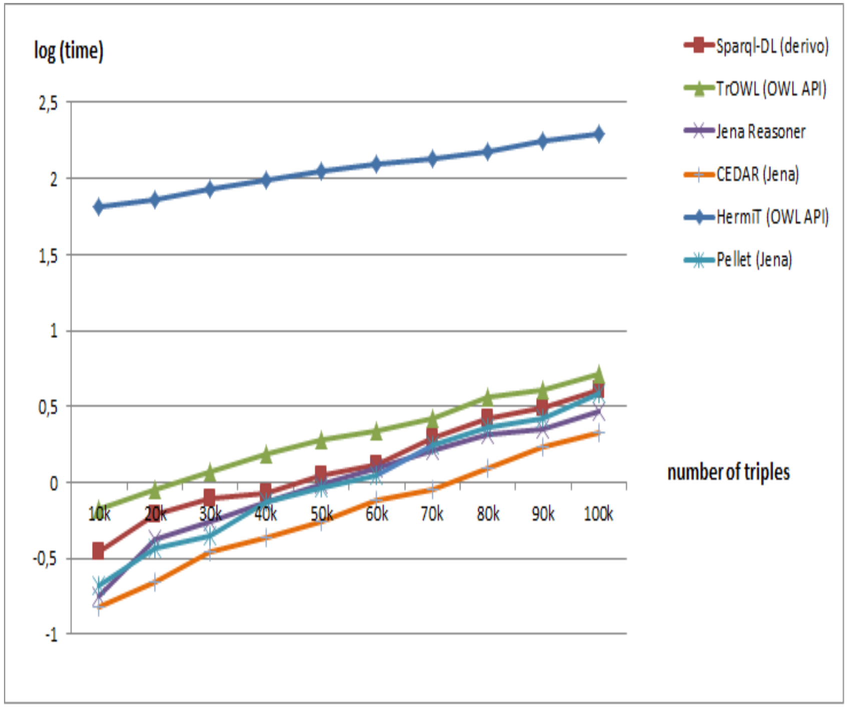
- LUBM Generator (10 K - 5M triples)
- Pellet, HermiT, RacerPro, TrOWL, FaCT++, Jena, SPARQL-DL, CEDAR.

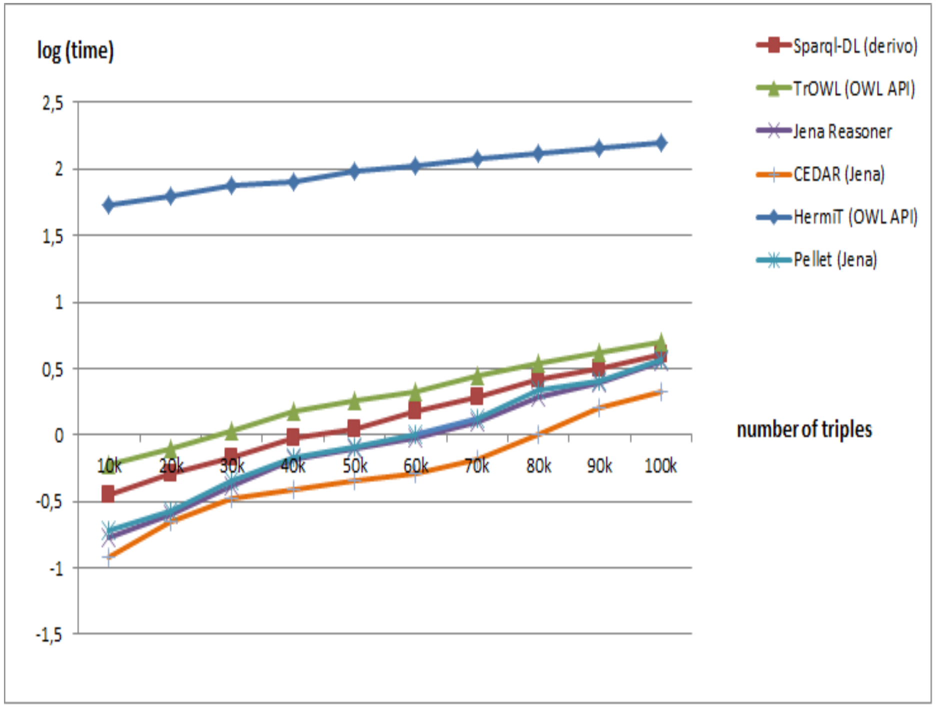
CEDAR *vs* OWL reasoners (ABox reasoning)

LUBM Generator

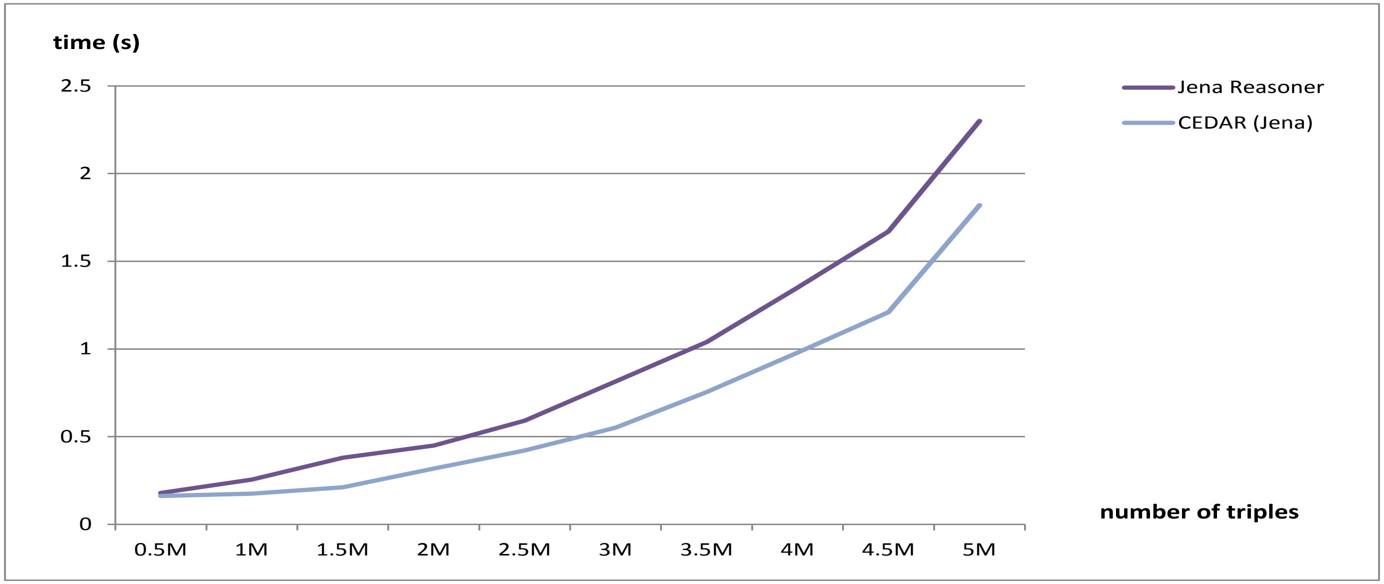
Q1: person (takesCourse =>course)

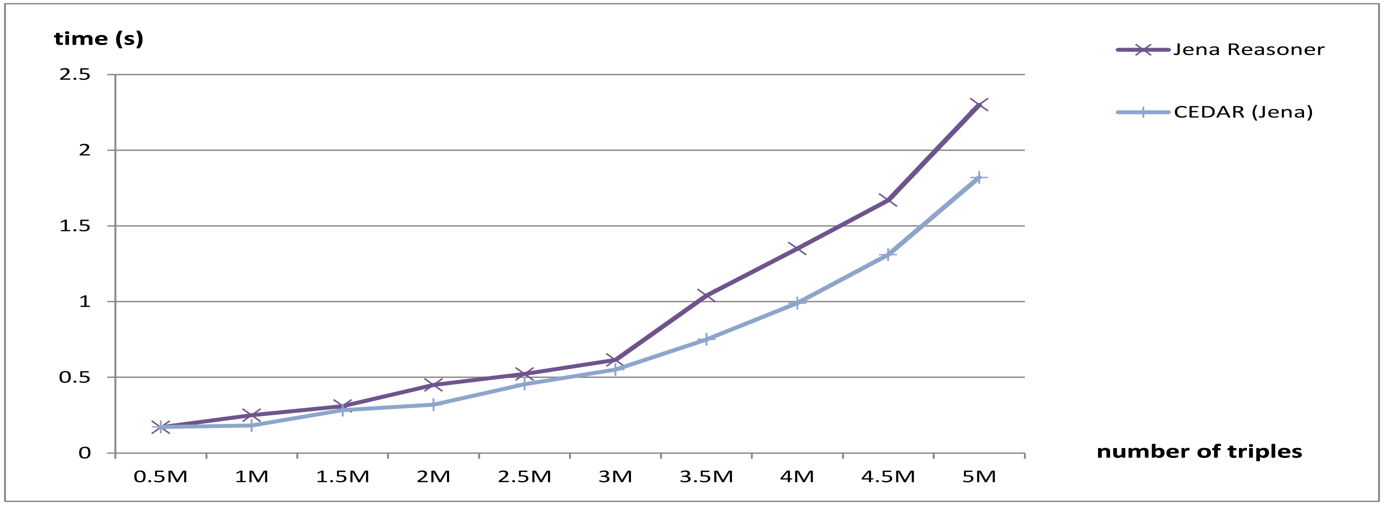
Q2: person (worksFor => organization, headOf => department)



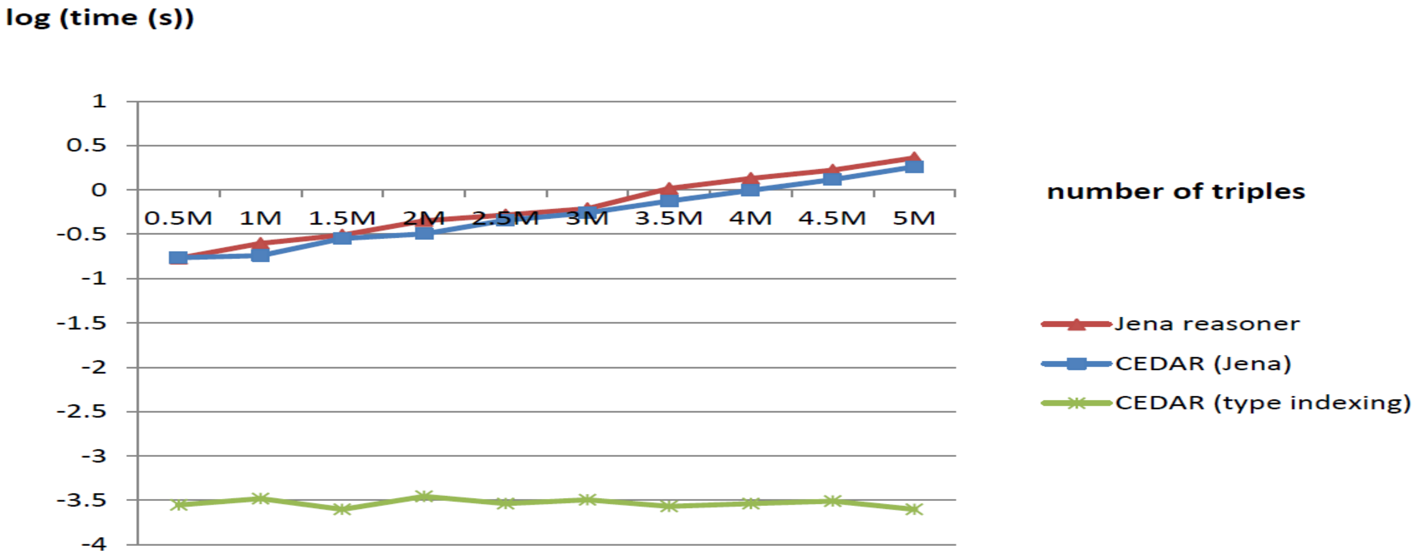


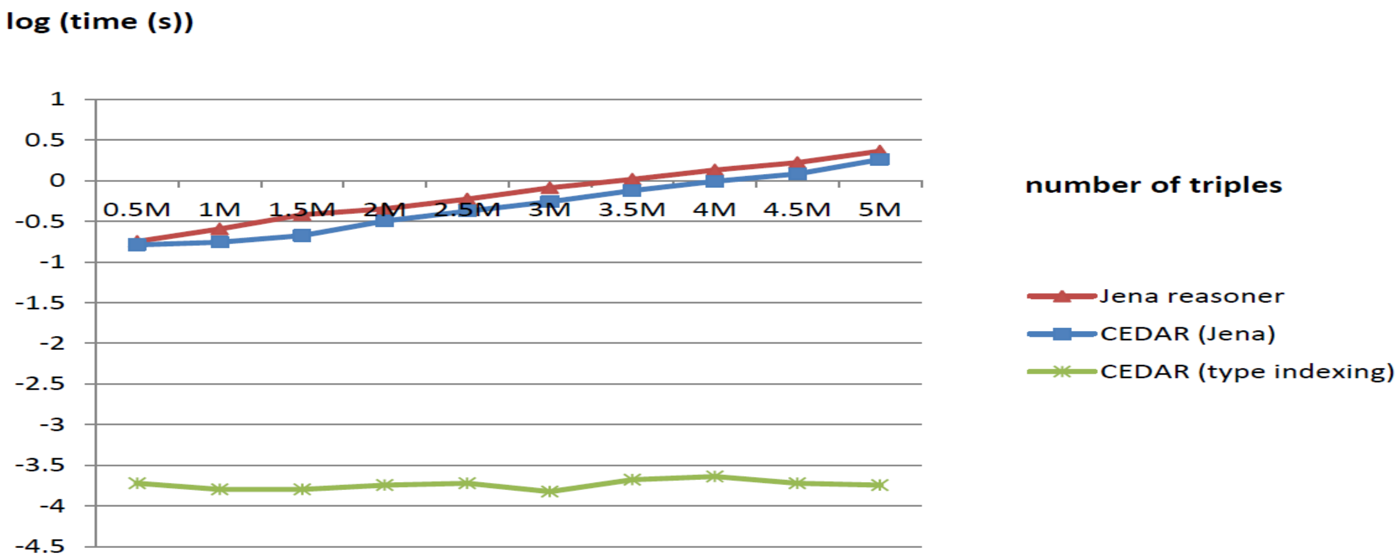
CEDAR *vs* OWL reasoners (ABox reasoning)



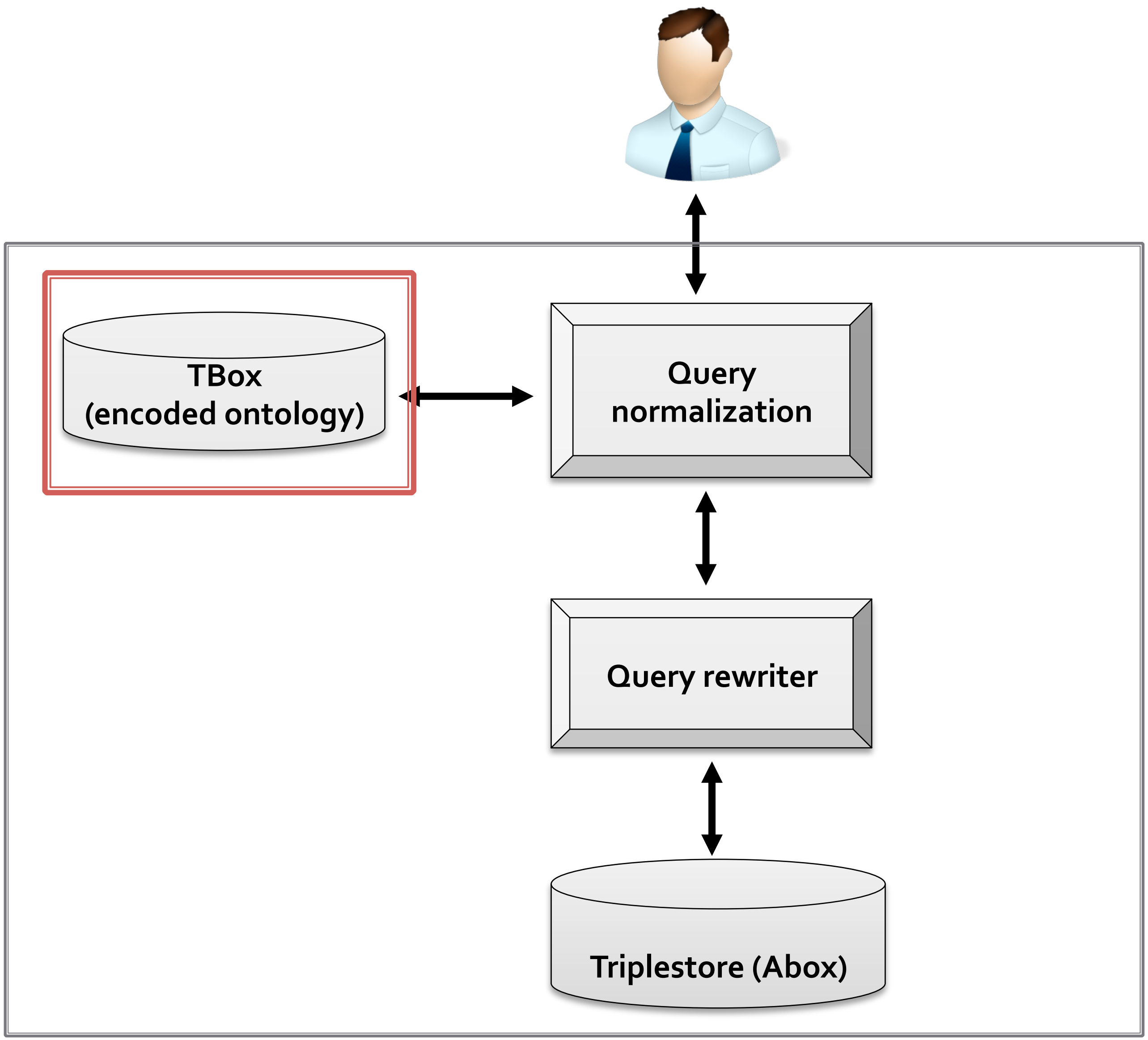


CEDAR *vs* OWL reasoners (ABox reasoning)





CEDAR architecture



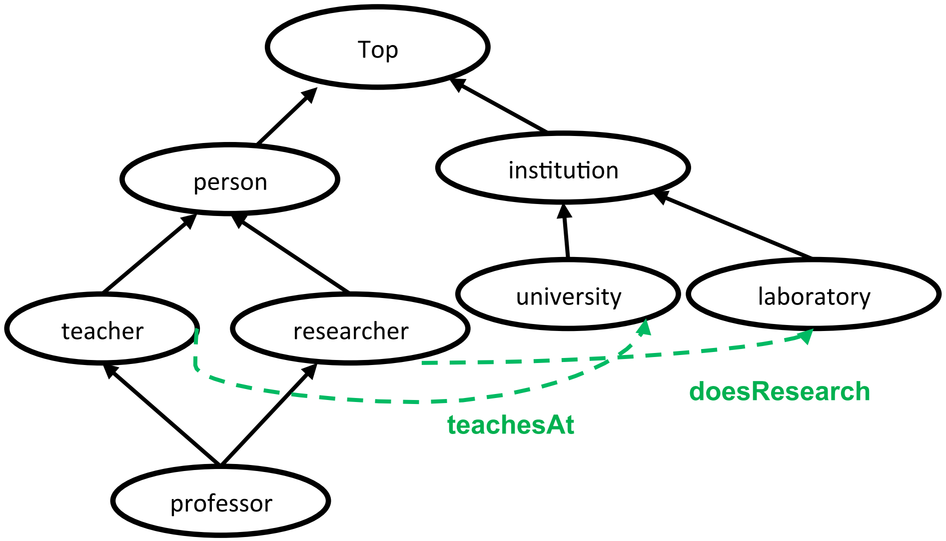
CEDAR Building Blocks

Classification

Ontology encoding (classification)

- ❑ Binary Encoding (transitive closure)
- ❑ Feature Propagation.
- ❑ Consistency Checking.
- ❑ Cycle detection.

Binary encoding [Aït-Kaci 89]



Top	**11111111**	8
person	**01111000**	7
teacher	**00101000**	6
researcher	**00011000**	5
professor	**00001000**	4
institution	**00000111**	3
university	**00000010**	2
laboratory	**00000001**	1

professor **is-a** teacher, teacher **is-a** person ➔ professor **is-a** person

person ➔ **01111000**

teacher **and** researcher ➔ professor (00101000 and 00011000 ➔ 00001000)

teacher **teachesAt** university ➔ professor **teachesAt** university

Classification

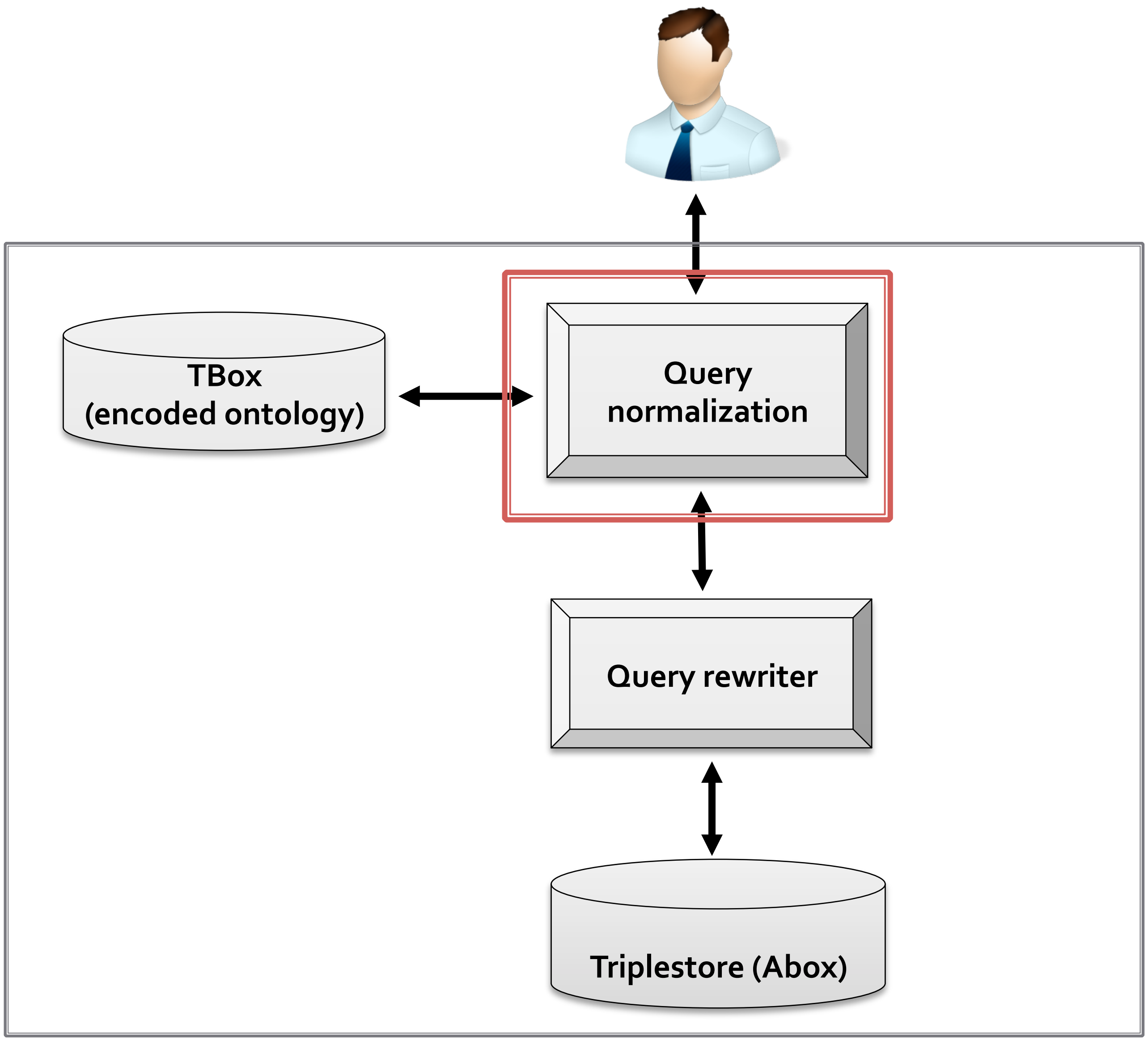
Ontology encoding (classification)

- ❑ Binary Encoding.
- ❑ Feature Propagation.
- ❑ Consistency Checking.

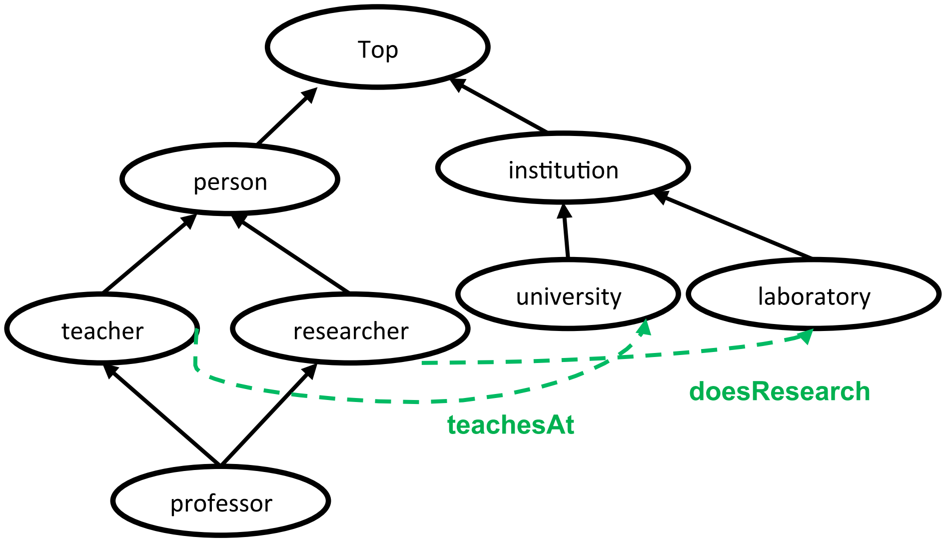
- ❑ **The encoded ontology is saved on the disk**

  ➔ **There is no need to recompute the classification.**

Query normalization



Query normalization



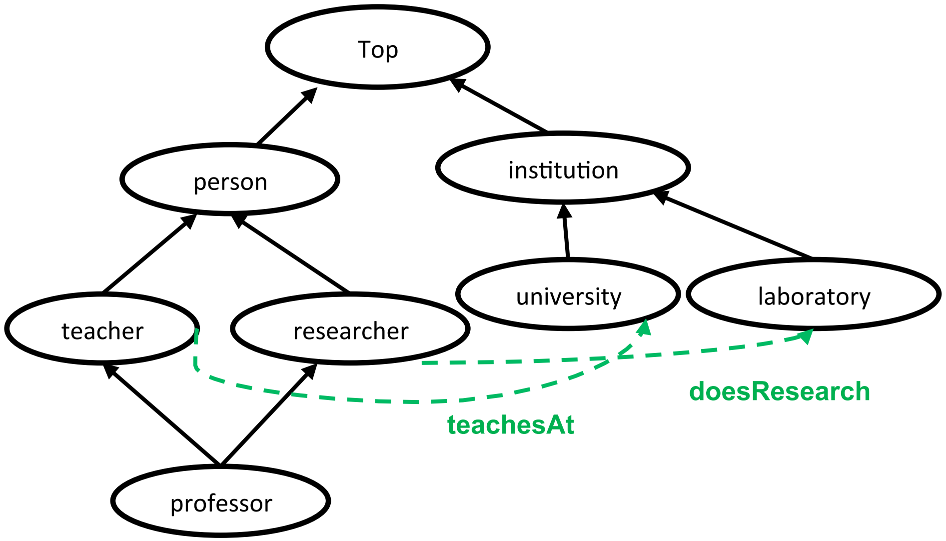
Top	**11111111**	**8**
person	**01111000**	**7**
teacher	**00101000**	**6**
researcher	**00011000**	**5**
professor	**00001000**	**4**
institution	**00000111**	**3**
university	**00000010**	**2**
laboratory	**00000001**	**1**

Q1 : person (teachesAt => institution, doesResearch => laboratory).

➔ person **and** teacher **and** researcher (teachesAt => institution **and** university, doesResearch => laboratory)

➔ **professor (teachesAt => university, doesResearch => laboratory)**

Query normalization



Top	11111111	8
person	01111000	7
teacher	00101000	6
researcher	00011000	5
professor	00001000	4
institution	00000111	3
university	00000010	2
laboratory	00000001	1

Q1 : person (teachesAt => institution, doesResearch => laboratory).

➔ person **and** teacher **and** researcher (teachesAt => institution **and** university, doesResearch => laboratory)

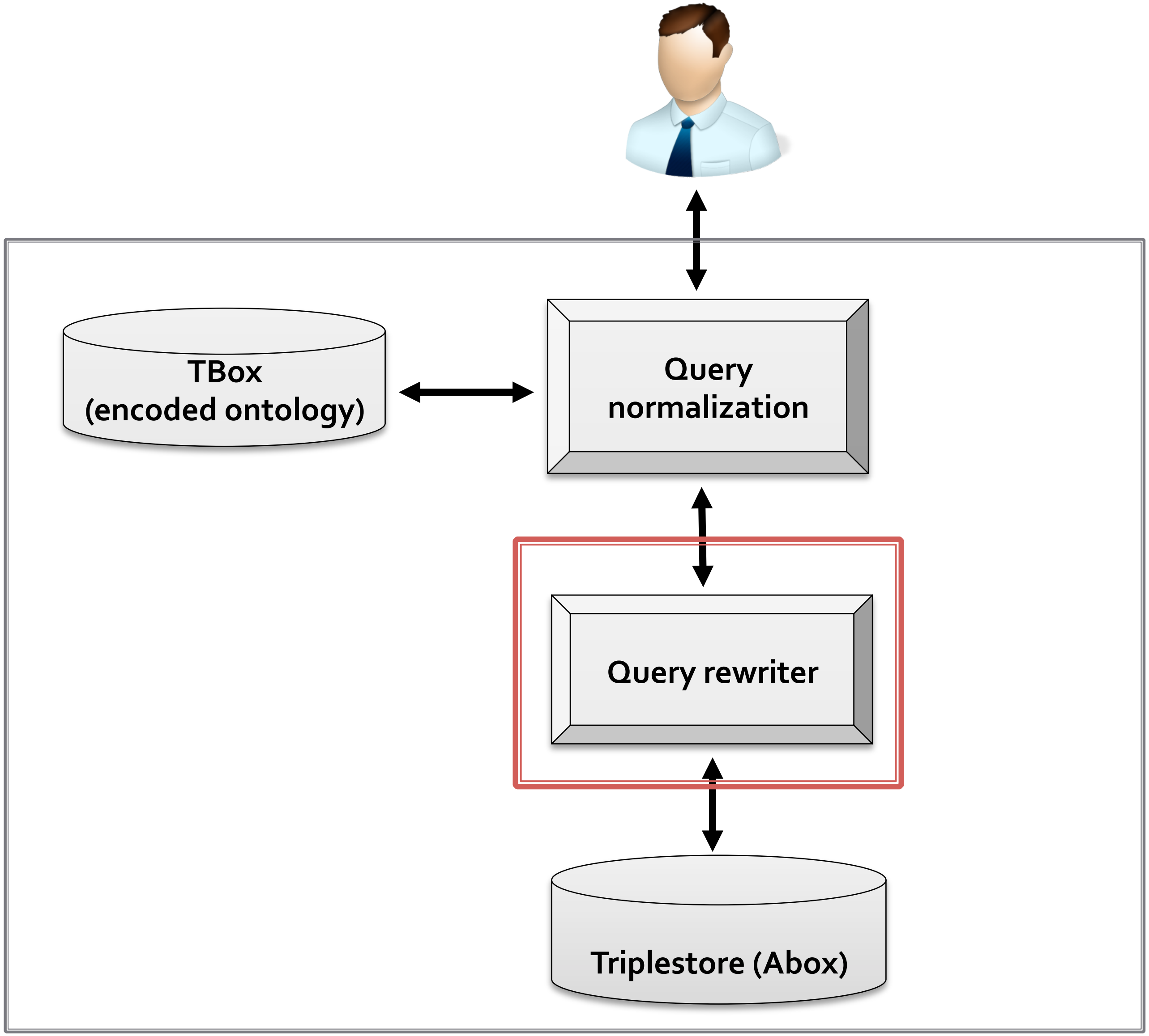
➔ professor (teachesAt => university, doesResearch => laboratory)

Q2 : person (teachesAt => laboratory).

➔person **and** teacher (teachesAt => laboratory **and** university)

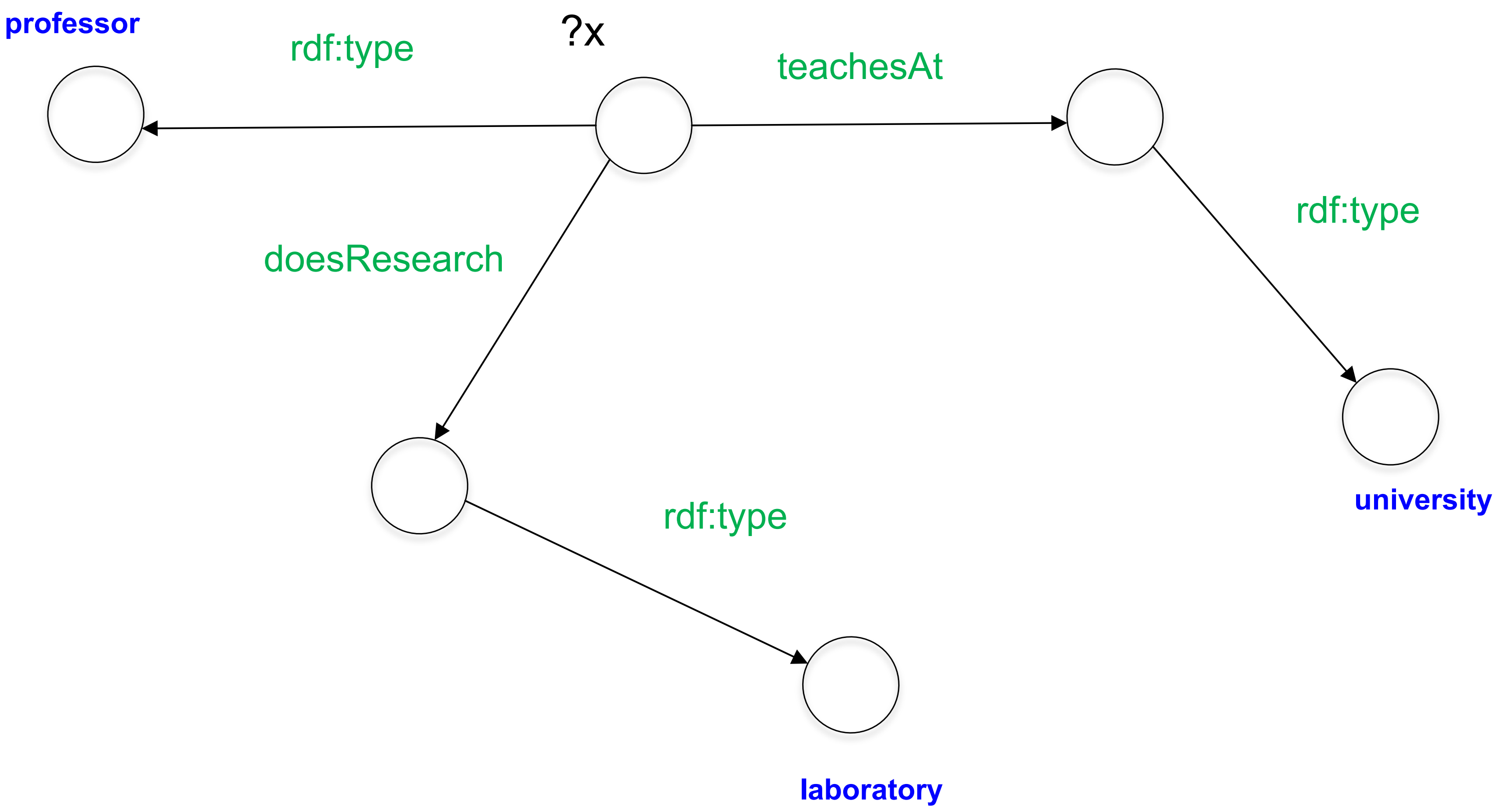
➔ Teacher (teachesAt => « Bottom ») ➔ **Q2 is not consistent**

Query rewriting



Example RDF : OSF mapping

professor (teachesAt => university , doesResearch => laboratory)



SPARQL query generation

Q1 without normalization

person (teachesAt => institution, doesResearch => laboratory)

Select x where {

?x rdf:type person.

?x teachesAt ?y.

?y rdf:type institution.

?x doesResearch ?z.

?z rdf:type laboratory

}

Q1 with normalization

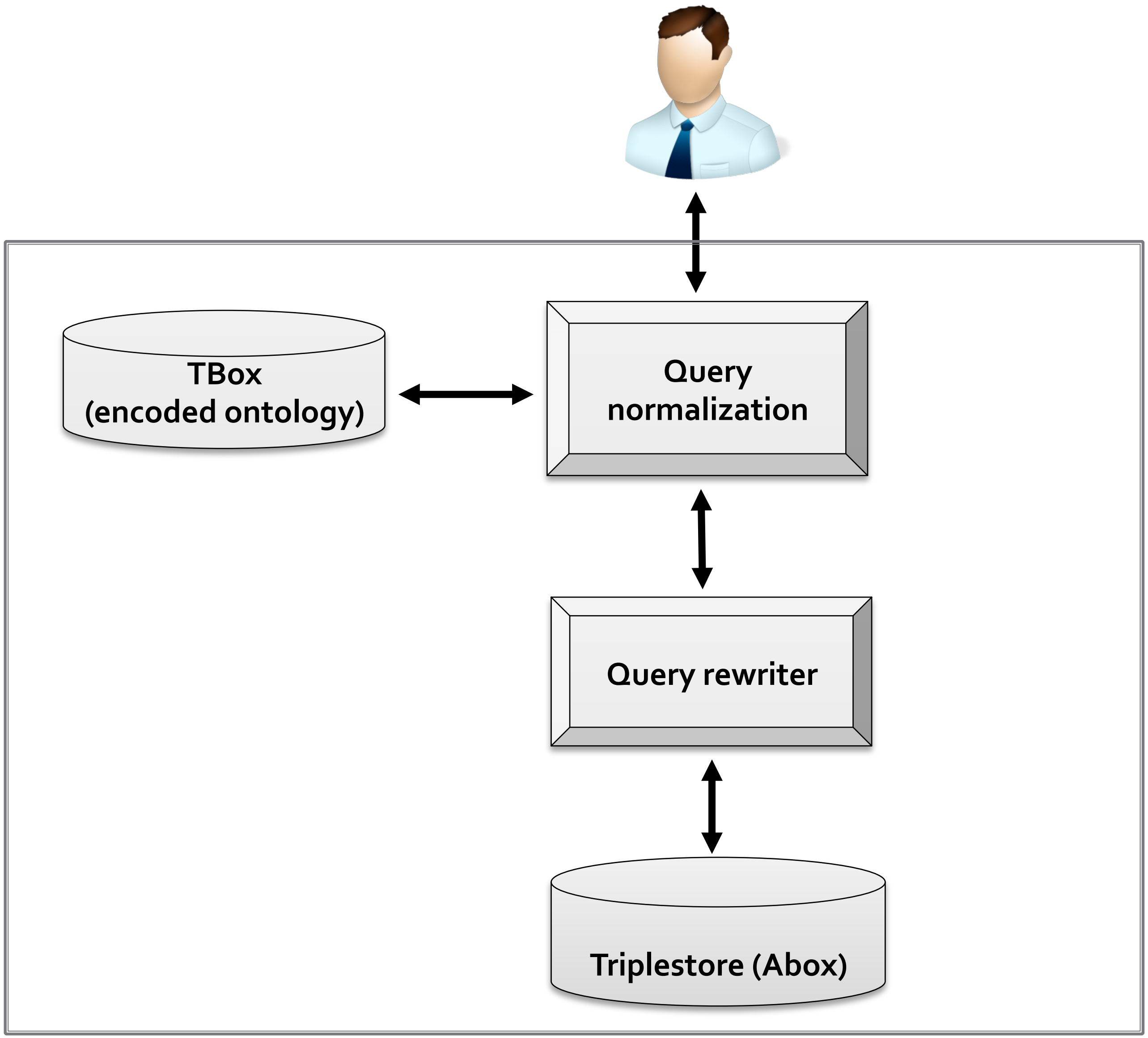
professor (teachesAt => university, doesResearch => laboratory)

Select ?x where {

?x rdf:type professor

}

CEDAR reasoner Building Blocks



Conclusion

- ❑ **Binary encoding**

  ➔ **Efficient Boolean query answering.**

- ❑ **Ontology is encoded, saved on the disk (no need to materialize)**

  ➔ **Materialization should be carried out each time the ABox changes.**
  ➔ **Materialization increases the number of facts.**

- ❑ **Normalization uses the TBox for query simplification and consistency check**

  ➔ **Simplify the query and reduce the search space.**
  ➔ **Detect the inconsistency.**

- ❑ **Type indexing**

  ➔**A has quick access to RDF triples.**

- ❑ **Support more complex Queries (disjunction, filtering, etc)**

Thank you