Rule-based Applications on Top of Ontologies

Architectures and Challenges

Georgios Meditskos

ITI-CERTH

Outline

- **Ontology Reasoning**
- **Combination of Ontologies and Rules**
- **The OWL 2 RL Profile**
  - Implementation aspects
  - The DLE framework
- **Conclusions**

Ontologies

- **Formal and explicit specifications of certain domains**
  - shared understanding of a domain
- **Form the backbone of the Semantic Web**
  - *"Web resources could be made much more usable if information was given a well defined meaning"*
  - *"define, annotate, share resources instead of documents"*
- **Provide shareable vocabularies of terms for annotation**
  - new terms can be formed by combining existing ones
  - specify relationships between terms of the same or different ontologies (links)
    - the meaning (semantics) is formally specified
    - reasoners can be used in order to derive implicit (hidden) knowledge

Examples of Existing Ontologies

- **The Gene Ontology**
  - in bioinformatics for representing and correlating genes
- **FOAF**
  - provides the schema for modelling persons, activities…
- **SKOS**
  - for defining classification schemes, taxonomies…
- **Music Ontology**
  - artists, albums, tracks …
- **Programmes Ontology (funded by BBC)**
  - describing programmes, episodes, events
  - reuses FOAF and the Music Ontology
- **many others…**

OWL 2 (A Revision of OWL)

- **A language for defining ontologies**
  - RDF/XML is the basic syntax

```
<owl:Class rdf:ID="Man">
        <rdfs:subClassOf rdf:about="Human" />
</owl:Class
```

"Man is subclass of Human"

- **Became a W3C recommendation in 2009**
  - A revision of OWL (2004) (more constructs and syntactic sugar)
- **An ontology consists of**
  - Classes and properties (TBox/schema)
  - Instances (ABox)

Class Constructors

- **Operators for constructing class expressions**

Constructor	Example
Classes	Human
intersectionOf	intersectionOf(Human Male)
unionOf	unionOf(Doctor Lawyer)
complementOf	complementOf(Male)
oneOf	oneOf(john mary)

Class Constructors

- **Class definitions based on property restrictions**

Constructor	Example
someValuesFrom	restriction(hasChild someValuesFrom Lawyer)
allValuesFrom	restriction(hasChild allValuesFrom Doctor)
minCardinality	restriction(hasChild minCardinality (2))
maxCardinality	restriction(hasChild maxCardinality (2))

An Introduction to OWL ISWC 2008

- **These are restrictions and not (integrity) constraints**

Example (restrictions)

Human ☐∀hasParent.Human

(all the values in the hasParent property belong to the Human class)

- **We can define an instance of Human without a value in the hasParent property**
  - interpretation*: we do not know the parent(s) yet, they may be defined in the future (Open-World Semantics)*
- **If we define a value for the hasParent property, then this value SHOULD belong to the Human class**
  - a reasoner classifies the instance value to the Human class, even if it belongs to some other class
    - it derives more inferences instead of checking the model

Axioms

- **Add further statements about concepts, properties and instances**

Axiom	Example
SubClassOf	SubClassOf(Human Animal)
EquivalentClasses	EquivalentClass(Man intersectionOf(Human Male))
DisjointClasses	DisjointClasses(Animal Plant)

An Introduction to OWL ISWC 2008

Individual Axioms

Axiom	Example
Individual	Individual(george type(Human))
Individual	Individual(george value(worksWith nick))
DifferentIndividuals	DifferentIndividuals(george nick)
SameIndividualAs	SameIndividualAs(GeorgeWBush PresidentBush)

An Introduction to OWL ISWC 2008

- Even if two or more individuals have different IDs, they may refer to the same resource

Property Axioms

Axiom	Example
SubPropertyOf	SubPropertyOf(hasMother hasParent)
domain	ObjectProperty (owns domain(Person))
range	ObjectProperty (employs range(Person))
transitive	ObjectProperty(hasPart Transitive)

An Introduction to OWL ISWC 2008

- Note: Properties are first-class citizens, so they may have properties

OWL 2 Profiles

- **Help developers to choose the right constructs**
  - in terms of modelling and reasoning capabilities
- **Trimmed down versions of OWL 2**
  - trade some expressive power for the efficiency of reasoning
- **Three profiles**
  - **OWL 2 EL**: useful in applications that employ ontologies with large number of properties and classes
  - **OWL 2 QL**: useful for efficient query answering, quite limited expressive power
  - **OWL 2 RL**: scalable reasoning using rule engines without sacrificing too much expressive power

we will describe this profile later

OWL Reasoning

- **Discover inferences based on asserted information**
  - deduce implicit knowledge
- **Main reasoning tasks**
  - **Subsumption**: compute all the subclass relationships among the classes (e.g. if concept A subsumes concept B)
  - **Consistency**: check if the assertions in a KB have a model (satisfiability)
  - **Realization**: compute the instance class memberships (e.g. the set of instances that belong to a certain concept)
    - query answering: *give me all the instances of the class Man*

DL Reasoning

- **OWL is based on Description Logics (DL)**
  - syntactic fragments of First-Order Logic (FOL)
- **Existing DL algorithms can be used**
  - e.g. tableaux
  - NEXPTIME-complete
    - scalability issues, mainly in large Aboxes
    - good performance in TBox reasoning
- **Basic characteristics of DL reasoning**
  - sound and complete reasoning paradigm
  - Open-World Assumption (OWA)
    - the lack of information is NOT equivalent to negative information (in contrast to database-like applications)
  - it does not follow the Unique Name Assumption (UNA) (different identifiers may reference the same resource)

Expressive Limitations of OWL

- **Need to model complex relationships beyond the expressiveness of OWL**
  - e.g. OWL is not able to capture relationships between a composite property and another property
    - standard example: composition of the ‘parent’ and ‘brother’ properties and the ‘uncle’ property
  - OWL 2 allows property chains only if the composite property is subproperty of one of the composed properties
- **Solution**
  - Combine ontologies and rules

Semantic Web Rule Language (SWRL)

- **The approach**
  - extending ontologies with first-order rules
    - hasUncle(x, z) ← hasFather (x, y), hasBrother (y, z)
  - SWRL follows the OWA and UNA but it is not decidable
    - In practice reasoners implement **Safe Rules**: each variable from the rule head must occur in the rule body

Logic Programming and Ontologies

- **There are many differences between LP and OWL (DL)**
  - Open vs. Closed-World Semantics
    - Datalog follows the CWA
  - Unique Name Assumption
    - Datalog follows the UNA (different identifiers always denote different objects)
- **However…**
  - Reasoning Complexity
    - Datalog -> performs reasoning in polynomial time
    - DL algorithms -> NEXPTIME-complete
- **Despite the differences there are many approaches**
  - e.g. hybrid, homogenous, intersection of DL and LP (Description Logic Programs)…
- **We will focus on the mapping of OWL on rules**
  - OWL 2 RL (in OWL 2)

OWL 2 RL

- **Syntactic subset of OWL 2**
  - e.g. it cannot infer individuals not explicitly present in the KB
- **The semantics can be implemented using rule-based technologies**
  - forward or backward chaining rule engines
- **Benefits**
  - scalable reasoning on a quite expressive subset of OWL 2
    - implemented by many state of the art large scale reasoners
  - enables the definition of rule-based applications on top of ontologies

Reasoning in OWL 2 RL

- **The reasoning is performed based on a predefined set of *entailment rules***
  - known as OWL 2 RL/RDF rules
- **Reasoning tasks**
  - ontology consistency, class satisfiability, class subsumption, instance checking, conjunctive query answering
- **Reasoning complexity**
  - PTime-complete (can be performed in polynomial time with respect to the size of the ontology)
- **The entailment rules follow a triple-based representation**
  - run on top of RDF Graphs (RDF triples)

Triples

- **Statements of the form (s p o)**
  - s: subject, p: predicate, o: object
- **All the ontologies can be represented as a set of triples**
  - a simple serialization of RDF/XML syntax
  - a triple corresponds to a “fact” in a rule engine
- **Example**

  RDF/XML: <owl:Class rdf:ID=“Person” />

  triple: (Person rdf:type owl:Class)

Entailment Rules

- **Simple if-then rules that assert new triples based on existing ones**

  condition-triples → conclusion-triples

- **Conditions contain triple patterns with variables**
  - triple patterns match existing triples (facts)
- **Conclusions derive new triples**
  - based on the variable bindings in the conditions
  - Safe Rules: a variable in the conclusion should exist in the condition

Example rule (subclass transitivity)

(?B subClassOf ?C), (?C subClassOf ?D)
→ (?B subClassOf ?D)

- **Assuming that the KB contains the triples (facts)**

  (Boy subClassOf Man)

  (Man subClassOf Human)

- **then the rule asserts the fact**

  (Boy subClassOf Human)

Example rule (class membership)

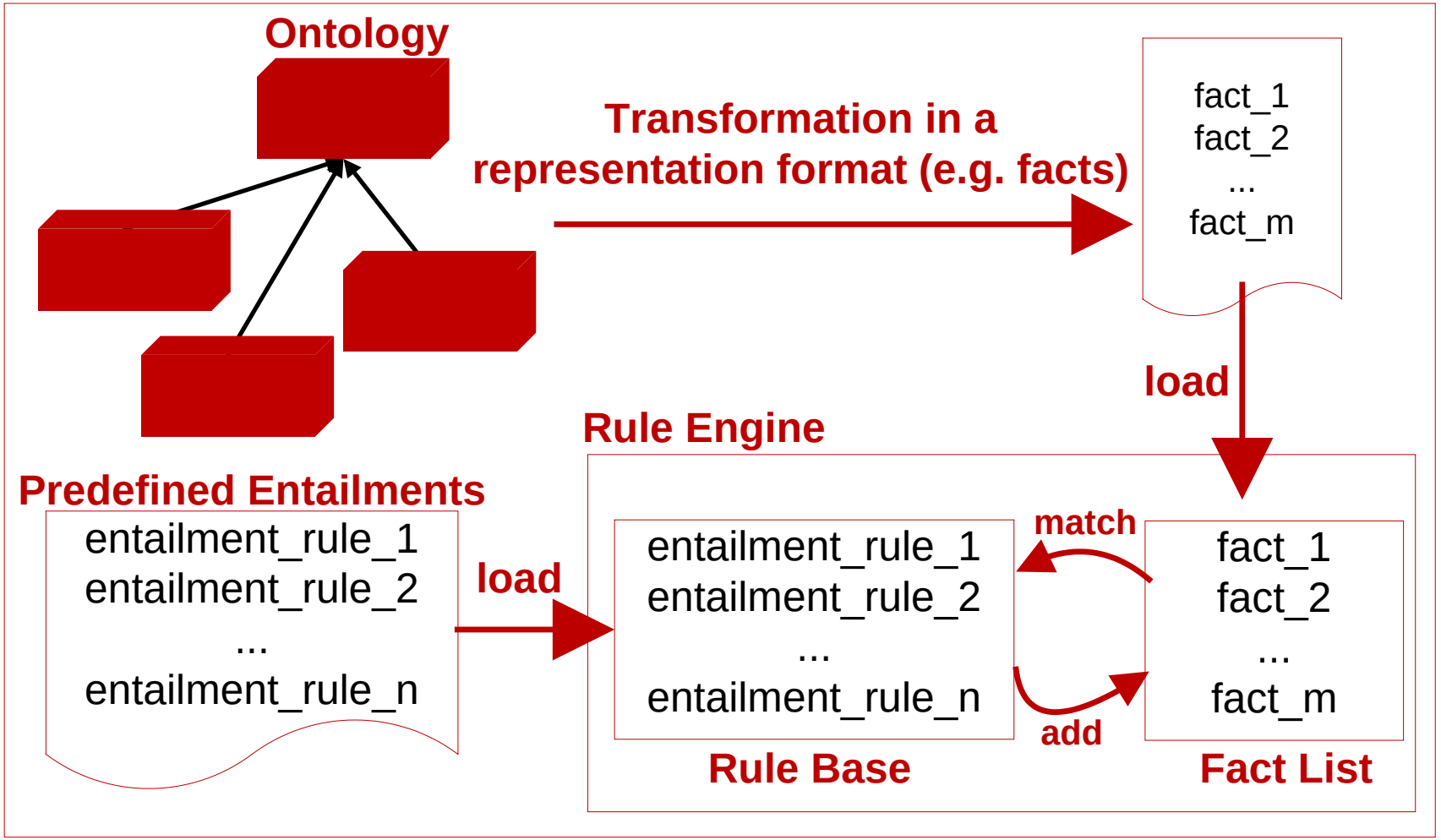
- **(?B subClassOf ?C), (?T type ?B)**
  → (?T type ?C)
- **Assuming that the KB contains the triples (facts)**

  (Boy subClassOf Man)

  (george type Boy)
- **then the rule asserts the fact**

  (george type Man)

A Typical Architecture



Custom Rules

- **Apart from entailment rules for reasoning, someone can define custom ones**
  - domain-dependent (rule-based applications)
- **The inference rules and the custom rules coexist in the same rule base**
- **This approach is followed by many reasoners**
  - Jena, Bossam, OWLIM, AllegroGraph…

The DLE Framework

- **An improvement of the typical entailment-based architecture**
  - more scalable ABox reasoning
- **Basic idea: 2 reasoning paradigms**
  - DL reasoning on the TBox
  - entailment rules for the ABox
    - the rules are generated dynamically
    - they are simple -> faster execution

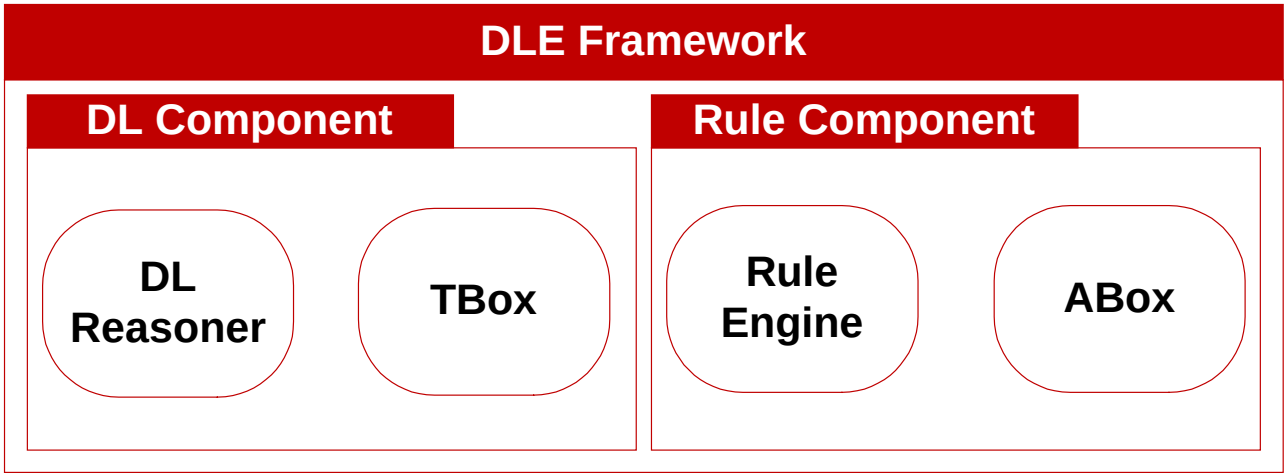
G. Meditskos, N. Bassiliades, "*Combining a DL Reasoner and a Rule Engine for improving Entailment-based OWL Reasoning*", 7th International Semantic Web Conference (ISWC), pp. 277-292, Karlsruhe, Germany, 2008

Entailment Rule Classification

- **Terminological**
  - the conditions have only TBox triple patterns
  - e.g. (?B subClassOf ?C), (?C subClassOf ?D) → (?B subClassOf ?D)
- **Hybrid**
  - the conditions have both TBox and ABox triple patterns
  - e.g. (?B subClassOf ?C) (?T type ?B) → (?T type ?C)
- **Exceptional**
  - the conditions contain only ABox triple patterns
  - e.g. (?B sameAs ?C) → (?C sameAs ?B)

Reasoning on DLE

- **DL Component**
  - DL reasoner for TBox reasoning
  - substitutes the terminological entailments
    - e.g. there is no need to implement the entailment for subclass transitivity
- **Rule Component**
  - exceptional entailments
  - **instantiated versions of the hybrid entailments**: they are generated dynamically based on the TBox reasoning results

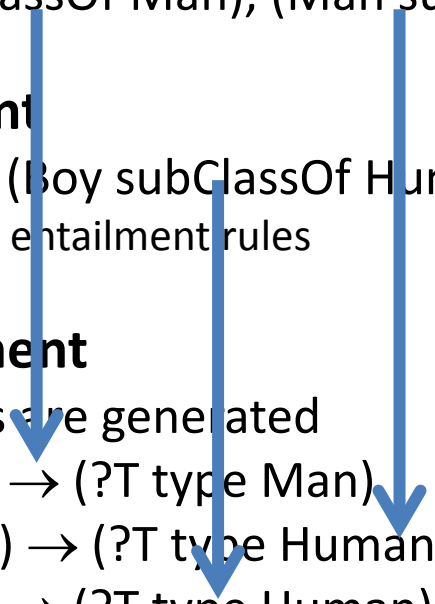


Instantiating the Hybrid Rules - Example

Class subsumption entailment rule
(?A subClassOf ?B) (?T type ?A) → (?T type ?B)

- **Assuming the triples:**
  - (Boy subClassOf Man), (Man subClassOf Human)
- **DL component**
  - Infers that (Boy subClassOf Human)
    - without entailment rules
- **Rule component**
  - three rules are generated

(?T type Boy) → (?T type Man)
(?T type Man) → (?T type Human)
(?T type Boy) → (?T type Human)



Although more rules are applied, the ABox reasoning procedure terminates faster

DLEJena

- **Implementation of the DLE framework**
  - **Jena:** a very popular framework for rule-based ontology reasoning
  - **Pellet:** DL reasoner
- **Goal:**
  - to improve the performance of ABox reasoning in Jena (that uses predefined entailment rules)

G. Meditskos, N. Bassiliades, "*DLEJena: A Practical Forward-Chaining OWL 2 RL Reasoner Combining Jena and Pellet*", Web Semantics: Science, Services and Agents on the World Wide Web, vol. 8, issue 1, pp. 89-94, March 2010

Rule Templates

- **Generate the entailment rules at runtime**
  - production rules that generate production rules
    - the conclusion is actually a production rule
- **Example (instance class membership)**

(?B subClassOf ?C) (?T type ?B) → (?T type ?C)

```
[ (?B subClassOf ?C)
    -> [(?T type ?B) -> (?T type ?C)]]
```

the number of the generated rules depends on
the number of subclass axioms in the TBox

Experiments

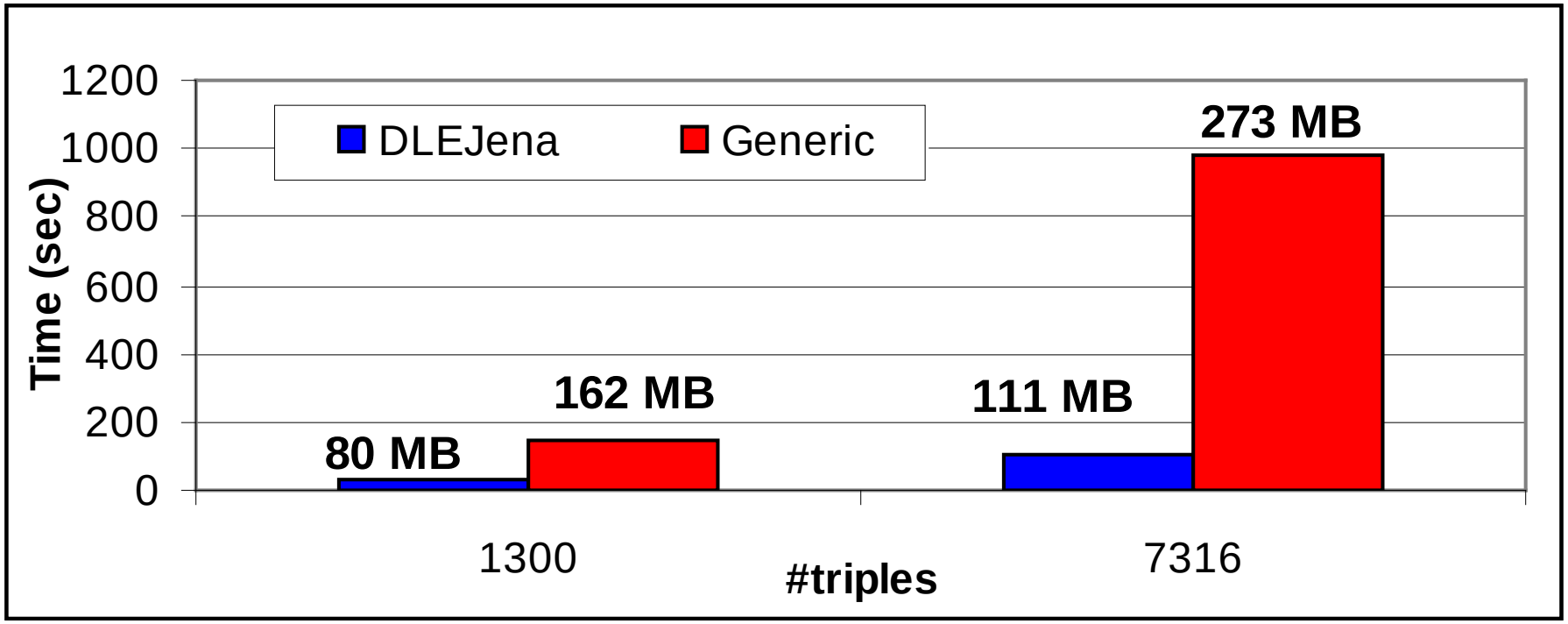
- **Two implementations**
  - **generic implementation:** typical architecture
    - 41 predefined entailment rules
  - **DLEJena:** dynamically generated entailments
    - no static rule set
- **Goal**
  - to compare the two implementations in terms of ABox reasoning time and memory requirements

Wine Ontology

(wines and foods and appropriate combinations of wines with meals)

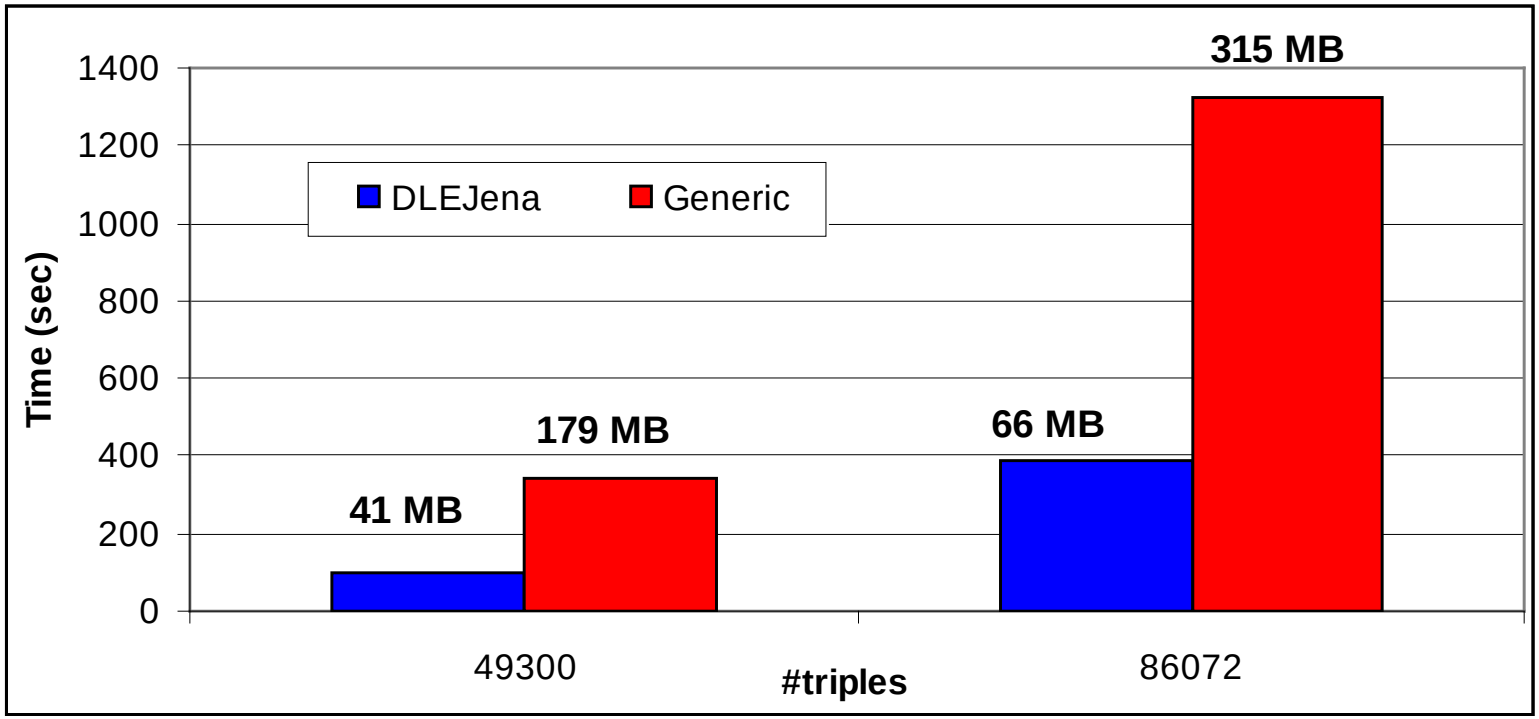
- **# DLEJena rules: 9726**
- **# Generic rules: 41**

- **DLEJena achieves better performance**
- **scalability issues (reasoning time) for the generic implementation**



UOBM Ontology (university domain)

- **# DLEJena rules: 1072**
- **# Generic rules: 41**
- **scalability issues for the generic implementation**



Conclusions

- **Large-scale ontology reasoning is a challenging task**
- **DL reasoners are sound and complete**
  - but they do not scale (consider a dataset of billions of triples…)
- **Rule-based reasoners are able to handle a subset of the ontology semantics**
  - but very efficiently
  - enable the development of rule-based applications
- **Rule-based reasoning is the standard reasoning paradigm for very large datasets**
  - e.g. in Linked Open Data cloud where there is a need for reasoning on large (interlinked) datasets