Ontology Reasoning for the Semantic Web and Its Application to Knowledge Graph

Guilin Qi

Knowledge Science and Engineering (KSE) Lab

Southeast University

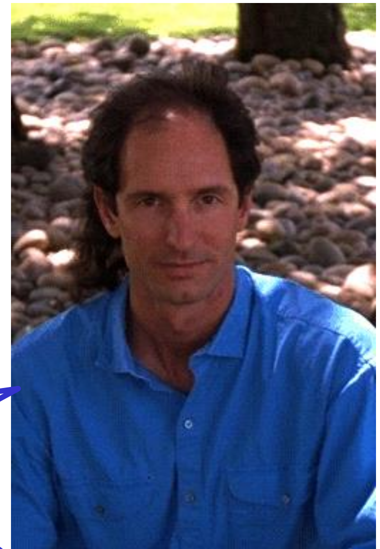
gqi@seu.edu.cn

Plan of the Talk

- **Part I: Introduction of Ontology Languages for the Semantic Web**
- **Part II: Application of Ontology Reasoning**
- **Part III: Reasoning with Large Imprecise Knowledge on the Semantic Web**

Ontology

- Different definitions: philosophy, AI,…
- Definition in Semantic Web:



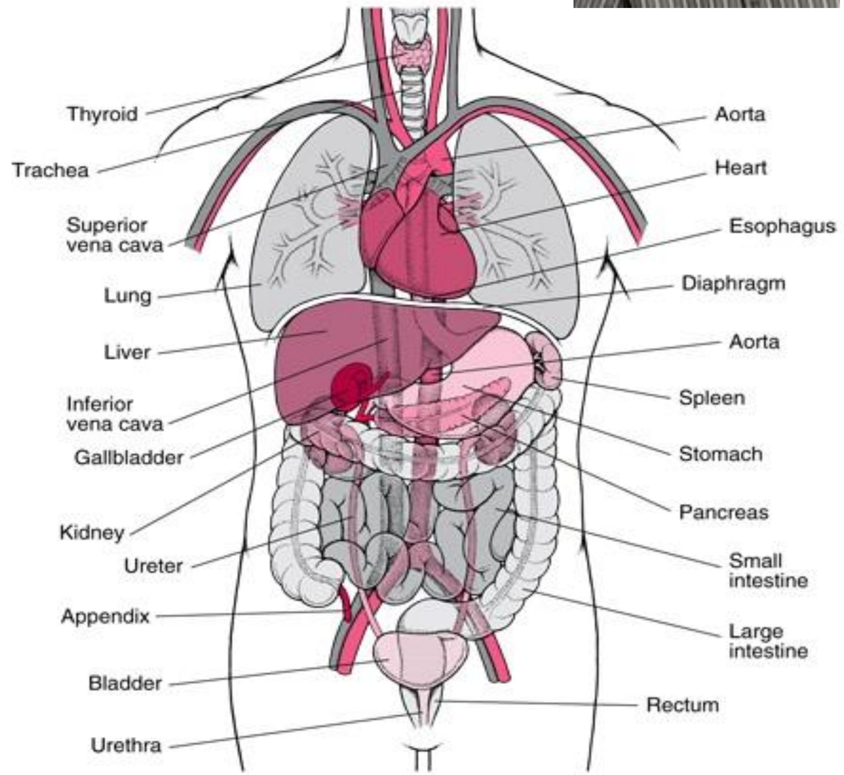
Gruber, 1993

An ontology is an explicit specification of a conceptualization

What is an Ontology?

A model of (some aspect of) the world

- **Introduces vocabulary relevant to domain, e.g.**
  - **Anatomy**

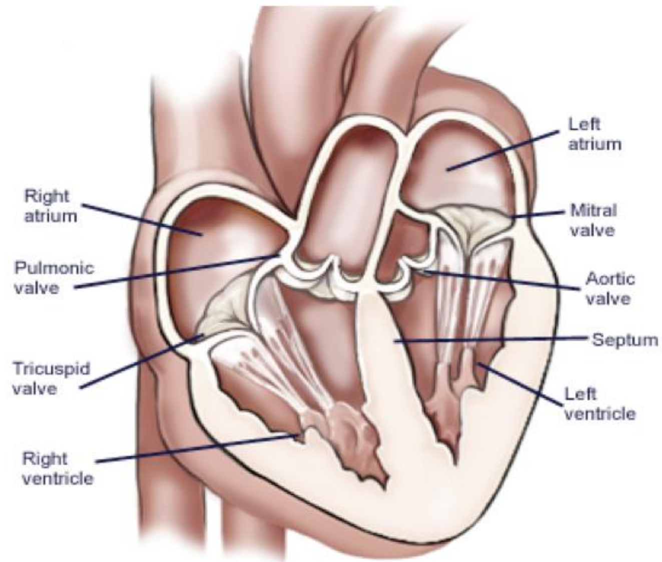


What is an Ontology?

A model of (some aspect of) the world

- **Introduces vocabulary relevant to domain**
- **Specifies meaning (semantics) of terms**

Heart is a muscular organ that is part of the circulatory system



Ontologies

- **The core of an ontology is usually a *taxonomy*:**
  - **classes of things, arranged in a hierarchy**

```
Human
 └MaleHuman              [every MaleHuman is a Human]
     └Son                [every Son is a MaleHuman]
         └Father         [every Father is a Son]
 └FemaleHuman
     └Daughter
         └Aunt
         └Niece
         └Mother
             └GrandMother
```

Ontology languages

- RDF (Resource Description Framework)
  - Specifies relationship between data
- RDFS(Resource Description Framework Schema)
  - Specifies relationship between schema
- OWL (Web Ontology Language)
  - Specifies more complex relationship between schema based on description logics

RDF idea

- **Use (directed) graphs as data model**

Subject **Predicate** **Object**

- **"Resource Description Framework"**

RDF Schema (RDFS)

- part of the W3C Recommendation RDF
- for schema/terminological knowledge
- uses RDF vocabulary with pre-defined semantics

Classes and Instances

- Classes stand for sets of things.
  In RDF: Sets of URIs.
- book:uri is a member of the class ex:Textbook

```
book:uri    rdf:type    ex:Textbook .
```

- a URI can belong to several classes

```
book:uri    rdf:type    ex:Textbook .
book:uri    rdf:type    ex:WorthReading .
```

- classes can be arranged in hierarchies: each textbook is a book

```
ex:Textbook    rdfs:subClassOf    ex:Book .
```

Implicit knowledge

- if an RDFS document contains

```
u    rdf:type    ex:Textbook .
```

and

```
ex:Textbook    rdfs:subClassOf    ex:Book .
```

then

```
u    rdf:type    ex:Book .
```

is *implicitly* also the case: it's a *logical consequence*. (We can also say it is *deduced* (deduction) or *inferred* (inference)

Implicit knowledge – another example

- **From**

```
ex:Textbook    rdfs:subClassOf    ex:Book .
ex:Book        rdfs:subClassOf    ex:PrintMedia .
```

the following is a logical consequence:

```
ex:Textbook    rdfs:subClassOf    ex:PrintMedia .
```

I.e. rdfs:subClassOf is *transitive*.

Using implicit knowledge

Ontology (Knowledge Base)
e.g. RDF or OWL

Reasoner (produces implicit knowledge)

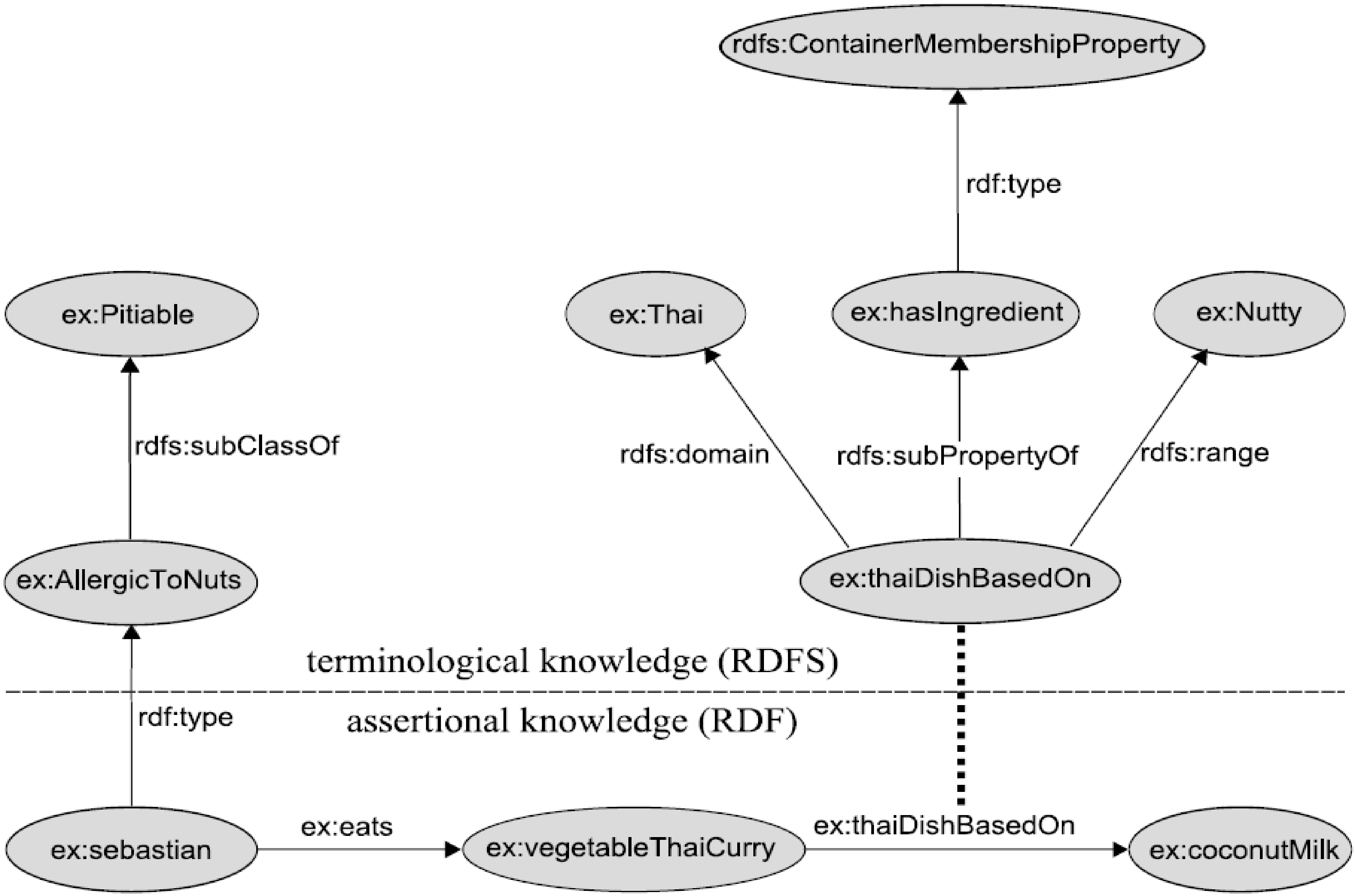
offline

Completed (materialized) knowledge base

Used like a database

Application

The same as graph



OWL – Web Ontology Language

Why do we need OWL?

Project Is either
Internal or
External

```
<owl:Class rdf:about="project">
  <rdfs:subClassOf>
     <owl:Class>
       <owl:unionOf
rdf:paseType="Collection">
         <owl:Class rdf:about="internal"/>
         <owl:Class rdf:about="external"/>
       </owl:unionOf>
     </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
```

OWL 2 Profiles

- **The OWL 2 spec describes three profiles (fragments, sublanguages) which have polynomial complexity.**
  - **OWL EL (the description logic EL++)**
    - **Represent medical knolwedge**
  - **OWL QL (the description logic DL Lite$_R$)**
    - **Targeted to data integration**
  - **OWL RL (the description logic DLP)**
    - **inspired by intersecting OWL with Datalog**
    - **implemented e.g. in Oracle 11g**

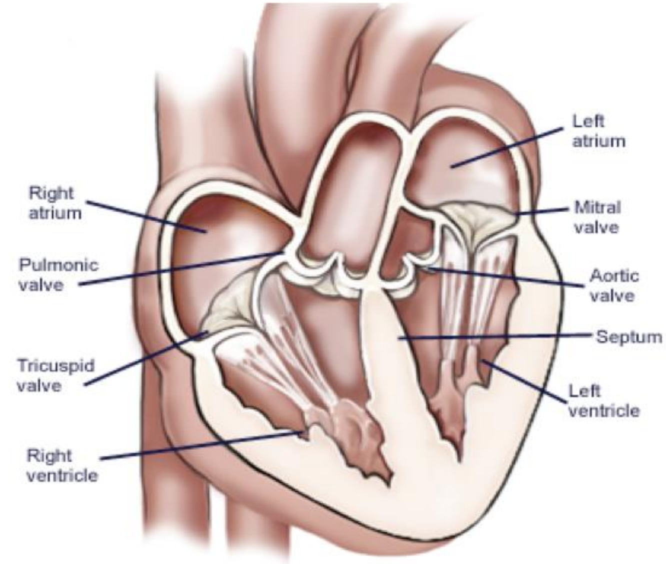
Description Logics

- **Description logics**
  - **Are (mostly) decidable fragments of first-order predicate logic**
  - **Provide logical underpinning of W3C standard OWL**
- **Building blocks**
  - **Concepts (unary predicates/formulae with one free variable)**
    - **E.g., Person, Lawer ⊔ Doctor**
  - **Roles (binary predicates/formulae with two free variables)**
    - **E.g., hasChild**
  - **Individuals (constants)**
    - **E.g., John, Mary**

Description Logics (Syntax)

- Description languages
  - Defining complex concepts: sets of individuals
  - Defining complex roles: binary relations on individuals
- Complex concepts are built by
  - Atomic concepts: Tissue, Heart
  - Constructors: Tissue$\sqcap\exists$part-of.Heart
- Complex roles are built by
  - Atomic roles: part-of, has-location
  - Constructors: HasFather^{-}

Example



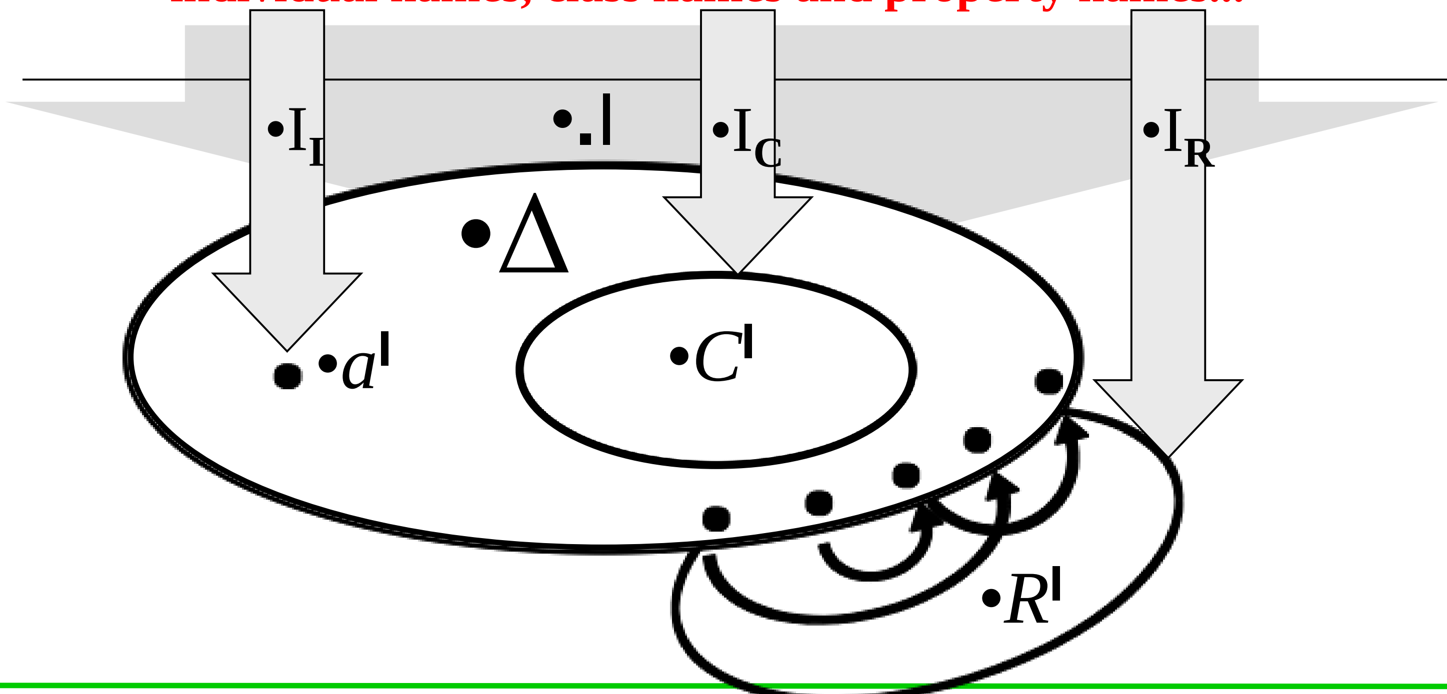
Heart is a muscular organ that is part of the circulatory system

Heart⊑MuscularOrgan⊓∃part-of.CirculatorySystem

Description Logics (Semantics)

- **Interpretation: $I=(\Delta^I,.^I)$**
  - **Domain: Δ^I**
  - **Assignment function $.^I$**

individual names, class names and property names...



Description Logics (Cont.)

- **Interpretation: $I=(\Delta^I,.^I)$**

Construct	Syntax	Example	Semantics
Atomic concept	A	Heart	$A^I \subseteq \Delta^I$
Atomic role	R	part-of	$R^I \subseteq \Delta^I \times \Delta^I$
Negation	$\neg$ C	$\neg$ Heart	$\Delta^I \setminus C^I$
Conjunction	C $\sqcap$ D	Lawyer$\sqcap$Doctor	$C^I \cap D^I$
Value restriction	$\forall$ R.C	$\forall$ part-of.Wood	$\{a \mid \forall b. (a,b) \in R^I, b \in C^I\}$
...	...	...	...

Description Logics (Ontology)

- **TBox $\mathcal{T}$: defining terminology of application domain**
  - **Inclusion assertion on concept :C ⊑ D**

    Pericardium ⊑ Tissue ⊓ ∃ part-of.Heart

  - **Inclusion assertion on roles: R ⊑ S**

    Part-of ⊑ has-location

- **ABox $\mathcal{A}$: stating facts about a specific "world"**
  - **membership assertion: C(a) or R(a,b)**

    HappyMan(Bob), HasChild(Bob, Mary)

Description Logics(Semantics)

- Given an interpretation I
- Semantics of TBox axioms
  - $I \models C \sqsubseteq D$ if $C^I \subseteq D^I$
  - $I \models R \sqsubseteq S$ if $R^I \subseteq S^I$
- Semantics of ABox assertions
  - $I \models C(a)$ if $a^I \in C^I$
  - $I \models R(a,b)$ if $(a^I,b^I) \in R^I$

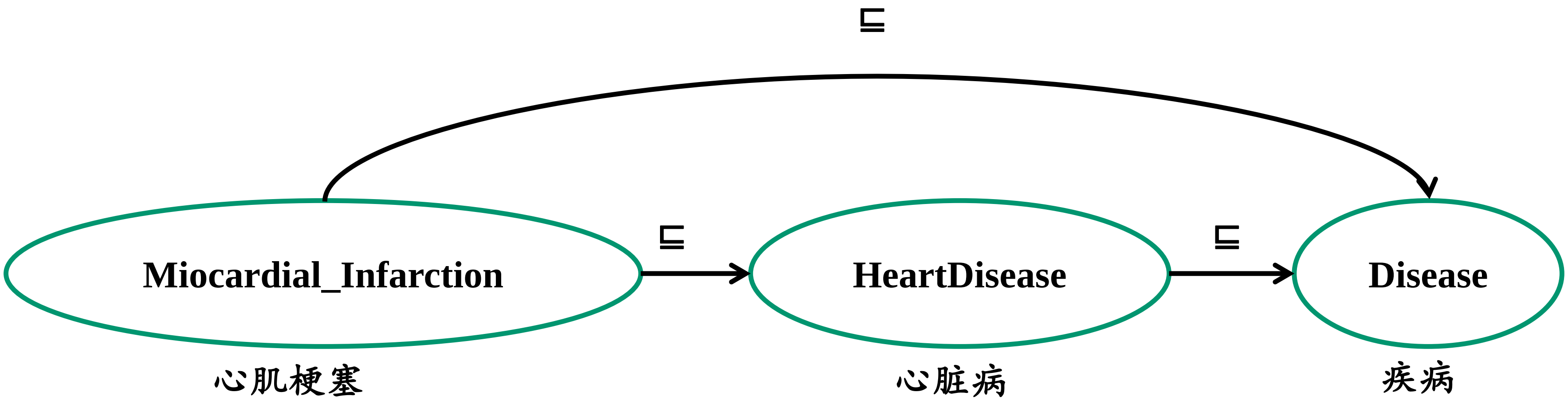
Description Logics(Semantics)

- **Model of an ontology $O=\langle T, A\rangle$**
  - I is a model of O if it satisfies all axioms in T and all assertions in A
- **Concept satisfiability**
  - Concept C is satisfiable in O if C^I is nonempty for some model I of O
- **Ontology Entailment:**
  - $O \models \phi$ iff $I \models \phi$ for all models I of O

Conclusion

- RDF is a flexible data model for Semantic Web
- RDF Schema provides simple inference capability
- OWL allows more expressive representation of knowledge but is hard to scale to Web data
- Semantic technologies have been adopted by major companies such as Google, Yahoo and Facebook

Reasoning Task: Classification



Reasoning Task: Classification

Example

Endocardium	$\sqsubseteq$	Tissue $\sqcap$ $\exists$ cont-in.HeartWall $\sqcap$ $\exists$ cont-in.HeartValve
HeartWall	$\sqsubseteq$	BodyWall $\sqcap$ $\exists$ part-of.Heart
HeartValve	$\sqsubseteq$	BodyValve $\sqcap$ $\exists$ part-of.Heart
Endocarditis	$\sqsubseteq$	Inflammation $\sqcap$ $\exists$ has-loc.Endocardium
Inflammation	$\sqsubseteq$	Disease $\sqcap$ $\exists$ act-on.Tissue
HeartDisease $\sqcap$ $\exists$ has-loc.HeartValve	$\sqsubseteq$	CriticalDisease
HeartDisease	$\sqsubseteq$	Disease $\sqcap$ $\exists$ has-loc.Heart

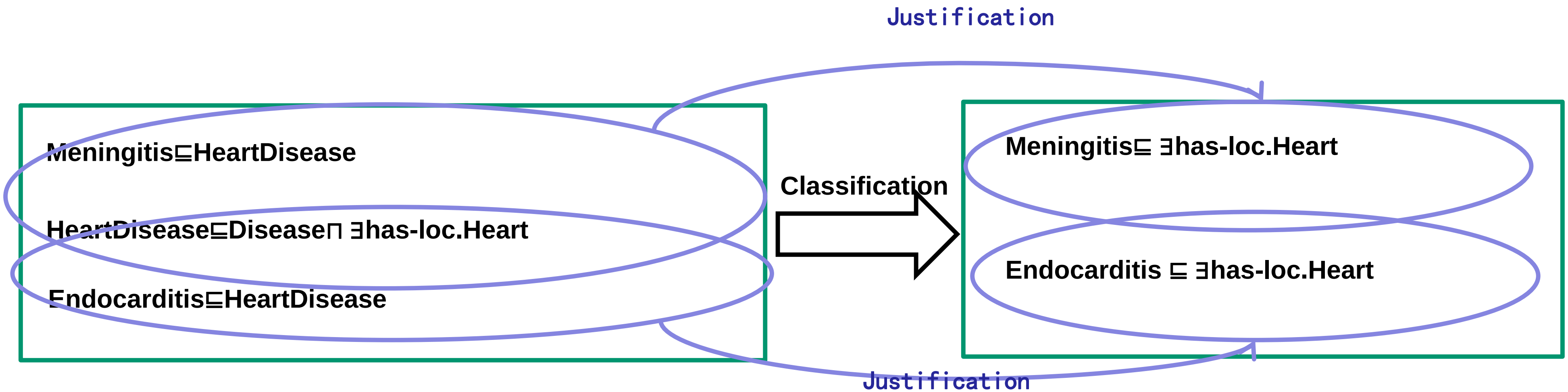
Endocarditis $\sqsubseteq$ HeartDisease

Endocarditis $\sqsubseteq$ CriticalDisease

Role of Classification

a) Enrich ontology
b) Query writting
c) Check satisfiability of KB

Reasoning Task: Finding Justification



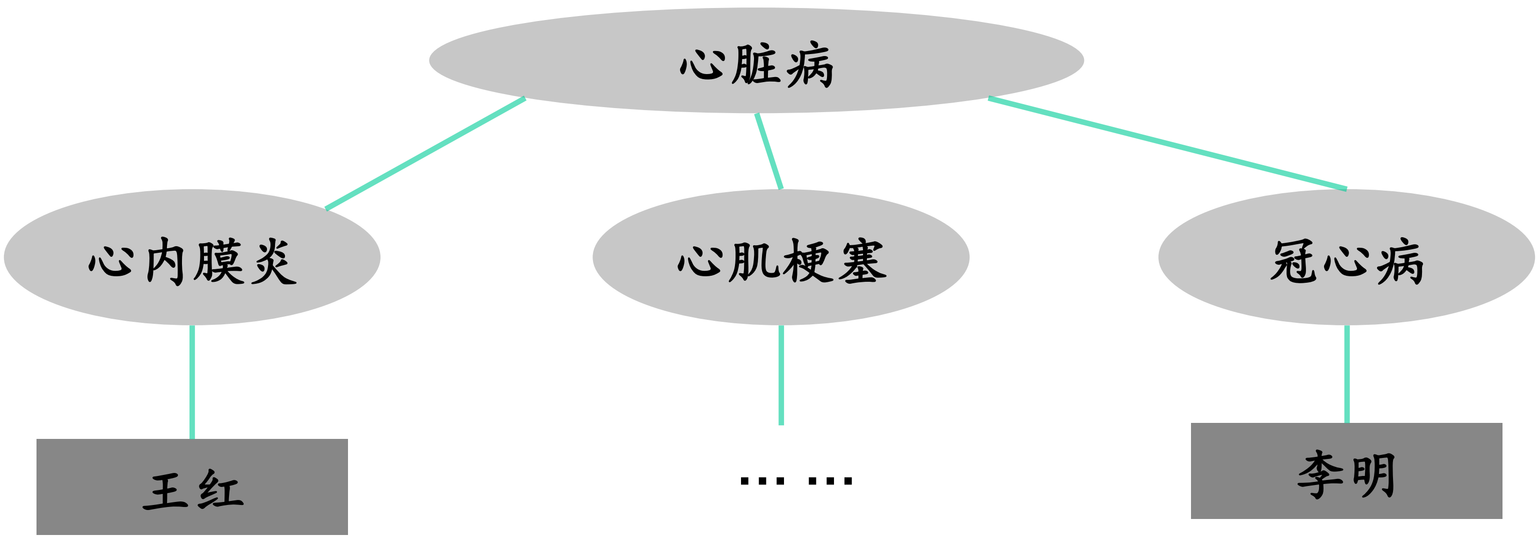
- ❒ **Repair KB using justifications**
  - ❖ **Through classification, we have “Meningitis⊑ ∃has-loc.Heart”**
  - ❖ **After finding justification, we found “Meningitis⊑HeartDisease” is wrong**

Reasoning Task: Query Rewritting

- Suppose we have the following query
  “心脏病患者有哪些？”
- This query can be used for medical statistics

Reasoning Task: Query Rewritting

- Suppose we have the following ontology

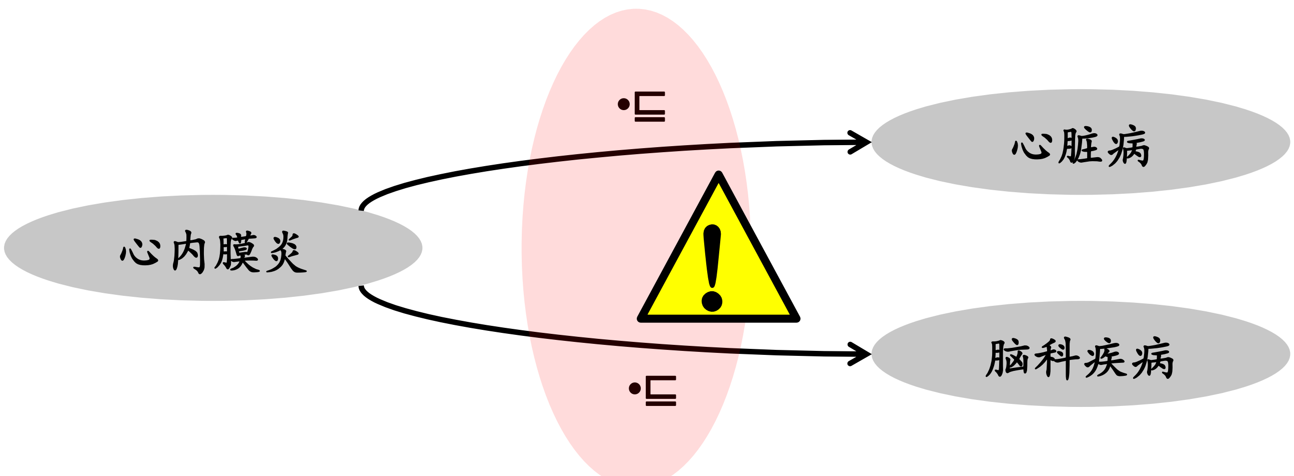


- We know“王红”is 心内膜炎患者，we should include her as心脏病患者

List(X) :- Endocarditis(X) ∨ Miocardial_Infarction(X) ∨ Coronary_disease(X).

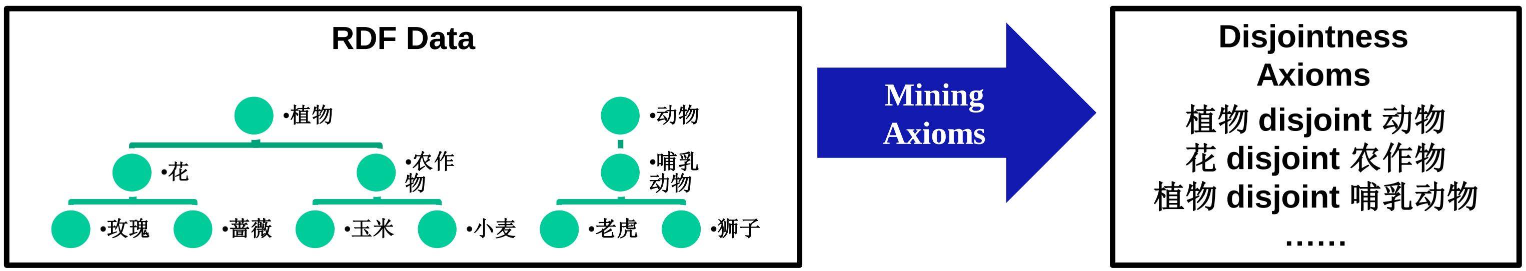
Reasoning Task: Checking Inconsistency

- Inconsistency may occur during ontology construction
  - One source of inconsistency comes from disjoint axioms



Reasoning Task: Checking Inconsistency

- Mining disjoint Concepts：Association rule mining, Inductive logic programming



- In Zhishi.me, hudong:大豆食心虫 not only belongs to animal but also contains in plant
- In Zhishi.me, there are 50 common instances between animal and plant

Reasoning Task: Checking Inconsistency

- In Dbpedia, we find 42153 disjointness axioms by mining axioms algorithm

Number of Common individuals	Pairs of disjoint classes
[1,10)	317
[10,100]	27
[100,1000)	7

Reasoning Task: Checking Inconsistency

- **Detecting noisy type assertion**

Disjoint concepts with common instance

< *pope* Place Person>
< *Dibra* Place Person >
…
< *Xbox_Music* Place Person >

Classifier →

Noisy type assertion detection

Place(*pope*) error
Person(*Dibra*) error
…
Place(*Xbox_Music*) and Person(*Xbox_Music*) error

- **Experiment result of detecting noisy type assertion in DBpedia**

Classifier	Precision	Recall	F1-Measure
J48	93.6%	93.6%	93.6%
J48(boost)	95.6%	95.6%	95.6%

Reasoning with Large Scale Imprecise Knowledge on the Semantic Web

Motivation

- More and more semantic data are published and linked
- Semantic data are inherently imprecise
  - Data extraction may result in imprecision
  - Data linking may result in imprecision
  - Reasoning with large imprecise semantic data
- Schema of the data may also be imprecise
  - Schema induction
  - Ontology enrichment
  - Reasoning with large imprecise ontologies

Background: Fuzzy pD*

Fuzzy Logic

A fuzzy statement is in form of $\phi[n]$

ϕ is a statement

n is called the fuzzy degree ($n \in [0,1]$)

T-norm operator

Lukasiewicz Logic	$a \otimes b = \max(a + b - 1, 0)$
Godel Logic	$a \otimes b = \min(a, b)$
Product Logic	$a \otimes b = \mathrm{a} \cdot b$

Fuzzy RDF triple

(Tom, like, pizza)[0.8]

Background: Fuzzy pD*

Fuzzy D^* rule

E.g. rule f-rdfs2 :

$$(p, \mathrm{domain}, u)[n], (v, p, w)[m] \Rightarrow (v, \mathrm{type}, u)[n \otimes m]$$

Fuzzy P rules

E.g. rule f-rdfsp4

$$(p, \mathrm{type}, \mathrm{TransitiveProperty})[n], (a, p, b)[m], (b, p, c)[k]$$
$$\Rightarrow (a, p, c)[n \otimes m \otimes k]$$

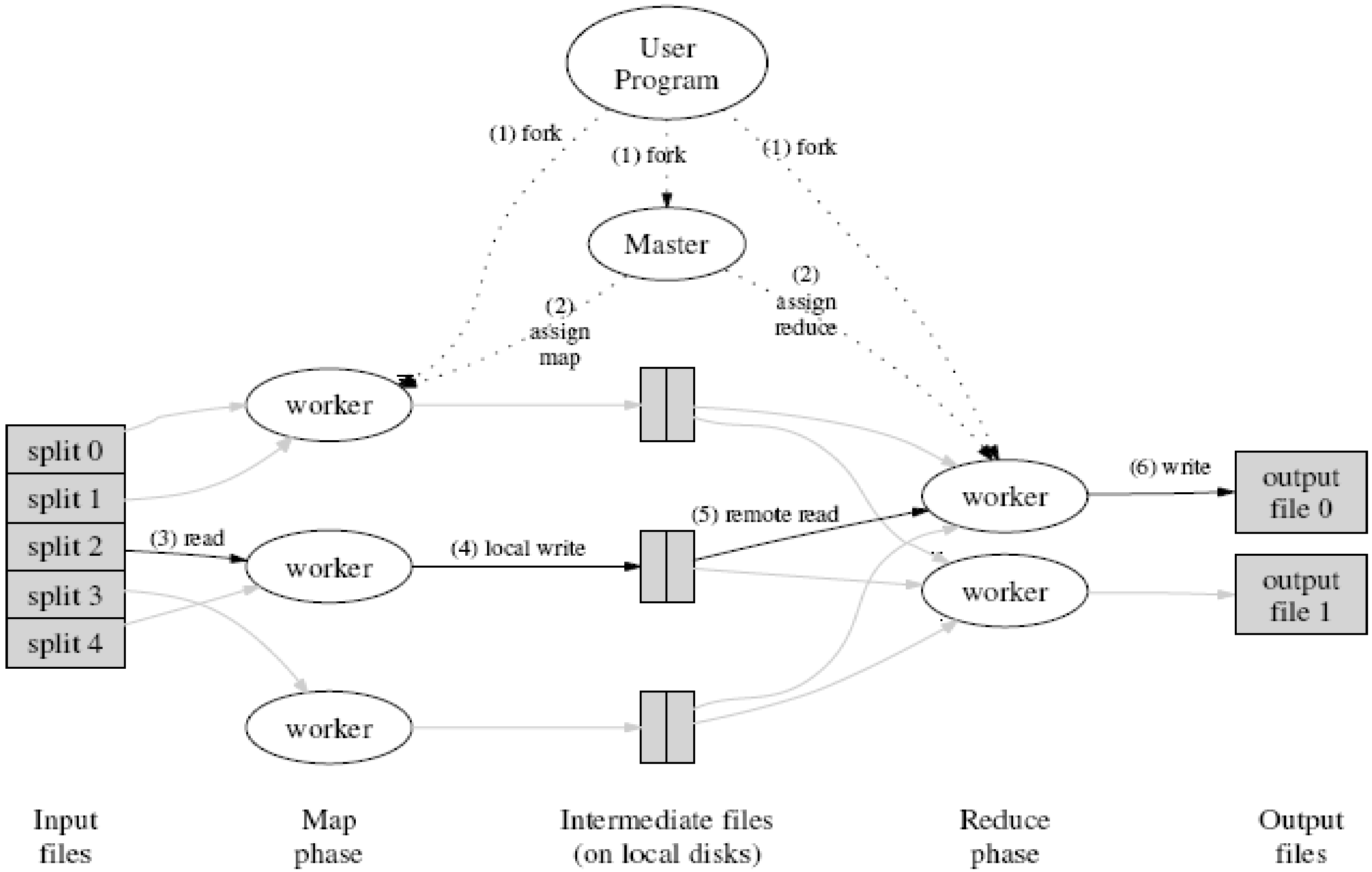
Best Degree Bound

$$(a, \mathrm{type}, u)[0.5], (a, p, b)[0.9], (p, \mathrm{domain}, u)[1]$$

Since $(a, p, b)[0.9], (p, \mathrm{domain}, u)[1] \Rightarrow (a, \mathrm{type}, u)[0.9]$

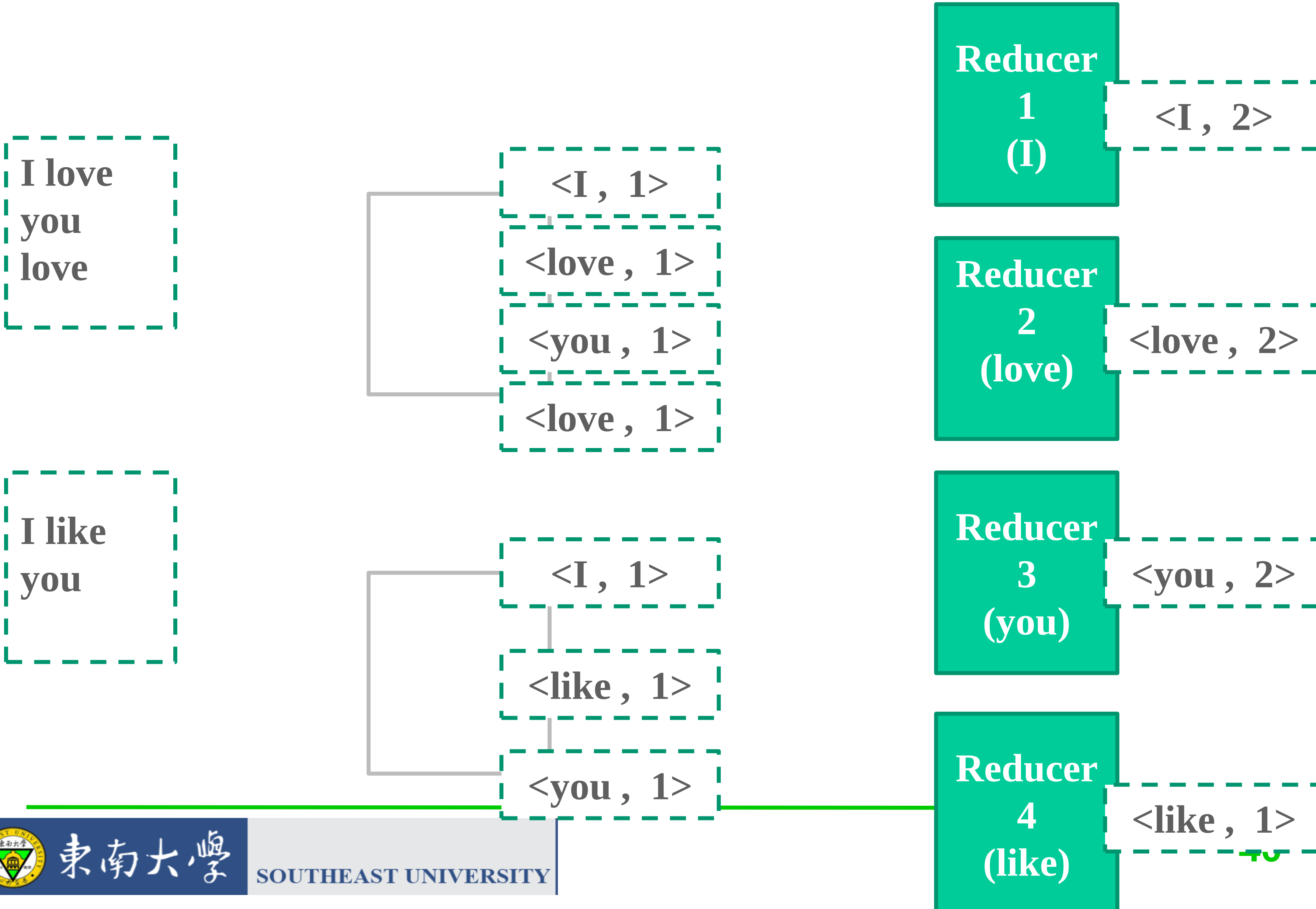
The BDB of (a, type, u) is 0.9

MapReduce



User Program
(1) fork
(1) fork
(1) fork
Master
(2) assign map
(2) assign reduce
worker
worker
worker
split 0
split 1
split 2
split 3
split 4
(3) read
(4) local write
(5) remote read
worker
worker
(6) write
output file 0
output file 1
Input files
Map phase
Intermediate files (on local disks)
Reduce phase
Output files

Example


I love
you
love
I like
you
<I , 1>
<love , 1>
<you , 1>
<love , 1>
<I , 1>
<like , 1>
<you , 1>
Reducer 1 (I)
<I , 2>
Reducer 2 (love)
<love , 2>
Reducer 3 (you)
<you , 2>
Reducer 4 (like)
<like , 1>

Challenges

Ordering the rule applications

Bad orders will generate more non-BDB fuzzy triples

The shortest path calculation

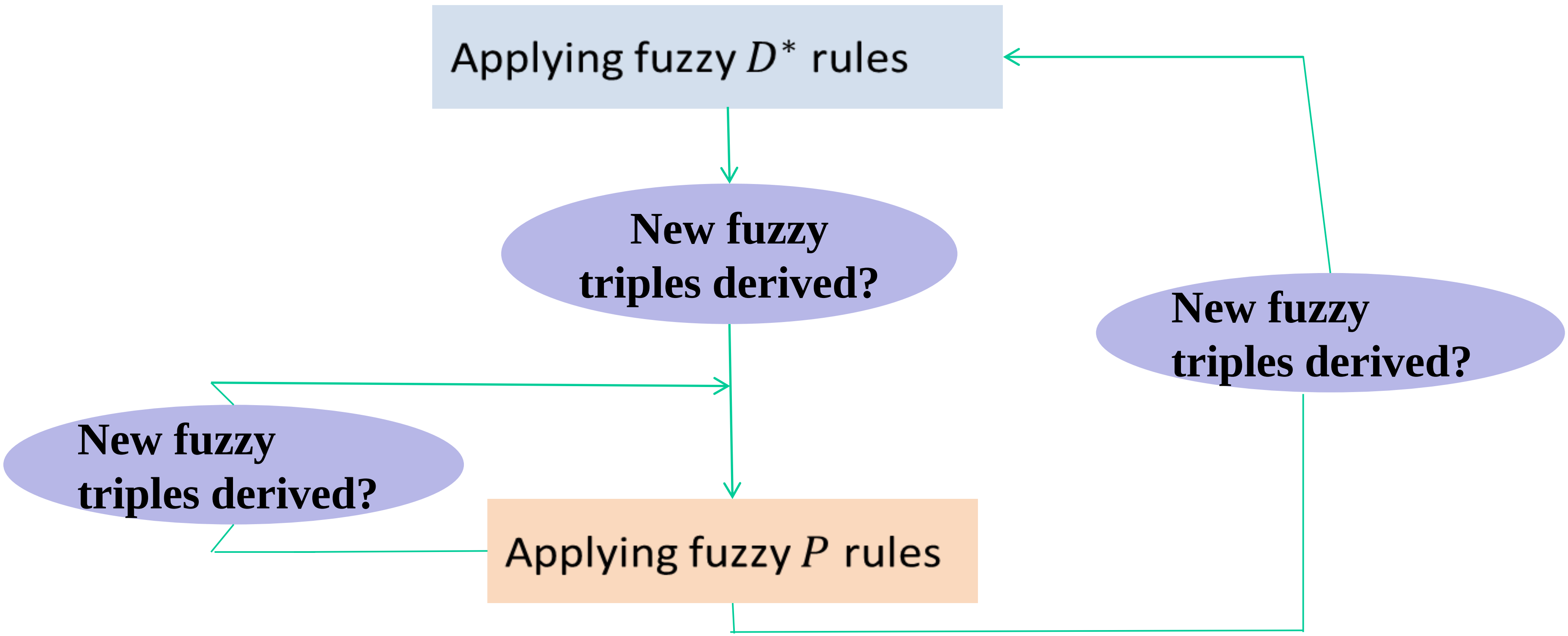
Some rules essentially calculates the all-pair shortest paths

Sameas rules

Canonical representation technique is not applicable to handle the semantics of vague sameas triples

Ordering the rule applications

- Control flow of the reasoning algorithms

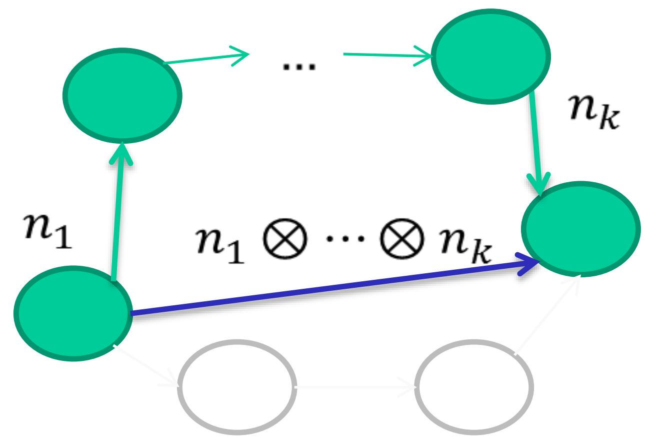


Shortest path calculation

- Some rules are essentially calculating the shortest path between instances in the fuzzy RDF graph

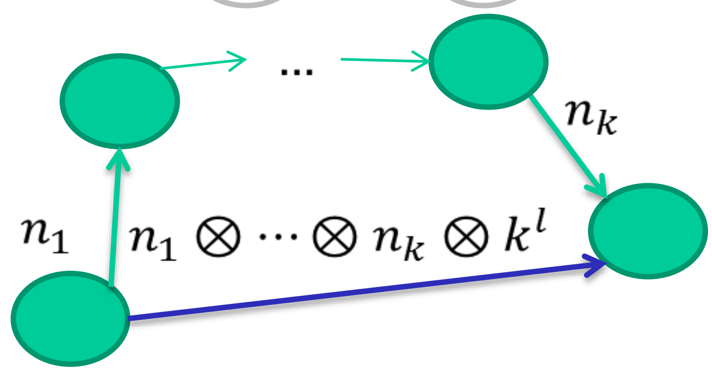
- Class and property hierarchy rules

  E.g. rule f-rdfs11,
  $(u, \text{subClassOf}, v)[n], (v, \text{subClassOf}, w)[m] \Rightarrow (u, \text{subClassOf}, w)[n \otimes m]$

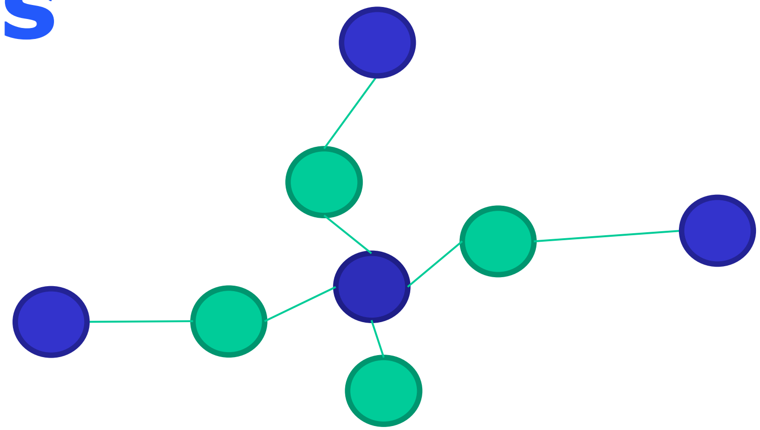


- Transitive property rules

  Rule f-rdfp4,
  $(p, \text{type}, \text{TransitiveProperty})[l], (a, p, b)[n], (b, p, c)[m] \Rightarrow (a, p, c)[n \otimes m \otimes l]$



Sameas rules



Traditional Method

Canonical representation

Drawback

Vague sameas triples

$(a, \text{sameas}, b)[0.8]$ $(b, \text{sameas}, c)[0.1]$ $(c, \text{sameas}, d)[0.8]$

$(a, \text{range}, r)[0.9]$ $(u, b, v)[0.9]$ $(c, \text{domain}, e)[1]$ $(u', d, v')[0.9]$

There is no canonical representation!

If we choose c as the representation

the RDF graph will be converted into

$(c, \text{range}, r)[0.1]$ $(u, c, v)[0.1]$ $(c, \text{domain}, e)[1]$ $(u', c, v')[0.8]$

The BDB of (v, type, r) is 0.1

However the BDB of (v, type, r) in the original graph is 0.8

Setting

- **Dataset**
  - Weighted DBPedia core ontology
  - wpdLUBM 1000, 2000, 4000, 8000

- **Cluster**
  - 25 machine with at most 75 mapper/reducer slots

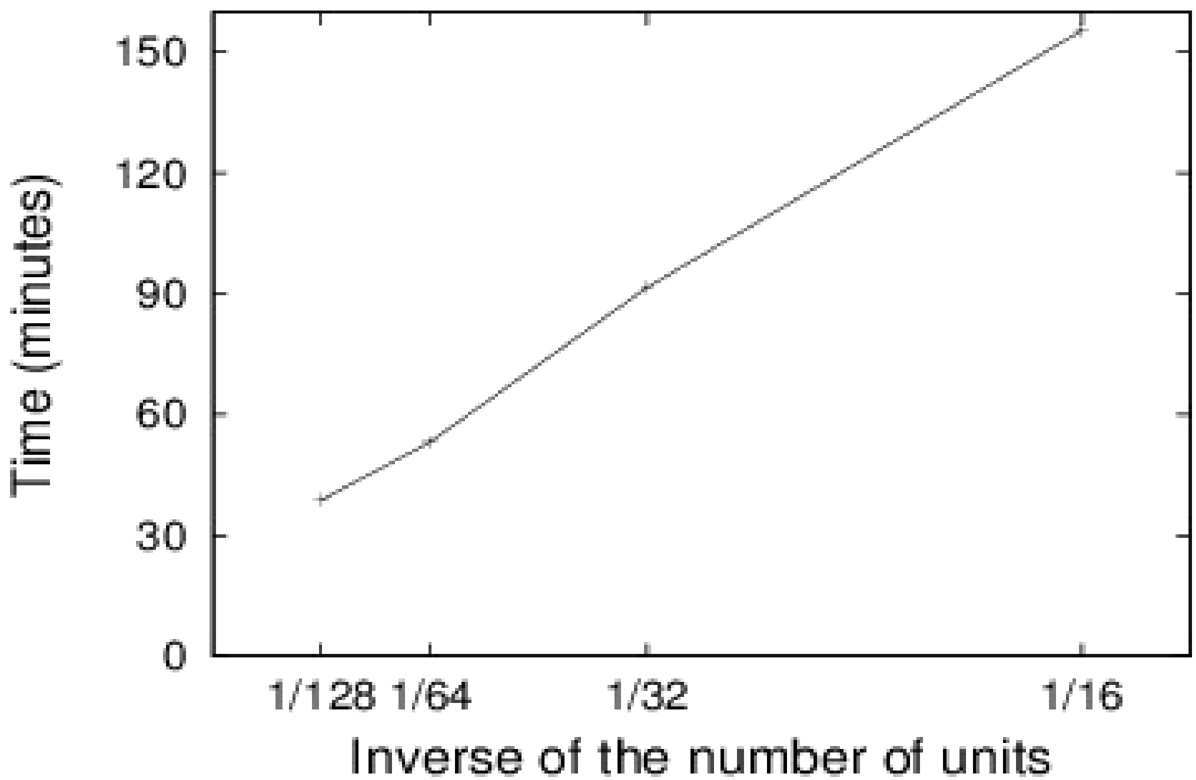
Experiment

- Dataset: Weighted DBPedia core ontology
- Results:

#units	128	64	32	16	8	4	2
Time(sec.)	122.653	136.861	146.393	170.859	282.802	446.917	822.269
Speedup	6.70	6.01	5.62	4.81	2.91	1.84	1.00

Experiment

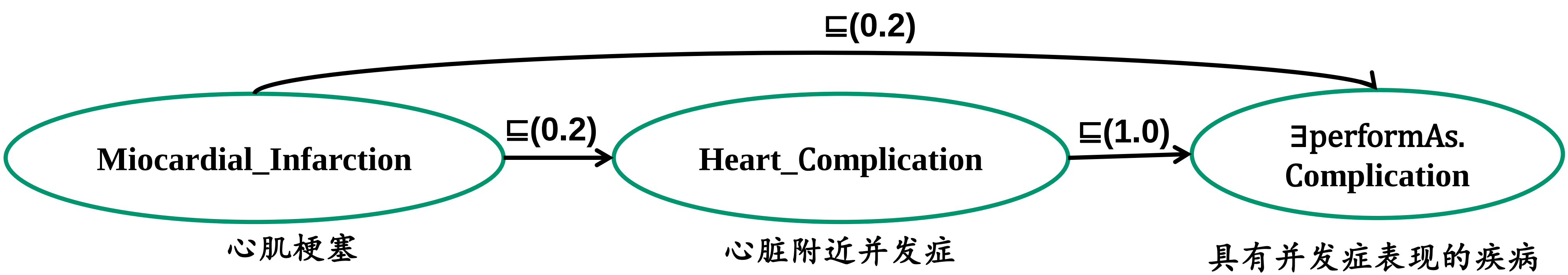
❑ **Scalability over number of units**



Reasoning on fuzzy-EL+

- Classfication rules for fuzzy-EL+

R2	If <*A*, *n*> ∈ ***S(X)***, <*A* ⊑ ∃*r*.*B*, *k*> ∈ O, and <***X*, *B*, *m***> ∉ ***R(r)***, where ***m*** = min (***n***, ***k***) then ***R(r)* : = *R(r)*** ∪ {<***X*, *B*, *m***>}, where ***m*** = min (***n***, ***k***)



Challenges and methods

- **Transforming rule-applying to an operation on tables**
- **The rules are given by operations on sets. It is more straightforward to treat them as operations on tables, in other words, relation algebra**

R2:

$\langle A, n\rangle \in S(X)$, $\langle A \sqsubseteq \exists r.B, k\rangle \in O \Rightarrow \langle X, B, m\rangle \in R(r)$

R2:

$(X, A, n) \in S$, $(A, r, B, k) \in O_{\sqsubseteq \exists} \Rightarrow (r, X, B, m) \in R$

$$(S \bowtie_A O_{\sqsubseteq \exists}) \cup R$$

Challenges and methods

- **Handling multi-way join**
- **MapReduce can handle a 2-way join in one job.**

R2 ($(\boldsymbol{S} \bowtie_{A} \mathbf{O}_{\sqsubseteq\exists}) \cup \boldsymbol{R}$) and R4, contain one 2-way join. They can be easily handled by MapReduce.

R1 contains a complex multi-way join.

R1: $(\boldsymbol{S} \bowtie_{X} \ldots \bowtie_{X} \boldsymbol{S} \bowtie_{A_1 \ldots A_l} \mathrm{O}_{\sqcap\sqsubseteq}) \cup \boldsymbol{S}$

R3 and R5 contains a 3-way join.

R3: $(\boldsymbol{R} \bowtie_{Y} \boldsymbol{S} \bowtie_{A} \mathrm{O}_{\exists\sqsubseteq}) \cup \boldsymbol{S}$

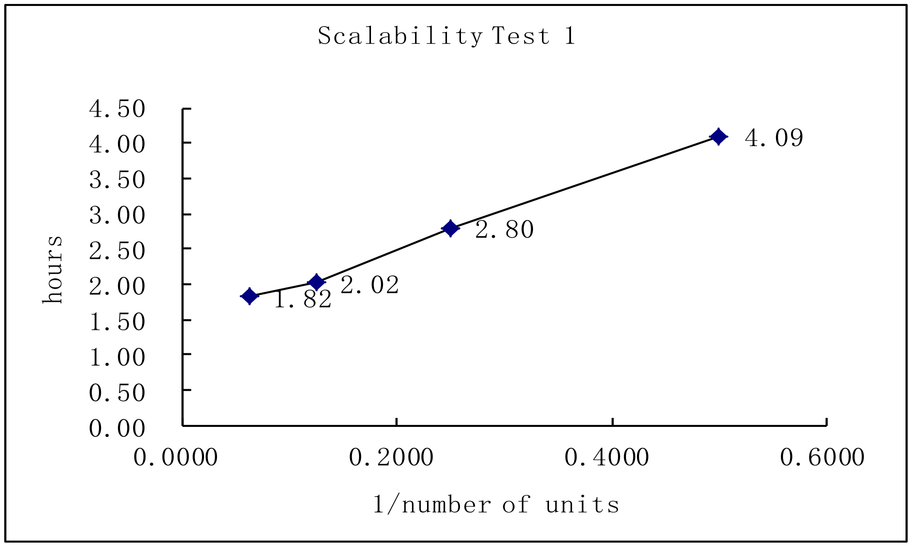
R5: $(\boldsymbol{R} \bowtie_{Y} \boldsymbol{R} \bowtie_{r,s} \mathrm{O}_{\circ\sqsubseteq}) \cup \boldsymbol{R}$

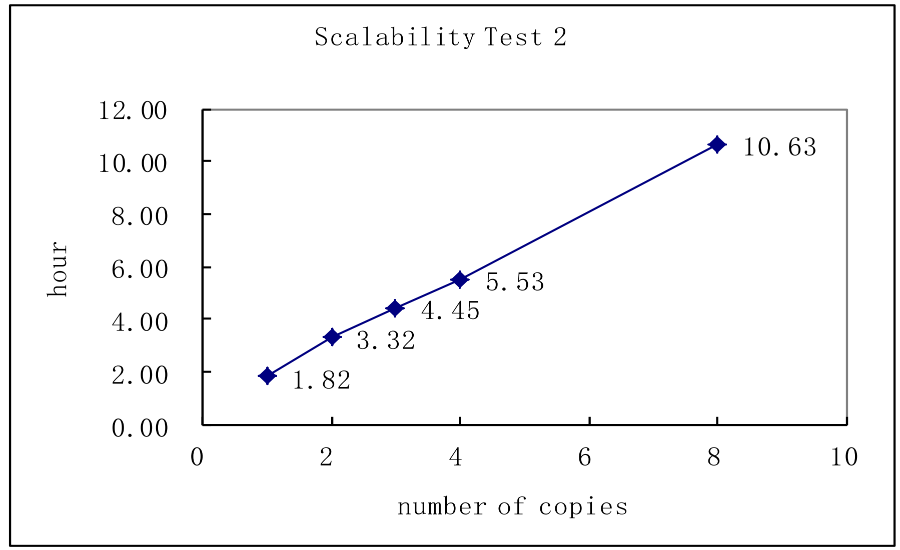
Basic idea: Transforming a 3-way join to two 2-way joins.

Evaluation on Galen

- EL-Galen
- 8nodes, 16 units

Concepts	Roles	GCI	RI
47,840	1,892	87,594	1,947



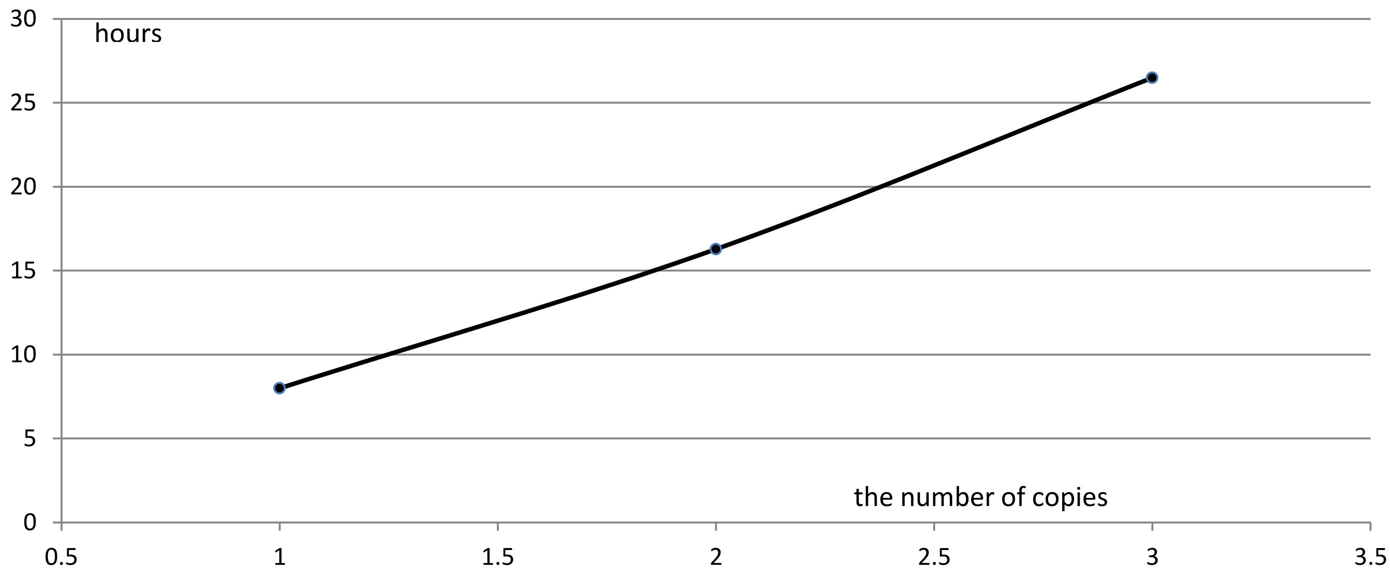


- Scalability test 1: The system's performance speeds up linearly to the increase of number of units
- Scalability test 2: The cost time increases linearly to the number of copies of the input ontology

Evaluation on Snomed-CT

- **Snomed-CT**

Concepts	Roles	GCI	RI
836,612	77	1,438,948	23



- **Scalability test： The cost time increases linearly to the number of copies of the Snomed-CT**

Conclusion and Discussion

- **Ontology reasoning plays an important role for KG**
  - **Detecting logical inconsistency and repair knowledge bases (improve the quality of the knowledge)**
  - **Extend knowledge bases（materialization, classification)**
  - **Query rewritting and extension**
- **Reasoning in KG need meta-reasoning**
  - **Variety of knowledge: terminological knowledge, rule, probabilistic knowledge ect.**
- **Challenging problems**
  - **Current work is mainly based on MapReduce and Hadoop, thus suffers from the problem of efficiency and dynamics**
  - **Only lightweight ontology languages, such as RDFS, OWL 2 RL are supported**
  - **Reasoning with imprecise knowledge is not well discussed**