Debugging ontologies and mappings in ontology networks
-
Tutorial at ISWC 2011

Patrick Lambrix, Guilin Qi, Christian Meilicke

Outline

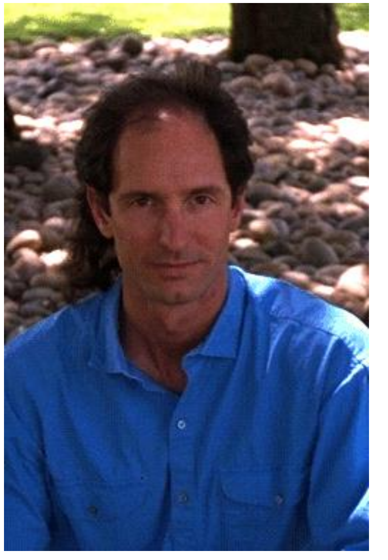
- Overview of defects in ontologies and ontology networks (30mins - all)
- Debugging semantic defects in ontologies (60mins – Guilin Qi)
  - including QA (10mins)
- Debugging ontology mappings (60mins – Christian Meilicke)
- Debugging missing is-a structure (60mins – Patrick Lambrix)

Overview of defects in ontologies and ontology networks

What is an ontology?

- It has different definitions in different domains
- In Semantic Web, a popular definition is:

An ontology is an explicit specification of a conceptualization



Gruber, 1993

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

Description Logics

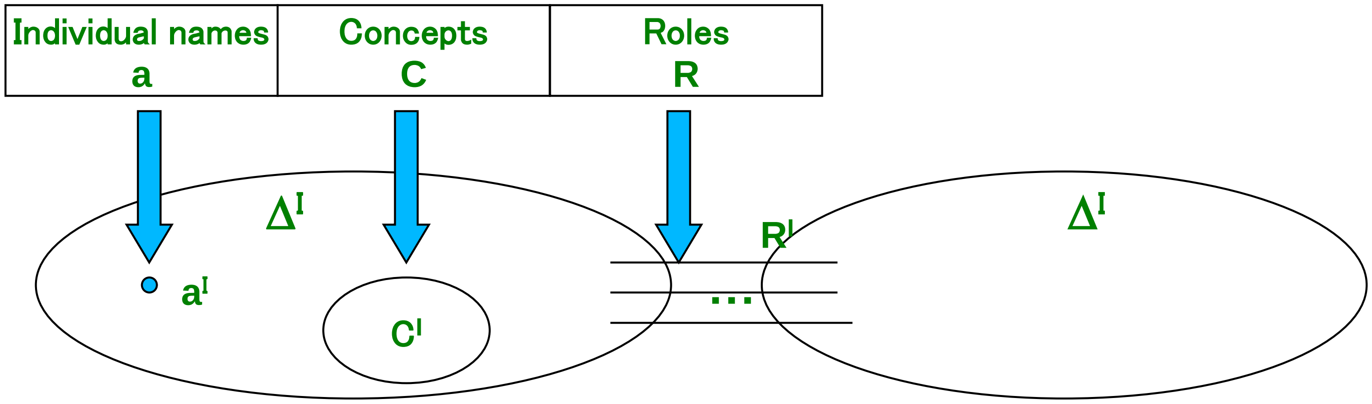
- Description logics
  - Are (mostly) decidable fragments of first-order predicate logic
  - Provide logical underpinning of W3C standard OWL
- Building blocks
  - Concepts (unary predicates/formulae with one free variable)
    - E.g., Person, Lawer $\sqcup$ Doctor
  - Roles (binary predicates/formulae 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^{-}

Description Logics (Semantics)

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



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 T: defining terminology of application domain
  - Inclusion assertion on concept :$C \sqsubseteq D$

    Pericardium $\sqsubseteq$ Tissue $\sqcap$ $\exists$ part-of.Heart

  - Inclusion assertion on roles: $R \sqsubseteq S$

    Part-of $\sqsubseteq$ has-location

- ABox 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 $R^I \subseteq D^I$
- 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=<T, A>
  - 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:

$$\mathcal{O} \models \sigma \Leftrightarrow: \mathcal{I} \models \sigma \text{ for all models } \mathcal{I} \text{ of } \mathcal{O}$$

$$\mathcal{O} \models \mathsf{Unsat}(C) \Leftrightarrow: \mathcal{O} \models C \sqsubseteq \bot$$

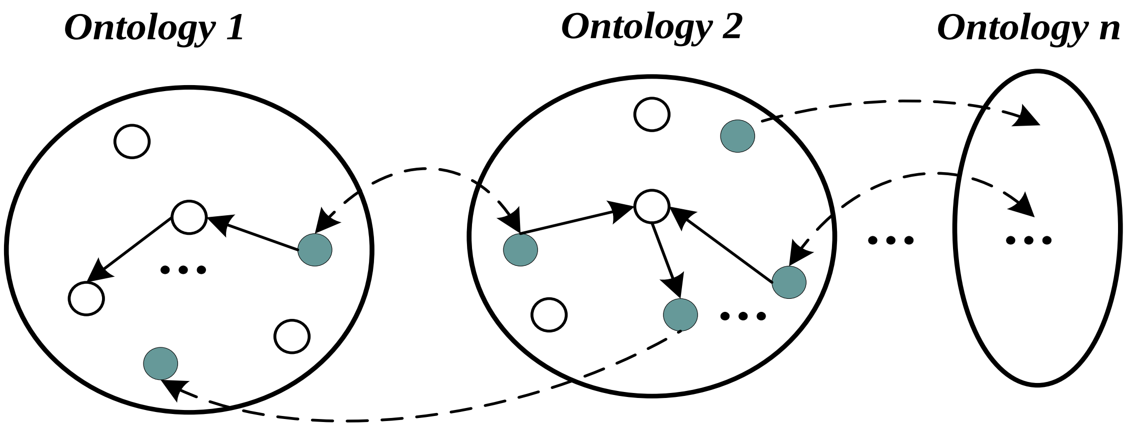
Description Logics(Semantics)

- Incoherent ontology: ontology with at least one unsatisfiable concept
  - Example: {PhDStudent $\sqsubseteq$ Student,
    PhDStudent $\sqsubseteq$ Employee,
    Student $\sqsubseteq \neg$Employee}
- Inconsistent ontology: ontology without a model
  - Example: {PhDStudent $\sqsubseteq$ Student,
    PhDStudent $\sqsubseteq$ Employee,
    Student $\sqsubseteq \neg$Employee,
    PhDStudent(John)}

Incoherent ontology can be consistent!

Ontology networks

An **ontology network** consists of a set of **ontologies** and sets of **mappings** between those ontologies.



Defects in ontologies and ontology networks

- Neither developing ontologies nor finding mappings between ontologies is an easy task.

- It may happen that
  - ontologies are not correct/complete
  - mappings between ontologies are not correct/complete
  - the integrated ontology network is not consistent

Defects in ontology networks

- ontologies are not correct/complete
  - Ontology debugging
  - Ontology learning
- mappings between ontologies are not correct/complete
  - Ontology alignment
  - Debugging mappings
- the integrated ontology network is not consistent
  - Ontology network debugging

Defects in ontologies

- Syntactic defects
  - eg. wrong tags or incorrect format
- Semantic defects
  - eg. unsatisfiable concepts, inconsistent ontologies
- Modeling defects
  - eg. wrong or missing relations

Defects in ontologies and ontology networks

- Ontologies and ontology networks with defects, although often useful, also lead to problems when used in semantically-enabled applications.

→ Wrong conclusions may be derived or valid conclusions may be missed.

Example: Incoherent Ontology

- Example: DICE ontology
  - Brain $\sqsubseteq$ CentralNervousSystem $\sqcap$ $\exists$systempart.NervousSystem $\sqcap$ BodyPart $\sqcap$ $\exists$ region.HeadAndNeck $\sqcap$ $\forall$region.HeadAndNeck
  - CentralNervousSystem $\sqsubseteq$ NervousSystem
  - BodyPart $\sqsubseteq \neg$NervousSystem or DisjointWith(BodyPart,NervousSystem)

Example: Inconsistent Ontology

- **Example from Foaf:**
  - **Person(timbl)**
  - **Homepage(timbl, http://w3.org/)**
  - **Homepage(w3c, http://w3.org/)**
  - **Organization(w3c)**
  - **InverseFunctionalProperty(Homepage)**
  - **DisjointWith(Organization, Person)**
- **Example from OpenCyc:**
  - **ArtifactualFeatureType(PopulatedPlace)**
  - **ExistingStuffType(PopulatedPlace)**
  - **DisjointWith(ExistingObjectType,ExistingStuffType)**
  - **ArtifactualFeatureType $\sqsubseteq$ ExistingObjectType**

Example - missing is-a relations

- In 2008 Ontology Alignment Evaluation Initiative (OAEI) Anatomy track, task 4
  - Ontology MA : Adult Mouse Anatomy Dictionary (2744 concepts)
  - Ontology NCI-A : NCI Thesaurus - anatomy (3304 concepts)
  - 988 mappings between MA and NCI-A
    - 121 missing is-a relations in MA
    - 83 missing is-a relations in NCI-A

Influence of missing structure

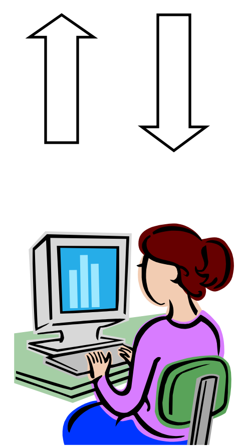
- Ontology-based querying.

Medical Subject Headings (MeSH)

All MeSH Categories

- **Diseases Category**
  - **Eye Diseases**
    - **Scleral Diseases**
      - **Scleritis**

...



return 1363 articles

Influence of missing structure

- Incomplete results from ontology-based queries

Medical Subject Headings (MeSH)

All MeSH Categories

- **Diseases Category**
  - **Eye Diseases**
    - **Scleral Diseases**
      - **Scleritis**

...

return 1363 articles

return 613 articles

55% results are missed !

Example mappings - OAEI Results 2008-2010

- Matching systems generate highly incoherent mappings
  - Up to 50% of all generated correspondences have to be removed until a coherent subset can be found
  - No system generated fully coherent mappings in the past
  - Some systems like ASMOV and Codi have been the exception (nearly coherent mappings) and LogMap in 2011

- Mapping coherence becomes more important!

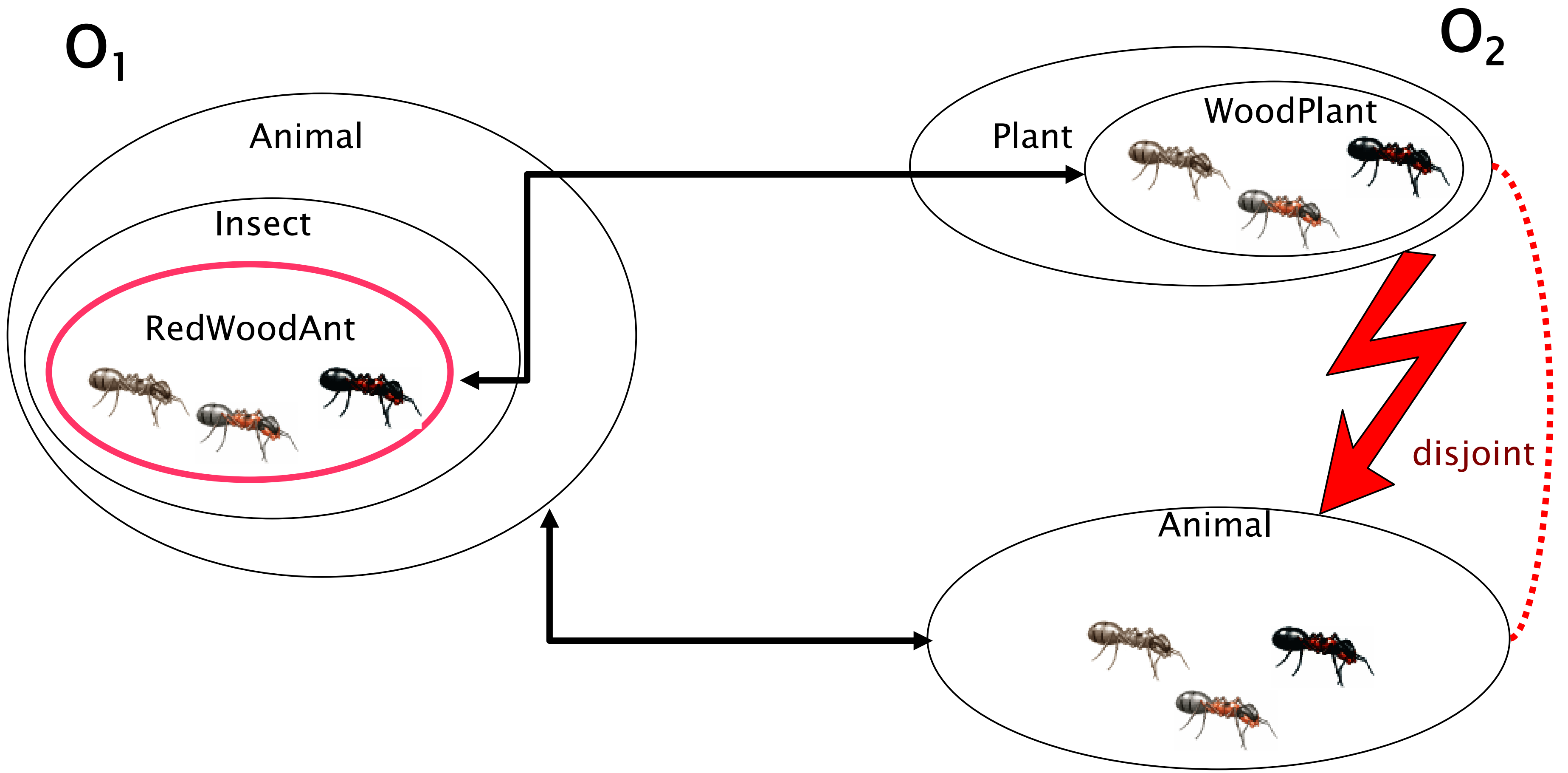
Example: Incoherent Mapping

- String-based matching techniques generates:
- 1#Animal = 2#Animal
  - 1#RedWoodAnt $\sqsubseteq$ 2#WoodPlant
- In ontology #1:
  - RedWoodAnt $\sqsubseteq$ Insect $\sqsubseteq$ Animal
- In ontology #2:
  - Animal $\sqsubseteq$ $\neg$Plant
  - WoodPlant $\sqsubseteq$ Plant

1#RedWoodAnt unsatisfiable!

Why?

Instance migration



O_2 is **inconsistent** after instance migration!

Debugging semantic defects in ontologies

Outline

- Justification for Debugging Ontologies
- Methods for Finding Justifications
- A Scalable Method for Debugging Large Inconsistent Ontologies
- Conclusion

Outline

- Justification for Debugging Ontologies
- Methods for Finding Justifications
- A Scalable Method for Debugging Large Inconsistent Ontologies
- Conclusion

What is a Justification?

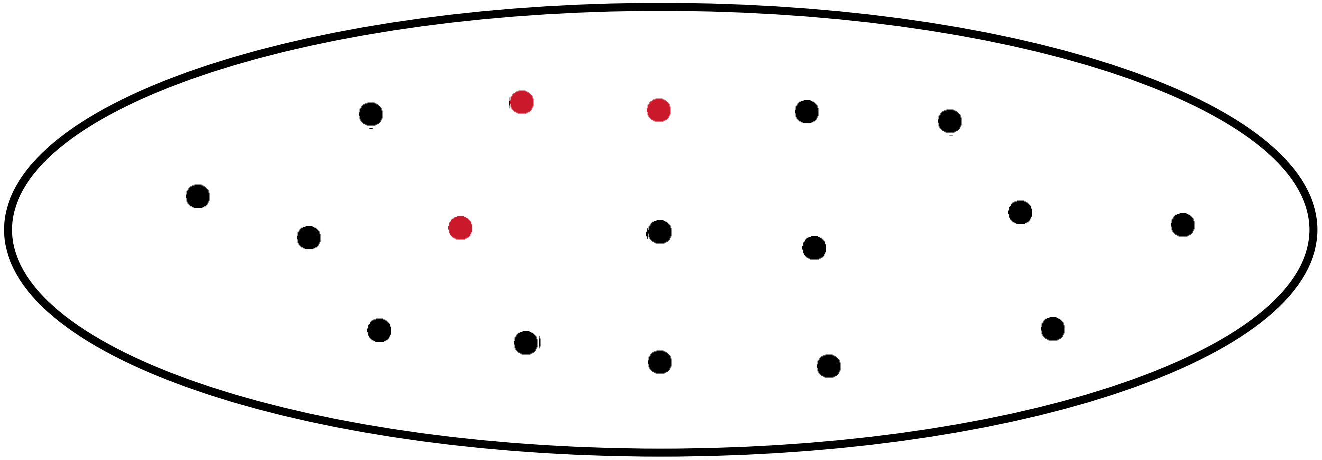
- Justification

Let $\mathcal{O} \models \sigma$. $J \subseteq \mathcal{O}$ is a justification for σ in $\mathcal{O}$ $\Leftrightarrow$:

- $J \models \sigma$

- for every $J' \subset J$, $J' \not\models \sigma$

minimal set of axioms having the entailment



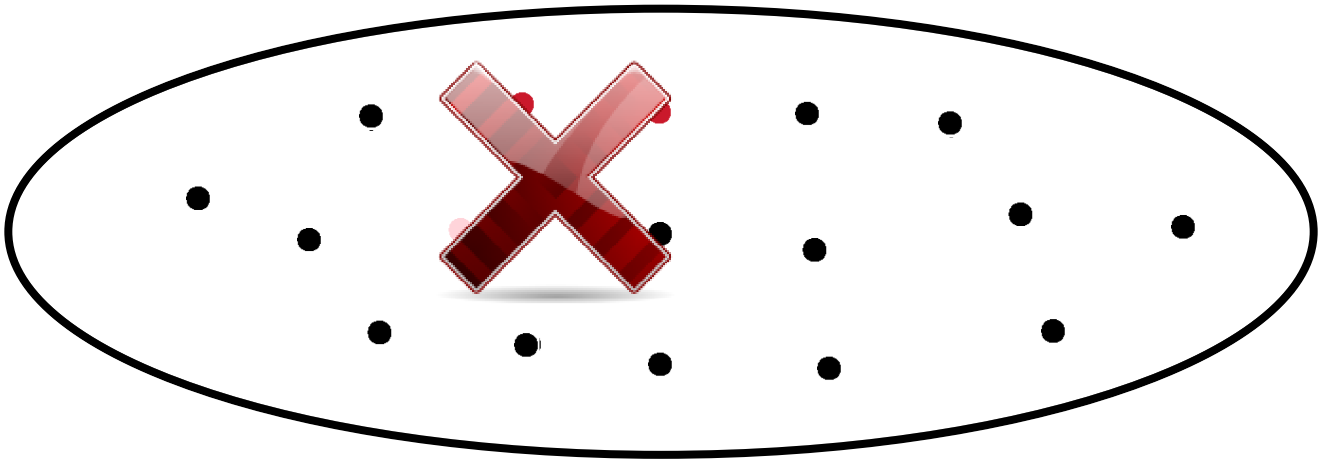
What is a Justification?

- Justification

Let $\mathcal{O} \models \sigma$. $J \subseteq \mathcal{O}$ is a justification for σ in $\mathcal{O}$ $\Leftrightarrow$:

- $J \models \sigma$
- for every $J' \subset J$, $J' \not\models \sigma$

minimal set of axioms having the entailment



Debugging Inconsistent Ontology

- Minimal Inconsistent Subset (MIS) O' of O:
  - O' is inconsistent (inconsistency)
  - O'' is consistent for $O'' \subset O'$ (minimalism)
- MIS and justifications
  - A MIS is a special justification
    - O' is a MIS of O iff O' is a justification for $\top \sqsubseteq \bot$

Debugging Incoherent Ontology

- Minimal Unsatisfiability Preserving Subset (MUPS) T' for A w.r.t. T: $T' \subseteq T$
  - A is unsatisfiable in T' (unsatisfiability)
  - A is satisfiable in any T'' where $T'' \subset T'$ (minimalism)
- MUPS and justifications
  - A MUPS is a special justification
  - Computing justification can be reduced to computing MUPS in OWL DL
    - $O \models C \sqsubseteq D$ iff $O \models C \sqcap \neg D \sqsubseteq \bot$

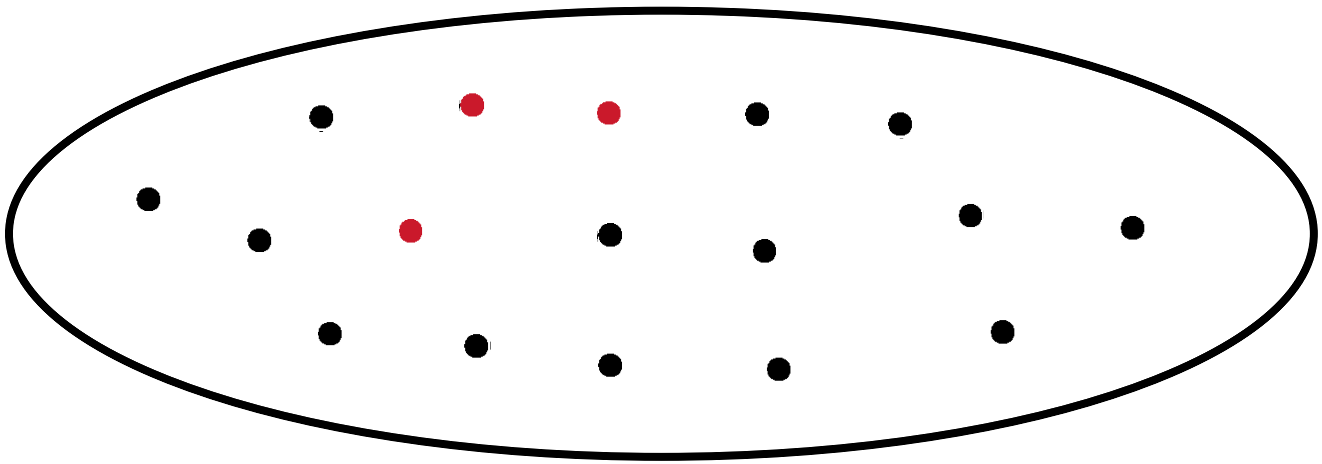
Outline

- Justification for Debugging Ontologies
- Methods for Finding Justifications
- A Scalable Method for Debugging Large Inconsistent Ontologies
- Conclusion

Finding One Justification

- Tableau-based approach
  - Is based on tableau algorithm for DLs
  - Apply tracing techniques
- Black-box based approach
  - Takes a DL reasoner as an oracle
  - Is easy to implement and still efficient

Compute One Justification



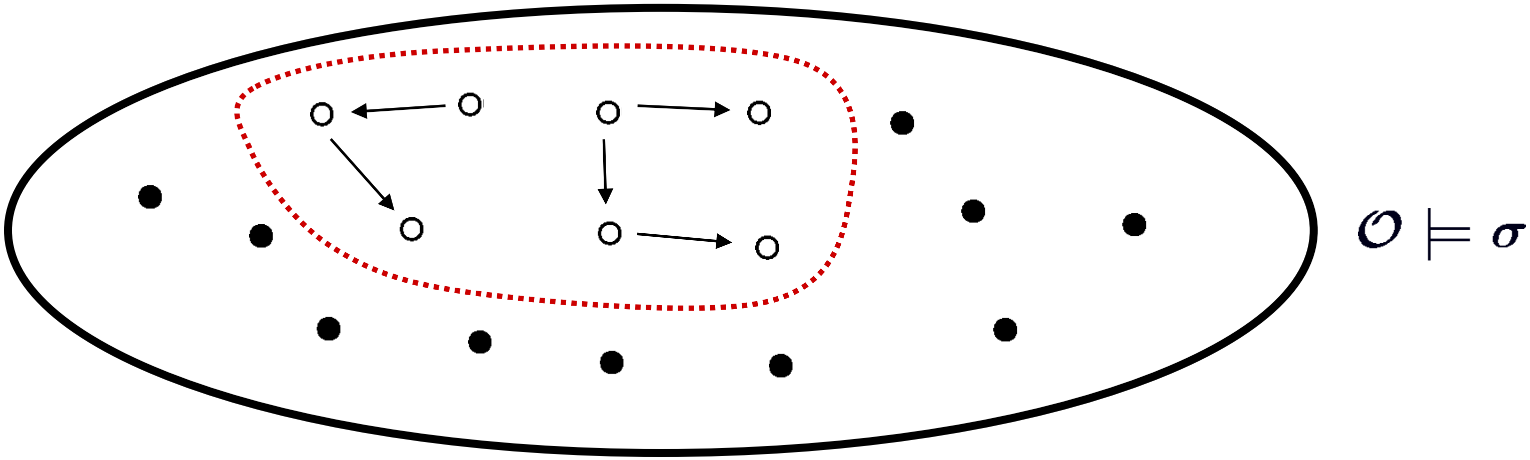
$\mathcal{O} \models \sigma$

- Start with $J \leftarrow \mathcal{O}$
- For each axiom $\alpha \in \mathcal{O}$, if $J \backslash \{\alpha\} \models \sigma$ then $J \leftarrow J \backslash \{\alpha\}$

It requires n subsumption tests with $n = \text{card}(\mathcal{O})$

Works on Finding Justifications

Relevance-based strategy



- Start with $J \leftarrow \{\alpha \in \mathcal{O} \mid \mathsf{Sig}(\alpha) \cap \mathsf{Sig}(\sigma) \neq \emptyset\}$
- Expand J with $\beta \in \mathcal{O} \setminus J$ with $\mathsf{Sig}(\beta) \cap \mathsf{Sig}(J) \neq \emptyset$ until $J \models \sigma$

Works on Finding Justifications

Finding all justifications

- Tableau-based approach
  - Is based on extensions of tableau algorithms
  - Reasoner dependent, hard to implement
- Automata-based approach
  - Automata-based algorithms for reasoning in DLs extended to pinpointing algorithms
- <u>Black-box based approach</u>
  - Reuses existing techniques for diagnosis
    - i.e., Hitting Set Tree algorithm

Works on Finding Justifications

Finding all justifications

Given a justification J for σ in $\mathcal{O}$, there is another J' $\Leftrightarrow$:

$\mathcal{O} \setminus \{\alpha\} \models \sigma$ for some $\alpha \in J$

$J' \subseteq \mathcal{O} \setminus \{\alpha\}$

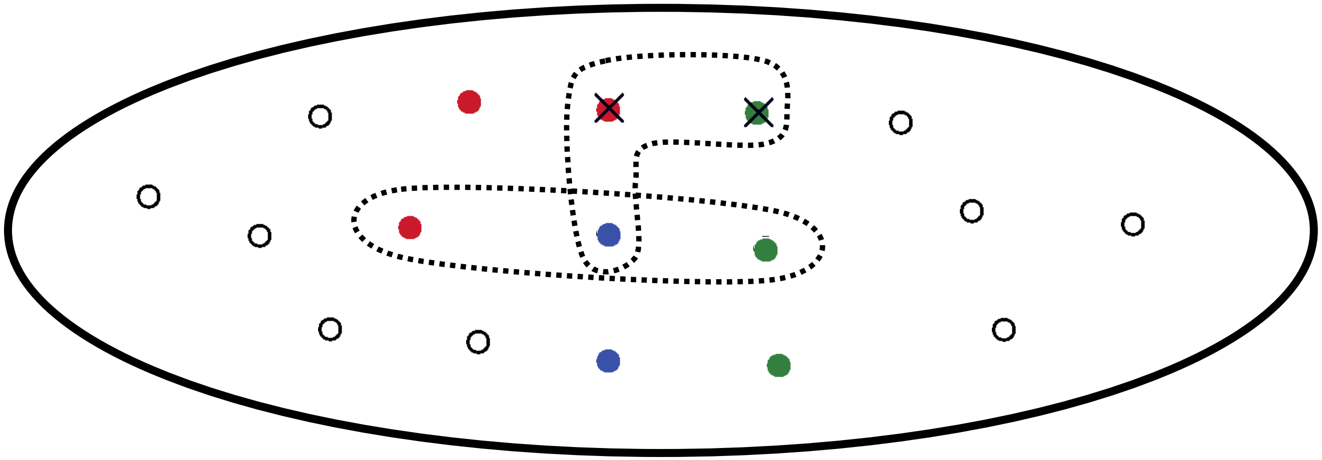
Works on Finding Justifications

Finding all justifications

Given a justification J for σ in $\mathcal{O}$, there is another $J' \Leftrightarrow$:

$\mathcal{O} \setminus \{\alpha\} \models \sigma$ for some $\alpha \in J$

$J' \subseteq \mathcal{O} \setminus \{\alpha\}$



$\mathcal{O} \models \sigma$

Given justifications $J_1, \ldots, J_n$ for σ in $\mathcal{O}$, there is another $J' \Leftrightarrow$:

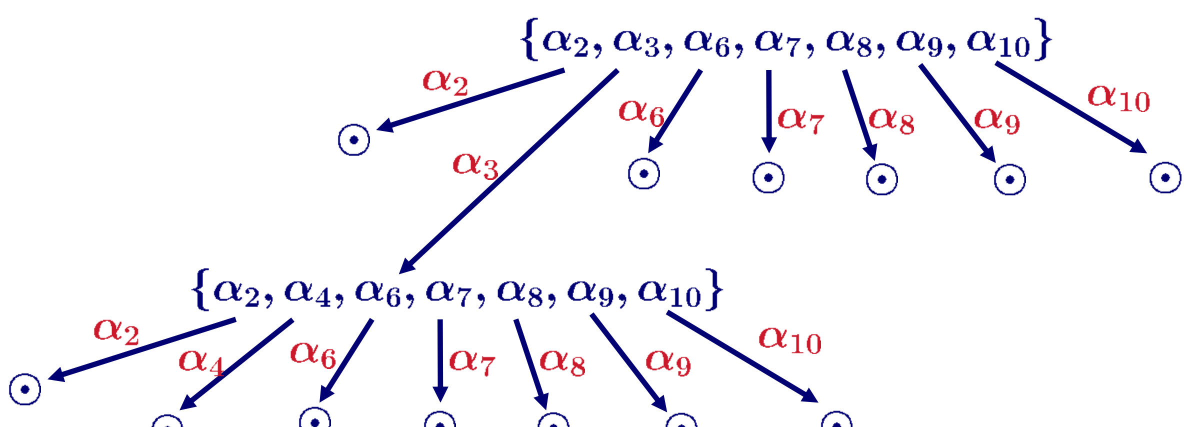
$\mathcal{O} \setminus H \models \sigma$ for some set H s.t. $H \cap J_i \neq \emptyset$

Works on Finding Justifications

Hitting set tree algorithm

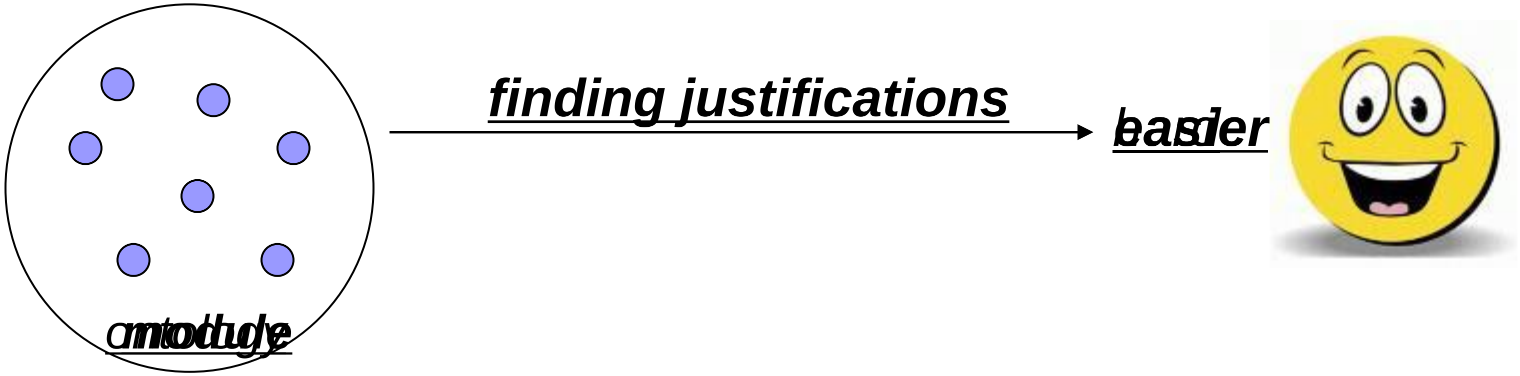
		α_1	Pericardium	$\sqsubseteq$	Tissue $\sqcap$ $\exists$part-of.Heart
✔	✔	α_2	Endocardium	$\sqsubseteq$	Tissue $\sqcap$ $\exists$part-of.HeartValve $\sqcap$ $\exists$part-of.HeartWall
	✔	α_3	HeartValve	$\sqsubseteq$	BodyValve $\sqcap$ $\exists$part-of.Heart
✔		α_4	HeartWall	$\sqsubseteq$	BodyWall $\sqcap$ $\exists$part-of.Heart
		α_5	Pericarditis	$\equiv$	Inflammation $\sqcap$ $\exists$has-location.Pericardium
✔	✔	α_6	Endocarditis	$\equiv$	Inflammation $\sqcap$ $\exists$has-location.Endocardium
✔	✔	α_7	Inflammation	$\sqsubseteq$	Disease $\sqcap$ $\exists$acts-on.Tissue
✔	✔	α_8	Disease $\sqcap$ $\exists$has-location.Heart	$\sqsubseteq$	HeartDisease
✔	✔	α_9	part-of	$\sqsubseteq$	has-location
✔	✔	α_{10}	Trans(has-location)		

$$\mathcal{O}_{\text{med}} \models \sigma = (\text{Endocarditis} \sqsubseteq \text{HeartDisease})$$

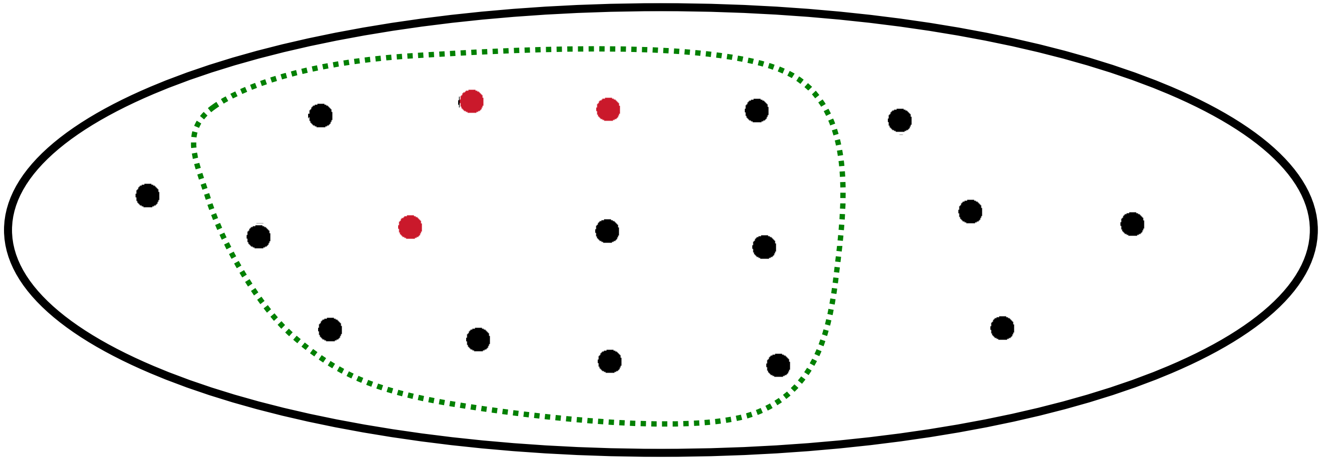


Challenging Problems

- Fine-grained justifications
  - Irrelevant parts of an axiom
  - Example: $\{B \sqsubseteq C \sqcap D,\ D \sqsubseteq E\} \models B \sqsubseteq E$
- Scalability
  - NP-hard even for tractable DLs



Modularization-based Strategy



- Start with $J \leftarrow \mathcal{O}'$ with $\mathcal{O}'$ a σ-module in $\mathcal{O}$

$\mathcal{O} \models \sigma \Leftrightarrow \mathcal{O}' \models \sigma$

Apply Hitting Set Three algorithm to the module

Reachability-based modules (S,08) and **Locality**-based modules (Grau et al.,07)

Module extraction is purely syntactic; 1st phase is cheap

The modules are reasonably small

Modularization-based Strategy

Reachability-based modules (S,08) and **Locality**-based modules (Grau et al.,07)

Module extraction is purely syntactic; 1st phase is cheap

The modules are reasonably small

Theorem: The (minimal) locality-based module for concept {A} in a SHOIQ ontology O contains all the relevant axioms for any subsumption $\sigma=\{A \sqsubseteq B\}$

Experiments

Three algorithms have been implemented using KAON2 as the subsumption reasoner:

1. ALL_JUSTS (Kalyanpur et al.,07)
2. REL_ALL_JUSTS (Ji et al.,08)
3. MODULE_ALL_JUSTS

Ontologies	♯Axioms	♯Concepts	♯Roles	Module size		Extraction time (sec)
				Average	Maximum	
GALEN	4 529	2 748	413	75	530	6
GO	28 897	20 465	1	16	125	40
NCI	46 940	27 652	70	29	436	65

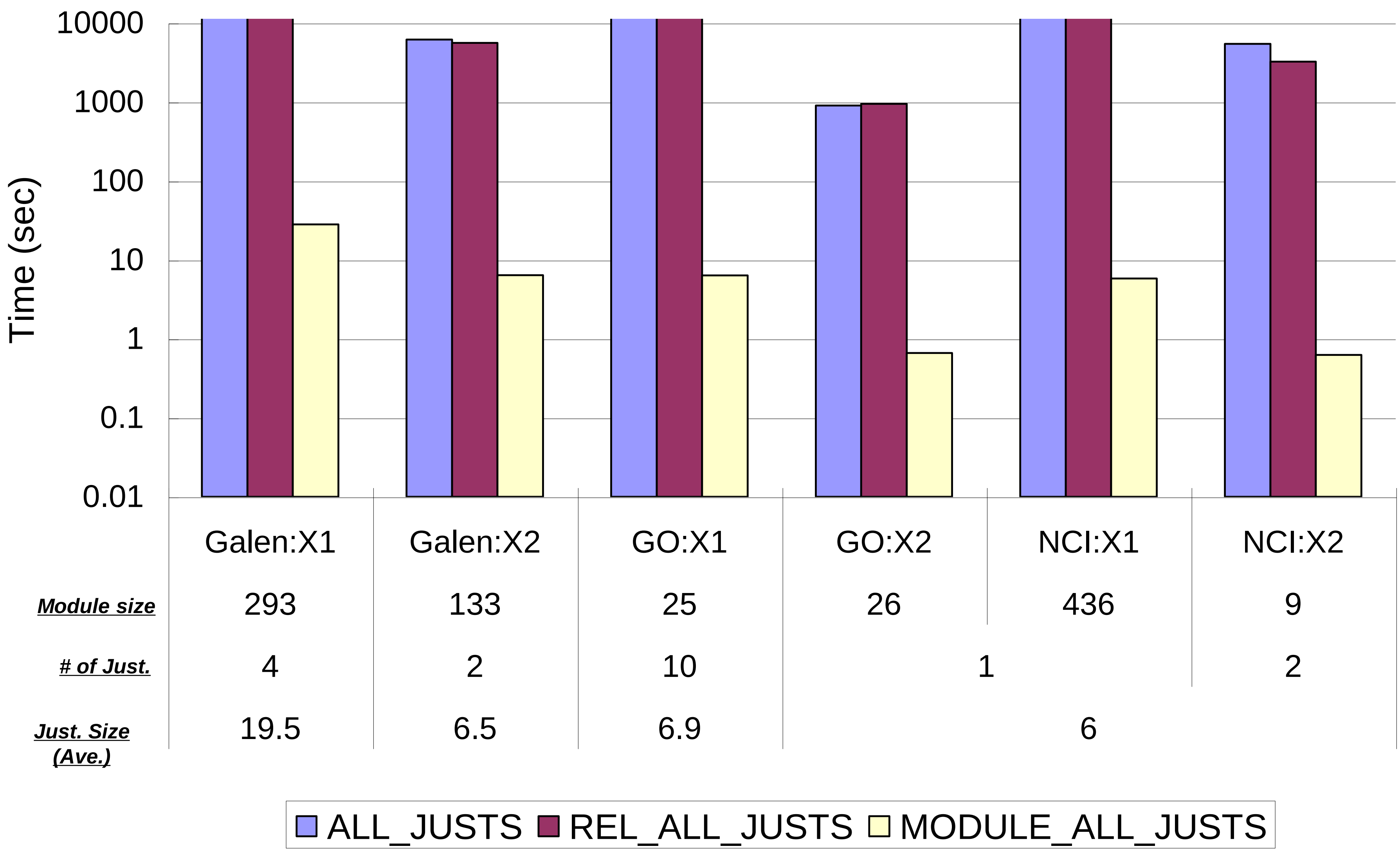
Experiments (Cont.)

Performance comparison of the three algorithms:

Randomly select σ_1 and σ_2 from subs($\mathcal{O}$)

GALEN:σ_1	AcuteErosionOfStomach	$\sqsubseteq$	GastricPathology
GALEN:σ_2	AppendicularArtery	$\sqsubseteq$	PhysicalStructure
GO:σ_1	GO_0000024	$\sqsubseteq$	GO_0007582
GO:σ_2	GO_0000027	$\sqsubseteq$	GO_0044238
NCI:σ_1	CD97_Antigen	$\sqsubseteq$	Protein
NCI:σ_2	APC_8024	$\sqsubseteq$	Drugs_and_Chemicals

Results


Time (sec)
10000
1000
100
10
1
0.1
0.01
Galen:X1
Galen:X2
GO:X1
GO:X2
NCI:X1
NCI:X2
Module size
293
133
25
26
436
9
# of Just.
4
2
10
1
2
Just. Size (Ave.)
19.5
6.5
6.9
6
ALL_JUSTS
REL_ALL_JUSTS
MODULE_ALL_JUSTS

Problem

- Problem 1: not goal-directed, i.e., independent of the super-concept in a given concept subsumption entailment

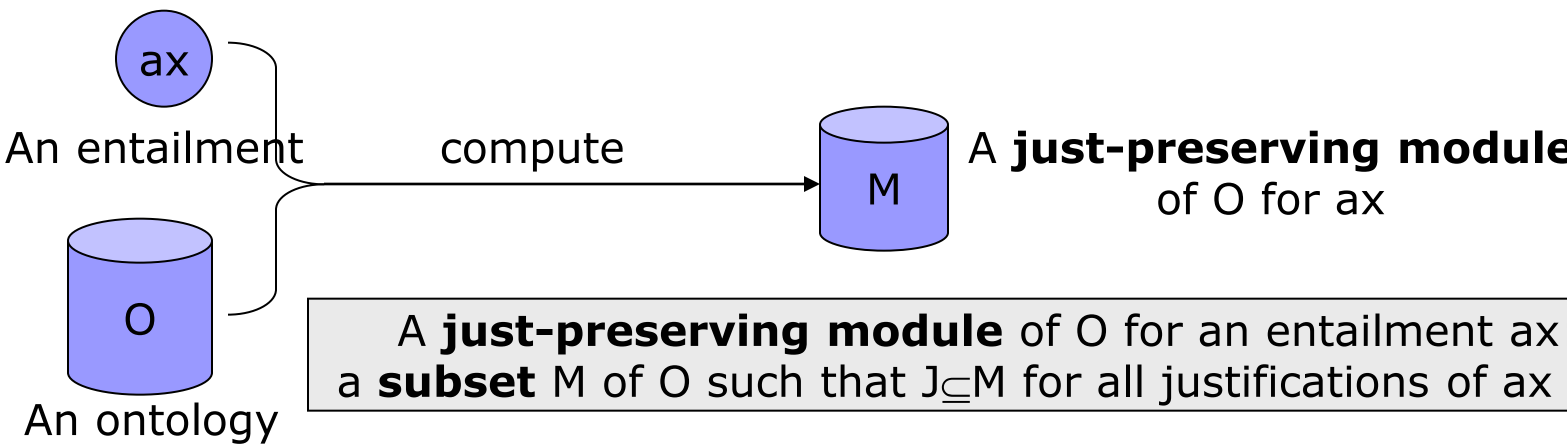
  Example:

$$\left.\begin{array}{ll} ax_1: & \mathsf{ChiefActress} \sqsubseteq \mathsf{Person} \\ ax_2: & \mathsf{ChiefActress} \sqsubseteq \mathsf{Actress} \\ ax_3: & \mathsf{Actress} \sqsubseteq \mathsf{Woman} \\ ax_4: & \mathsf{Person} \sqsubseteq \mathsf{Man} \sqcup \mathsf{Woman} \\ ax_5: & \mathsf{ChiefActress} \sqsubseteq \neg\mathsf{Man} \end{array}\right\} O$$

- The syntactic locality-based module w.r.t. ChiefActress is O={ax_1,…,ax_5}
- ☞ Size of the module can be still large
- Problem 2: contain all concept/role assertions

Goal-directed Approach

- The problem to be solved

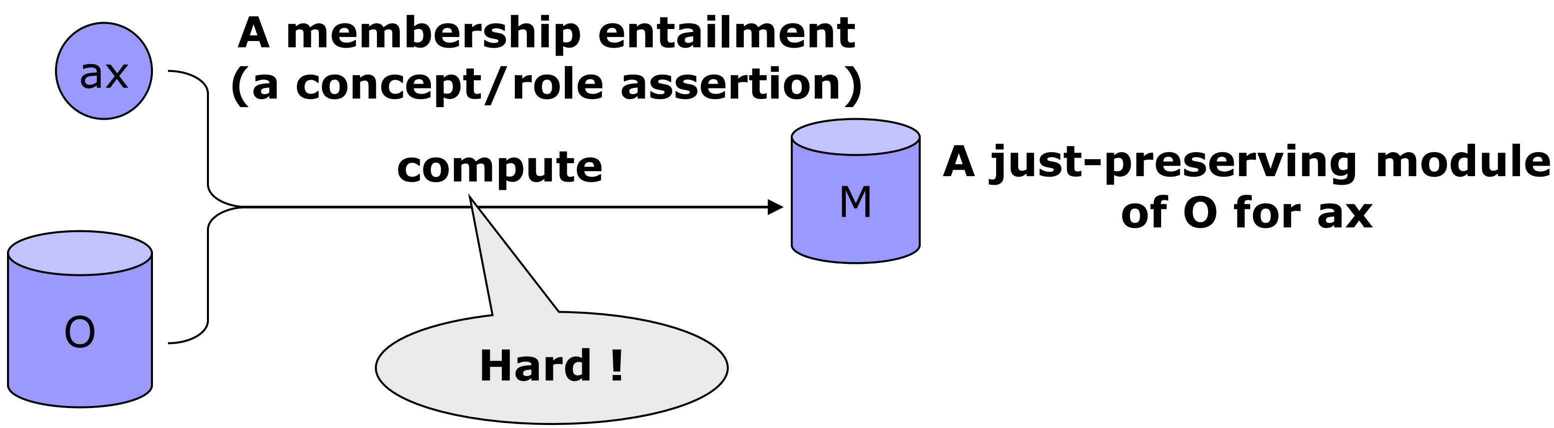


A **just-preserving module** of O for an entailment ax:
a **subset** M of O such that J$\subseteq$M for all justifications of ax in O

Analysis of the Problem

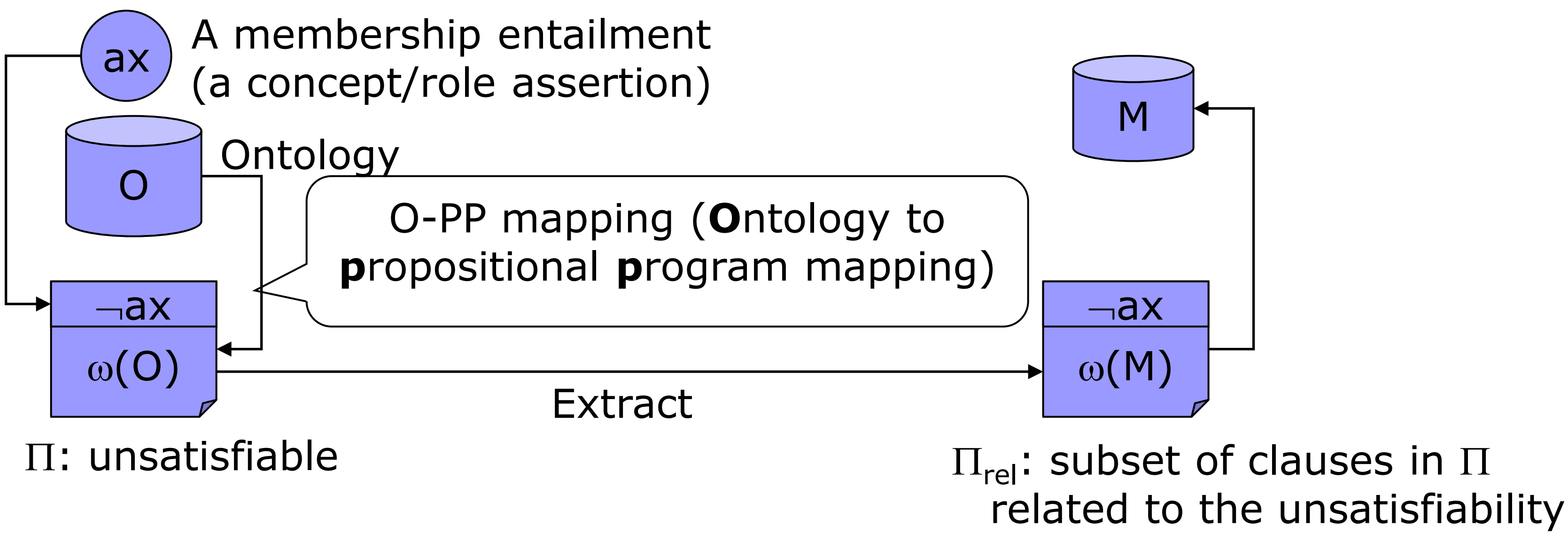
A just-preserving module of O for A⊆B ⇔ A just-preserving module of O' for B(a_A)

So, the problem we focus on is



Key Idea

- Inspiration
  - Relatively easy to analyze models of a propositional program
  - OWL DL can be translated to propositional logic



Experiments

- Aim: module extracted by goal-directed (GD) approach vs. syntactic locality-based module
  - Module size
  - Efficiency and scalability of the subsequent computation of all justifications
- Test ontologies
  - Real life ontologies: GALEN, GO, NCI

  ← 40 concept subsumption entailments per ontology
  - Benchmark ontologies: LUBM1/10, UOBM-Lite1/10

  ← 40 concept membership entailments per ontology

Offline Results

$\mathcal{O}$	$\lvert N_C \rvert$	$\lvert N_R \rvert$	$\lvert N_I \rvert$	$\lvert \mathcal{T} \rvert$	$\lvert \mathcal{A} \rvert$	Offline time(sec)
GALEN	2,748	412	0	4,529	0	1,431
GO	20,465	1	0	28,897	0	7,519
NCI	27,652	70	0	46,940	0	10,901
LUBM1	59	16	50,253	94	100,543	9
LUBM10	59	16	629,568	94	1,272,575	116
UOBM-Lite1	51	43	95,010	130	245,864	62
UOBM-Lite10	51	43	820,208	130	2,096,973	679

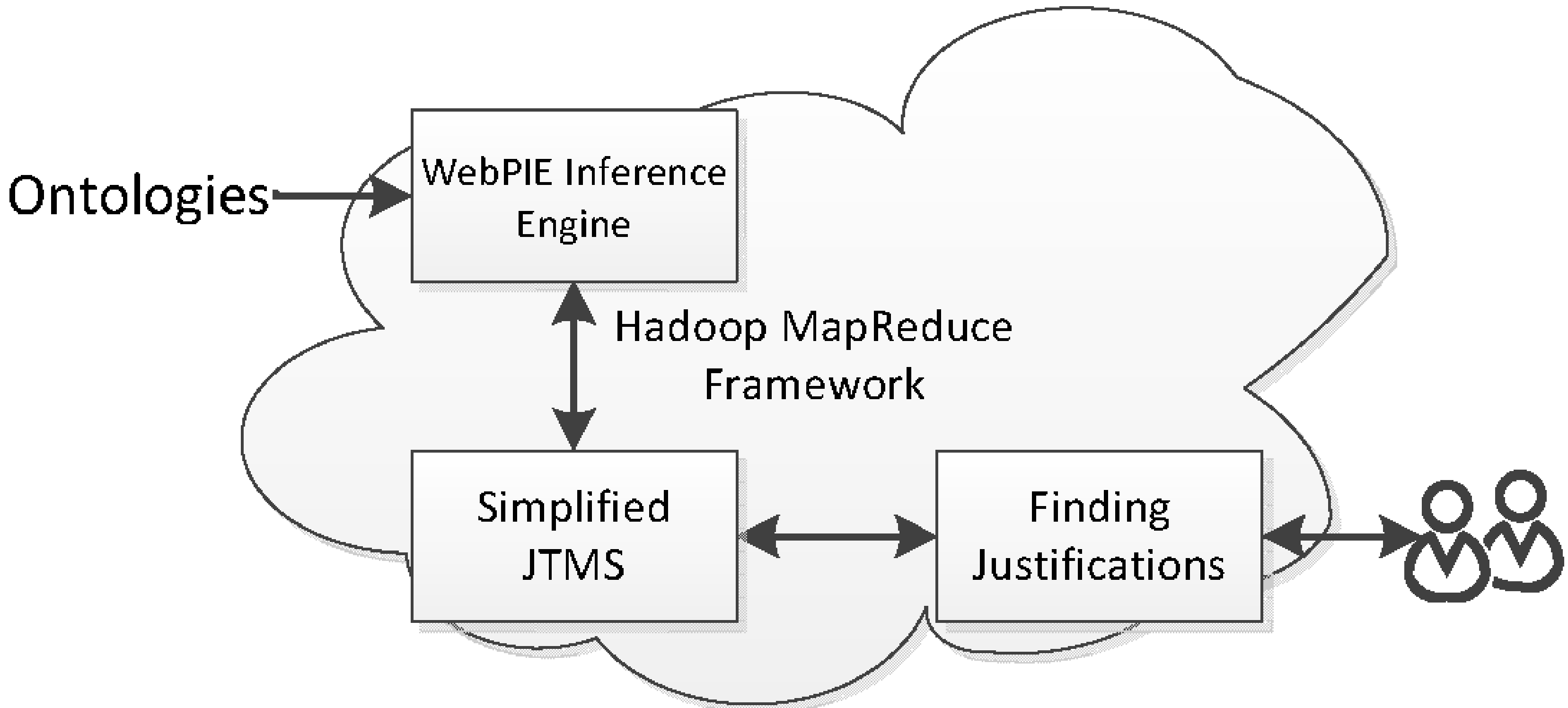
- The offline phase: costly, but reasonable
  - Independent of any given entailment on named objects
  - Tractable

Comparison Results

	Module Extr. by Our Method			Syn. Locality-based Module		
	#SH	SHT_{avg}(sec)	$Size_{avg}$	#SH	SHT_{avg}(sec)	$Size_{avg}$
GALEN	40	3.555	69.75	40	3.814	134.78
GO	40	7.314	9.55	40	11.985	32.25
NCI	40	4.065	7.23	40	7.518	70.95
LUBM1	40	69.061	22.15	20	201.481	100,596.00
LUBM10	40	95.721	20.48	0	MO_1	1,272,615.00
UOBM-Lite1	16	24.813	897.80	11	155.220	245,966.00
UOBM-Lite10	15	32.278	799.83	0	MO_2	2,097,047.00

- Modules extracted by GD modules $<_{size}$ locality-based modules
- Finding all justifications in GD modules $>_{efficient}$ Finding all justifications in locality-based modules
- Finding all justifications in GD modules $>>_{scalable}$ Finding all justifications in locality-based modules (against increasing number of ABox axioms)

MapReduce? Yes



OWL 2 RL

Fig.1:System Architecture

“Finding All Justifications of OWL Entailments Using TMS and MapReduce” CIKM 2011

Experiments (scalability)

DataSet	Time(min)	Speedup (Baseline: LUBM1000-2)
LUBM1000-2	39.25	1
LUBM1000-4	22.47	1.75
LUBM1000-8	16.98	2.31

DataSet	Time(min)	Speedup (Baseline: Dbpedia-1)
Dbpedia-1	11.78	1
Dbpedia-2	7.72	1.51
Dbpedia-4	6.26	1.88
Dbpedia-8	4.88	2.42

Speedup:

$$\frac{average\ \ just.time\ baseline}{average\ \ just.time}$$

Finding Justification in EL+

- **An incremental method to compute all Just**
  - Utilizes hierarch information obtained from classification
  - Reuse computed justifications
- Advantage: no labels are attached to entailed subsumption

"An Algorithm for Axiom Pinpointing in EL+ and its Incremental Variant" CIKM 2011 (poster)

Outline

- Justification for Debugging Ontologies
- Methods for Finding Justifications
- A Scalable Method for Debugging Large Inconsistent Ontologies
- Conclusion

Problem

- Problem of existing modularization-based optimizations
  - Are hard to be adapted
  - May not be useful if the union of all the MIS is large
- Example:

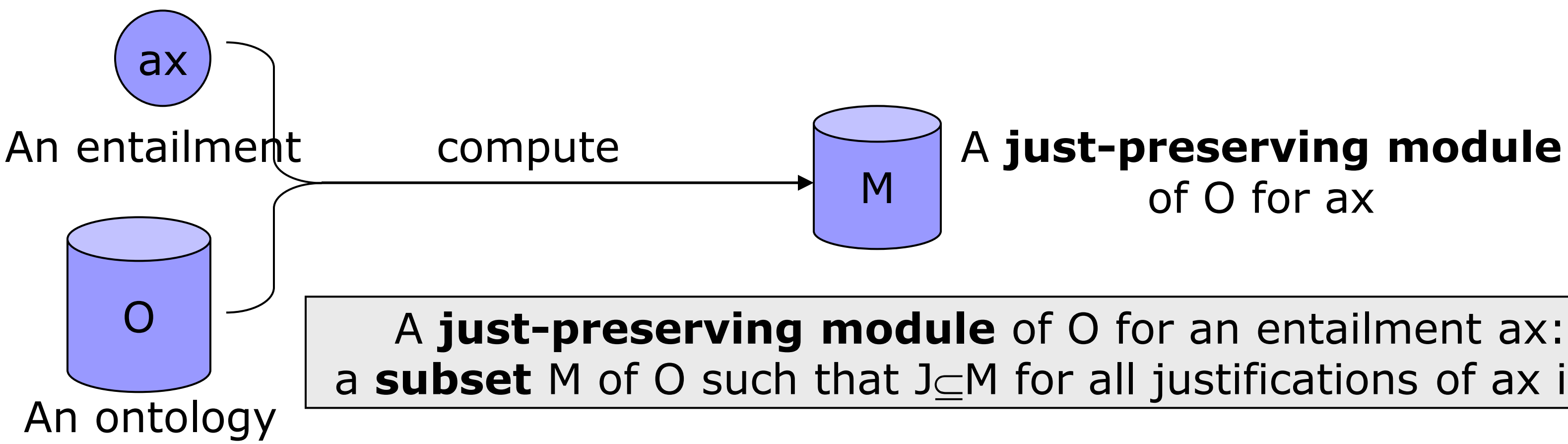
Tbox: $A \sqsubseteq B \quad B \sqcap D \sqsubseteq \bot$

Abox: $A(a1),\ldots,A(an)$
$C(a1),\ldots,C(an)$

The union of all the MIS is the ontology itself

Goal-directed Approach

- The problem to be solved

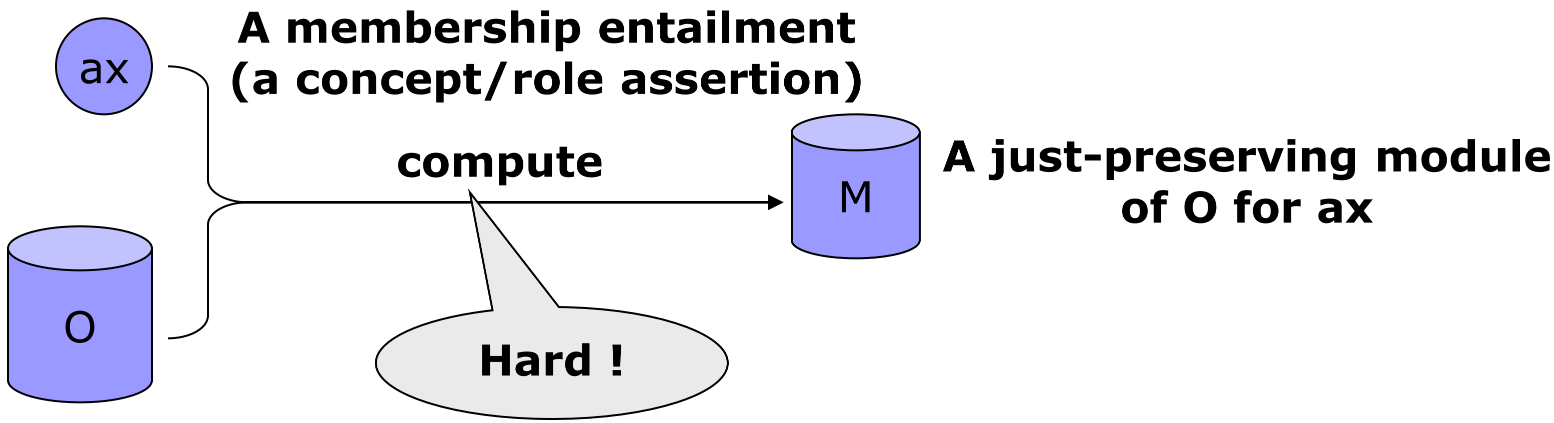


A **just-preserving module** of O for an entailment ax:
a **subset** M of O such that J⊆M for all justifications of ax in O

Analysis of the Problem

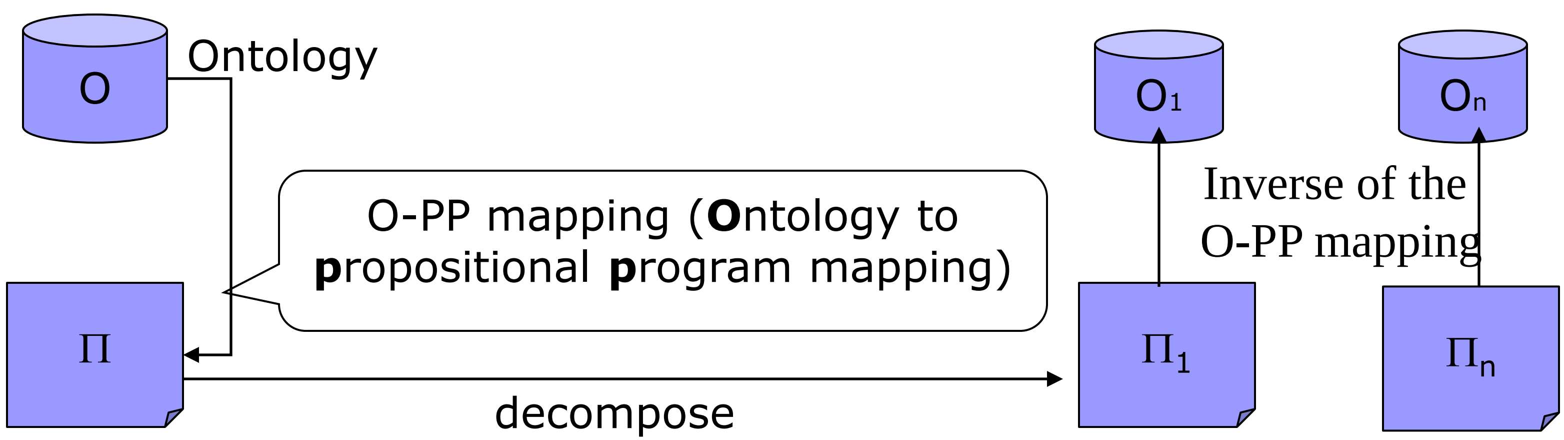
A just-preserving module of O for $A \sqsubseteq B$ ⇔ A just-preserving module of O' for $B(a_A)$

So, the problem we focus on is



Key Idea

- Compile an ontology to a propositional program
- Decompose the program
- Obtain a decomposition of the ontology



Experimental Data

Table 1. The characteristics of all test ontologies

$\mathcal{O}$	Expressivity	#C	#R	#I	#Ax
University	$\mathcal{SOIF}(\mathbf{D})$	30	12	34	74
Chemical	$\mathcal{ALCHF}$	48	20	48	162
MiniTambis	$\mathcal{ALCF}$	183	44	183	350
UOBM-Lite1$_{+50\sim+250}$	$\mathcal{SHIF}(\mathbf{D})$	51	43	95,113–95,522	246,144–246,744
UOBM-Lite5$_{+50\sim+250}$				420,251–420,662	1,075,340–1,075,940
UOBM-Lite10$_{+50\sim+250}$				820,358–820,958	2,097,253–2,097,853

Note: "#C", "#R"", "#I"" and "#Ax" are respectively the numbers of atomic concepts, atomic roles, individuals and axioms in $\mathcal{O}$.

Results

Table 2. Typical comparison results and some runtime statistics

$\mathcal{O}$	Pellet	Ours	Compile	Decompose	#MIS	#Sub	#A-S	#M-S
University	0:02:40	0:00:25	0:00:04	0:00:01	8	8	10	1
Chemical	>4:00:00	0:02:39	0:00:16	0:00:01	362	37	31	21
MiniTambis	>4:00:00	0:03:10	0:00:59	0:00:01	37	30	19	2
UOBM-Lite1$_{+50}$	>4:00:00	0:06:47	0:01:33	0:00:03	50	47	7	2
UOBM-Lite1$_{+250}$	>4:00:00	0:20:04	0:01:38	0:00:04	250	140	29	9
UOBM-Lite5$_{+50}$	>4:00:00	0:35:02	0:10:08	0:00:28	50	50	5	1
UOBM-Lite10$_{+50}$	out of mem	1:14:33	0:15:36	0:00:32	50	50	5	1

Note: "Pellet" (resp. "Ours") is the total time Pellet (resp. our system) spends to compute all MISs of $\mathcal{O}$; "Compile" (resp. "Decompose") is the time spent in the O-PP compilation process (resp. the decomposition process); "#MIS" is the number of MISs of $\mathcal{O}$; "#Sub" is the number of extracted sub-ontologies that are not in any consistent bin; "#A-S" (resp. "#M-S") is the maximum number of axioms (resp. MISs) in every extracted sub-ontology that is not in any consistent bin.

Results

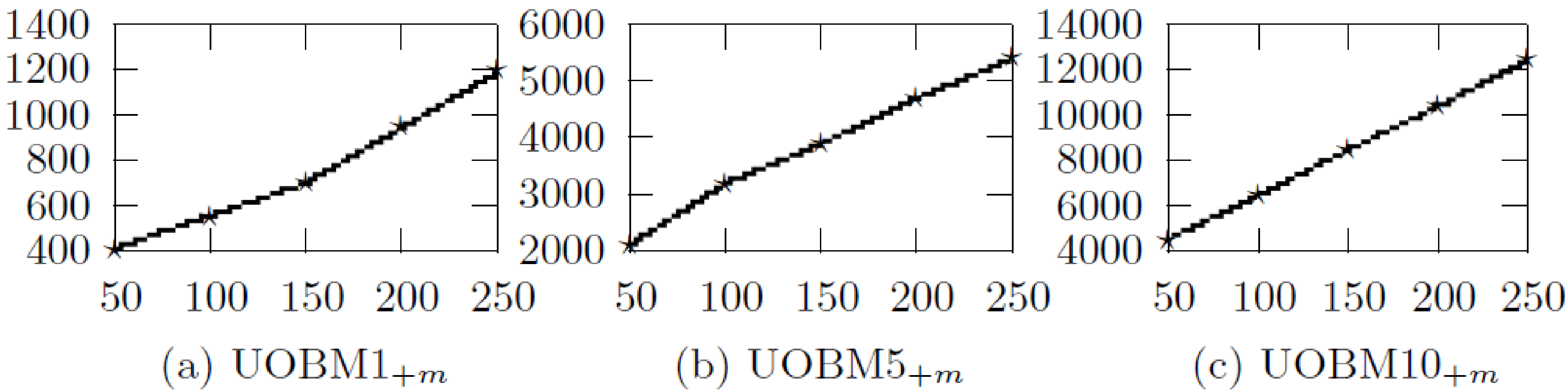


Fig. 1. The total execution time (seconds) against increasing numbers of conflicts

Conclusion

- Finding all justifications is a hard task
- There are methods that are practical
  - But they do not scale to large ontologies
- Our solutions
  - Modularization: syntactic locality-based module extraction and goal-directed module extraction
  - Optimization based on MapReduce
  - Incremental computation of justifications
  - A decomposition-based optimization method

References

- Gang Wu, **Guilin Qi**, Jianfeng Du. Finding All Justifications of OWL Entailments Using TMS and MapReduce, Proc. of19th ACM Conference on Information and Knowledge Management (CIKM), 2011.
- Xiaojun Cheng, **Guilin Qi**. An Algorithm for Axiom Pinpointing in EL+ and its Incremental Variant, Proc. of 19th ACM Conference on Information and Knowledge Management (CIKM), 2011.
- Jianfeng Du and **Guilin Qi**. Decomposition-based Optimization for Debugging of Inconsistent OWL DL Ontologies. Proceedings of the 4th International Conference on Knowledge Science, Engineering and Management (KSEM), 39–50, 2010.
- Jianfeng Du, **Guilin Qi** and Qiu Ji. Goal-Directed Module Extraction for Explaining OWL DL Entailments, In Proceedings of the 8th International Semantic Web Conference (ISWC), 163-179, 2009.
- Boontawee Suntisrivaraporn, **Guilin Qi**, Qiu Ji, Peter Haase. *A Modularization-based Approach to Finding All Justifications for OWL DL Entailments*, in the 3rd Asian Semantic Web Conference (ASWC), 1-15, 2008.

Debugging mappings in ontology networks

Outline

- Background & Motivating Example
- Preliminaries/Theory
- Algorithms
  - Reasoning Components
  - Local Optimal Diagnosis
  - Global Optimal Diagnosis
- Implemented System
- Experimental Results
- User Support

Outline

- **Background & Motivating Example**
- Preliminaries/Theory
- Algorithms
  - Reasoning Components
  - Local Optimal Diagnosis
  - Global Optimal Diagnosis
- Implemented System
- Experimental Results
- User Support

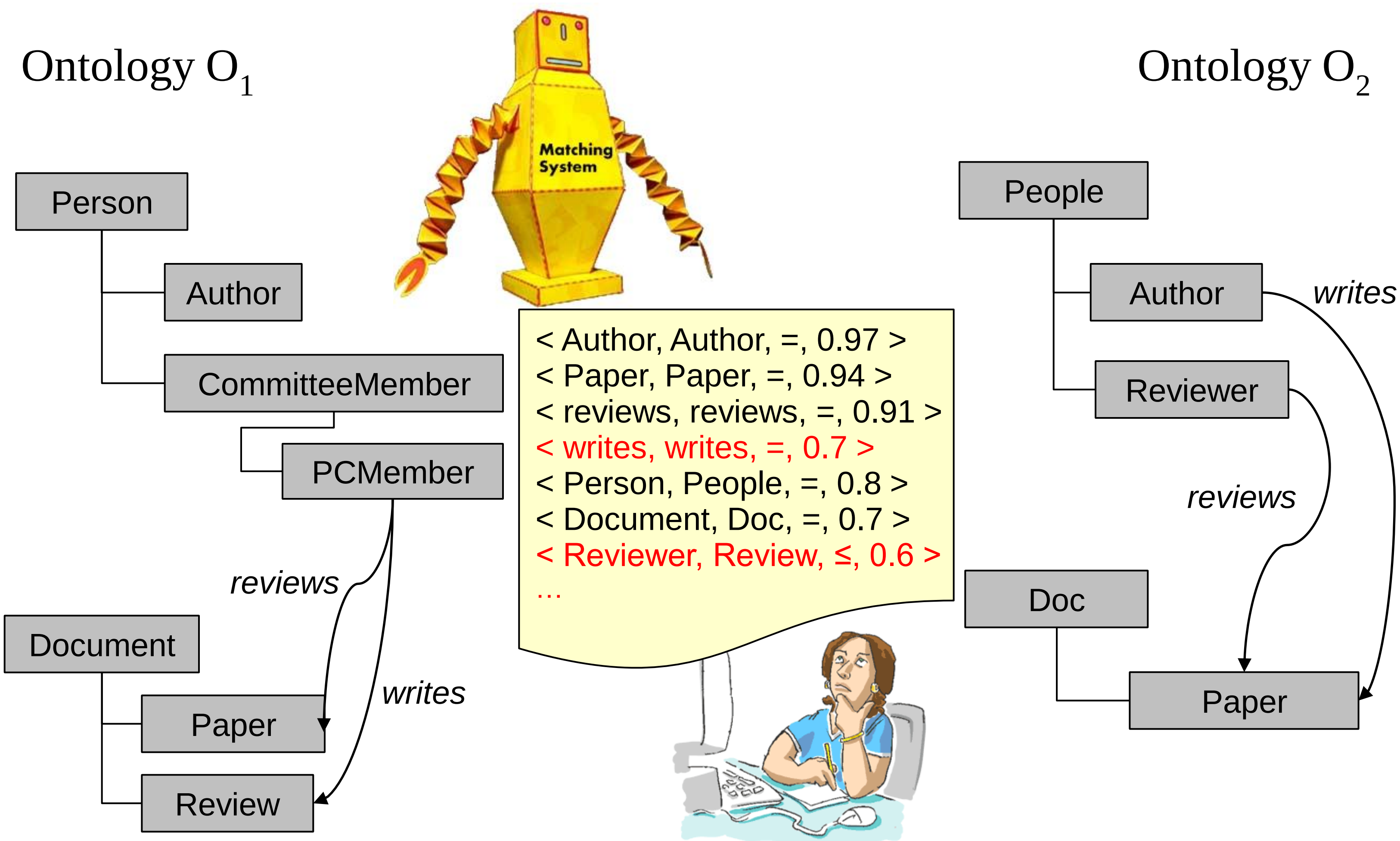
Terminology

- Ontology Matching is a process that creates alignments (or mappings)

- Alignments are sets of correspondences

- Correspondences are links between concepts, properties or instances of two ontologies

Ontology Matching

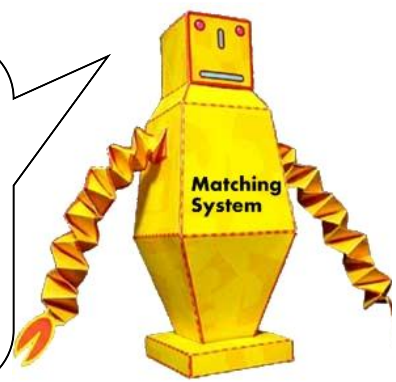


Correspondence

$$< e_1, \checkmark \quad e_2, \checkmark \quad r, ? \quad n \; ? >$$

Entity of O_1 e.g. a concept	Entity of O_2	Semantic relation e.g. subsumption	Confidence value e.g. $n \in [0, 1]$

Even though i <u>do not know exactly</u> how to interprete ‚subsumption' or ‚equivalence' don't bug me, **I can nevertheless do my job!**



Well, I have an **intuitive understanding** of these relations!

Reductionistic Alignment Semantic

- A reductionistic alignment semantic S is a function that maps an alignment A between O_1 and O_2 on a set of axioms X.

- The aligned ontology $A_S(O_1,O_2)$ is defined as $O_1 \cup O_2 \cup X$ where S refers to some semantics

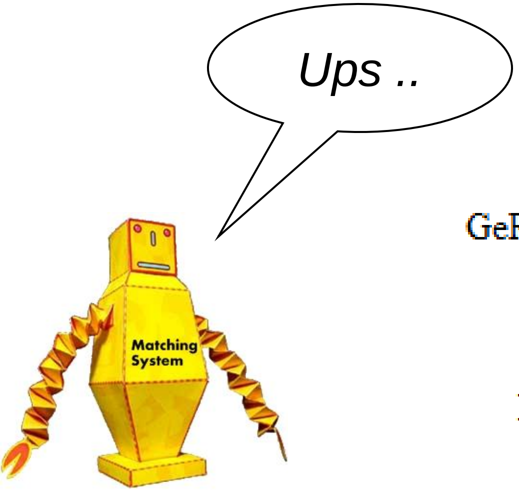
- Natural Semantics
  - X results from a 1:1 mapping from correspondences to axioms
    - $\langle$ Person, Human, =, 0.9 $\rangle \mapsto$ Person $\equiv$ Human
    - $\langle$ createdBy, writtenBy, >, 0.75 $\rangle \mapsto$ createdBy $\sqsupseteq$ writtenBy

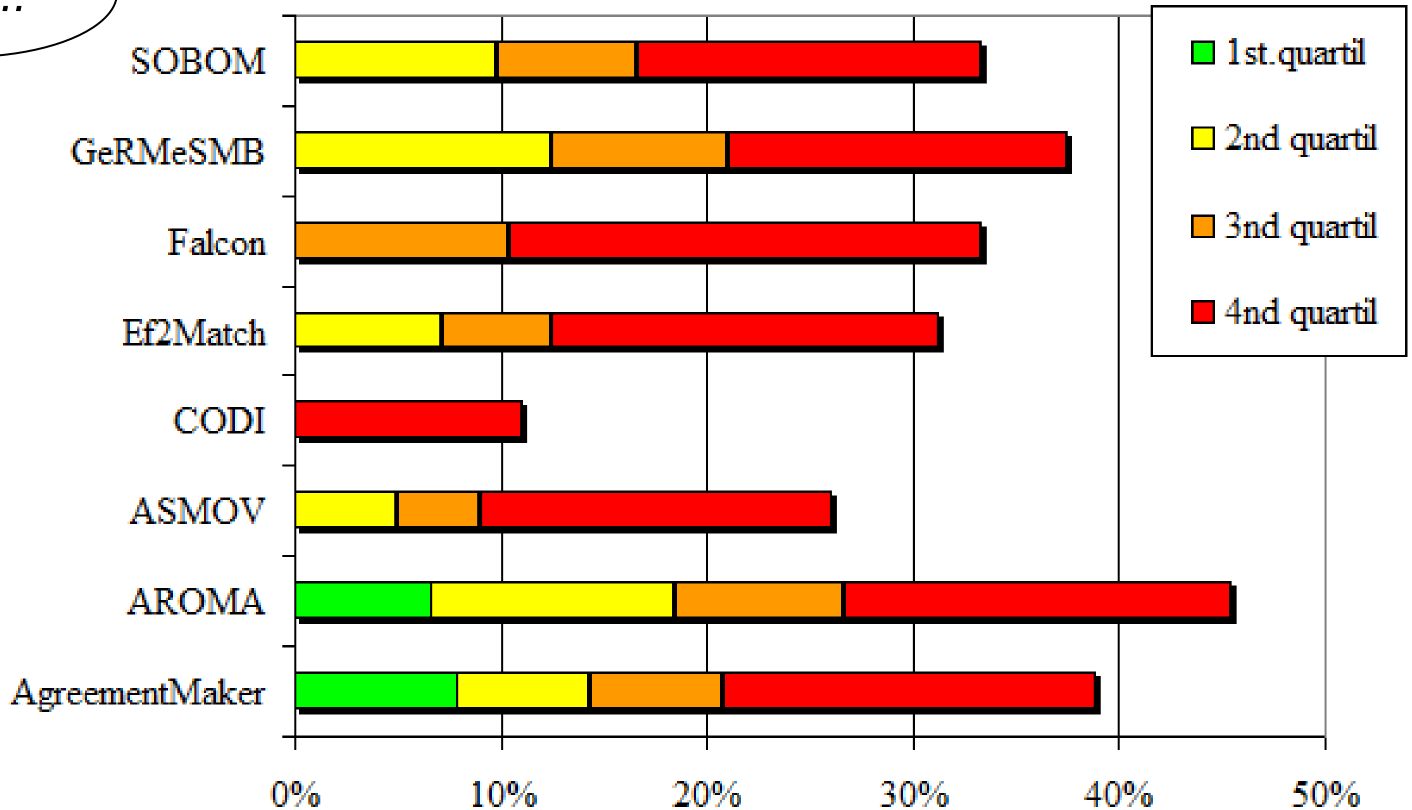
Alignment Incoherence

- Alignment makes an satisfiable concept unsatisfiable

- Concrete definition depends on:
  - What kind of concepts are taken into account?
    - Only atomic concepts as **Person, Document**
    - Also concepts of type **9 hasWritten.>**
  - Semantics of alignments/correspondences?
    - Distributed Description Logics
    - “Natural Semantics” = direct translation to axioms

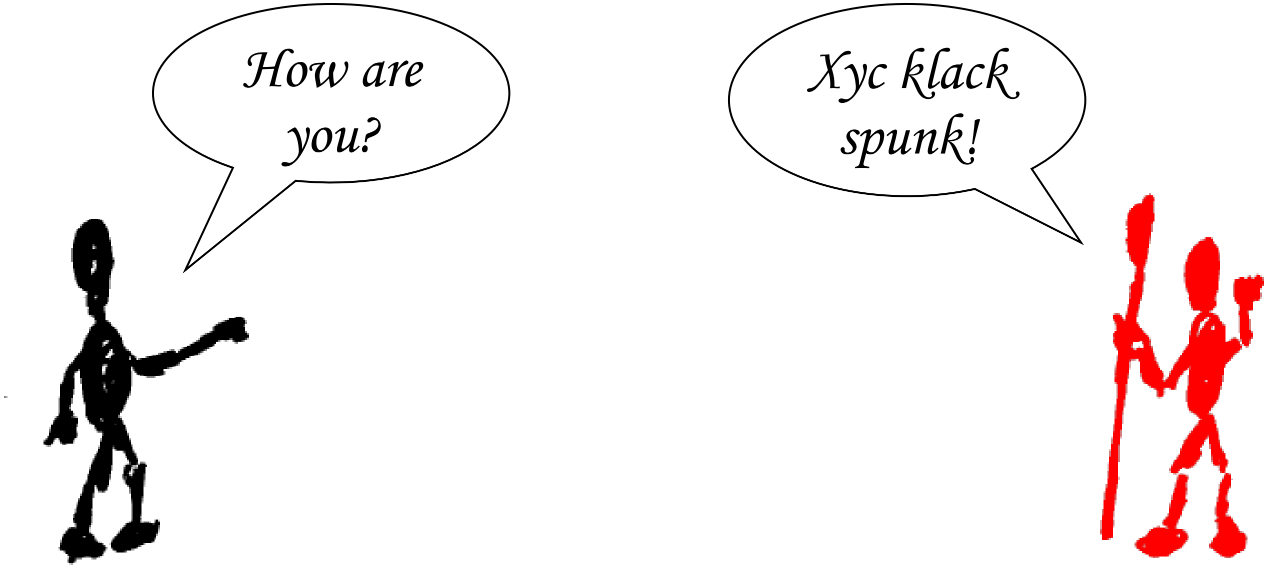
Degree of Incoherence



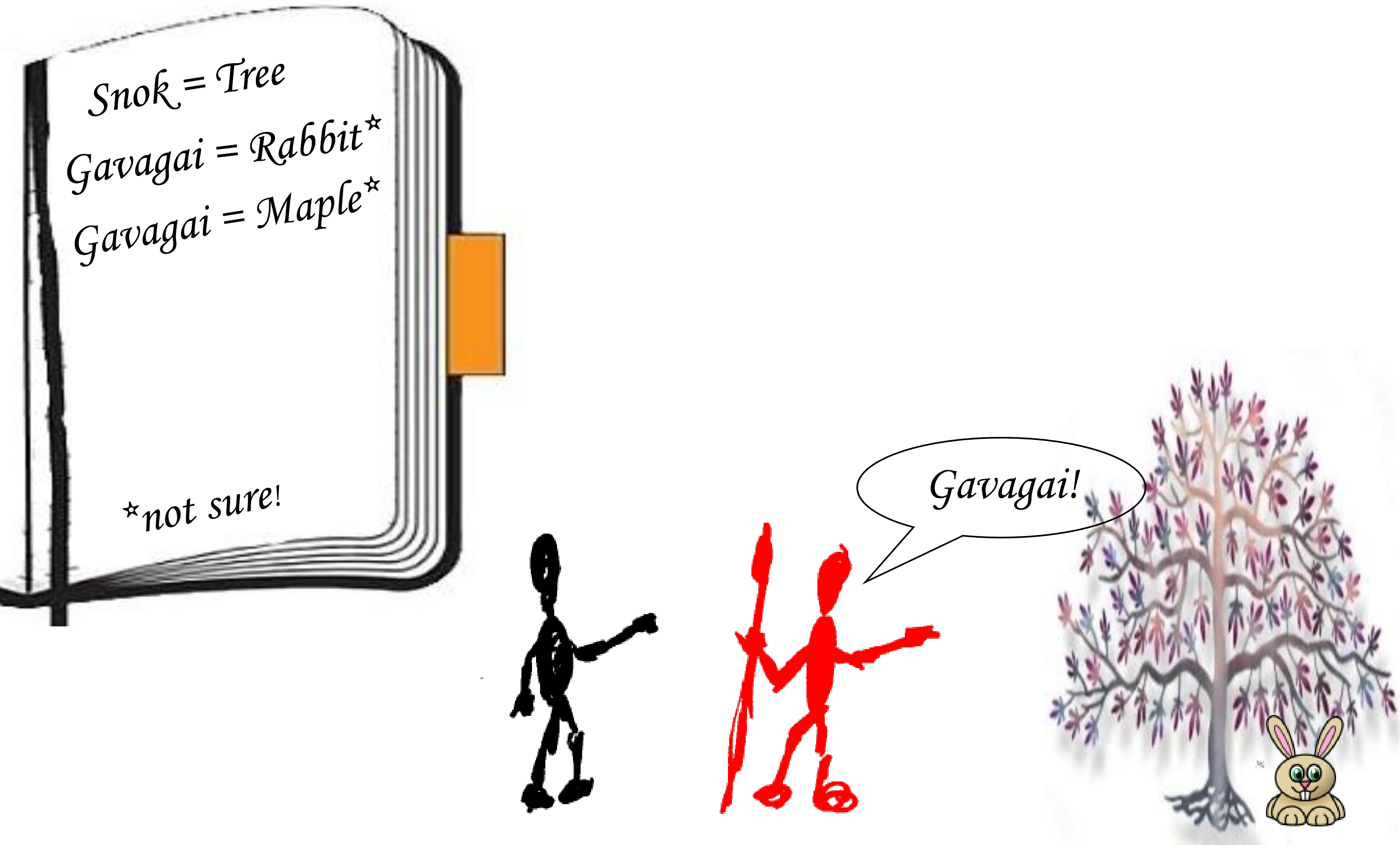


Motivating Example

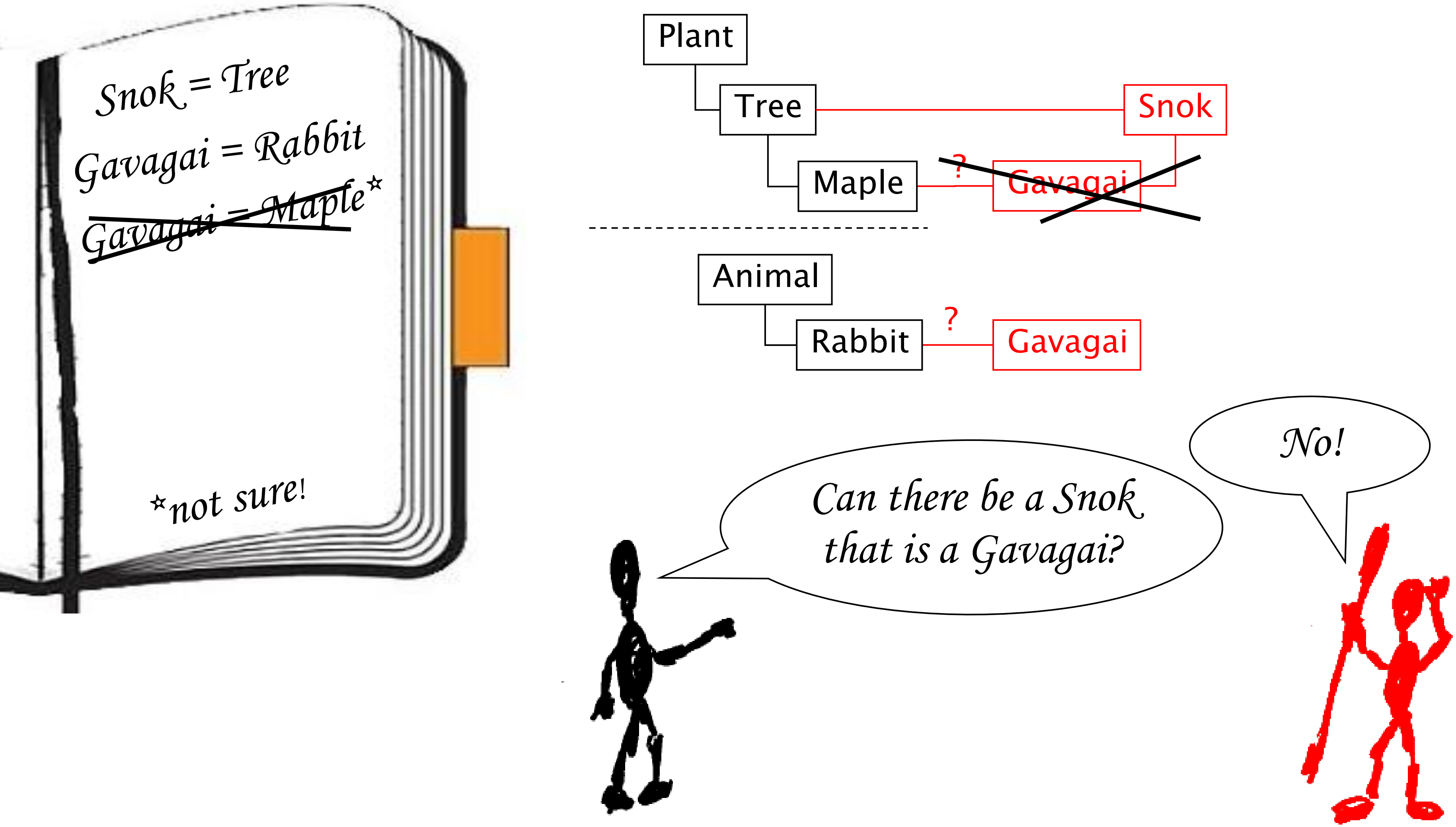
- Translating between English and an unknown language



Motivating Example



Motivating Example



Example in DL

$O_1 = \{$
$\quad Maple \sqsubseteq Tree \sqsubseteq Plant$
$\quad Rabbit \sqsubseteq Animal$
$\quad Animal \sqsubseteq \neg Plant$
$\}$

$A = \{$
$\quad Tree \equiv Snok$
$\quad Maple \equiv Gavagai$
$\}$

$O_2 = \{$
$\quad Gavagai \sqsubseteq \neg Snok$
$\}$

$A_S(O_1, O_2) \models Gavagai \sqsubseteq Snok$

$A_S(O_1, O_2) \models Gavagai \sqsubseteq : Snok$

… and thus $\mathbf{A_S(O_1, O_2) \models Gavagai \sqsubseteq \bot}$

Reasoning-sensitive Applications

- Alignments can be used in many applications
  - Instance Migration
  - Query Rewriting
  - Ontology Merging
  - …

- Incoherent alignments result in inconsistencies or empty result sets

Why Coherent Alignments?

- Reasoning-sensitive applications need coherent alignments
- Positive impact on the precision of alignments

Outline

- Background & Motivating Example
- **Preliminaries/Theory**
- Algorithms
  - Reasoning Components
  - Local Optimal Diagnosis
  - Global Optimal Diagnosis
- Implemented System / Availability
- Experimental Results
- User Support

Diagnosis - Idea

- Introduced by Reiter (1987):
  - Dermine a set of those system components which, when assumed to be functioning abnormally, explain the discrepancy between observed and correct behaviour.

- System is the aligned ontology $A_S(O_1, O_2)$
- Abnormal behaviour = Incoherence
- Elements from A are assumed to be incorrect

Conflict Sets and MIPS

- Reiter talks about conflict sets, which are minimal subsets of system components leading to abnormal behaviour

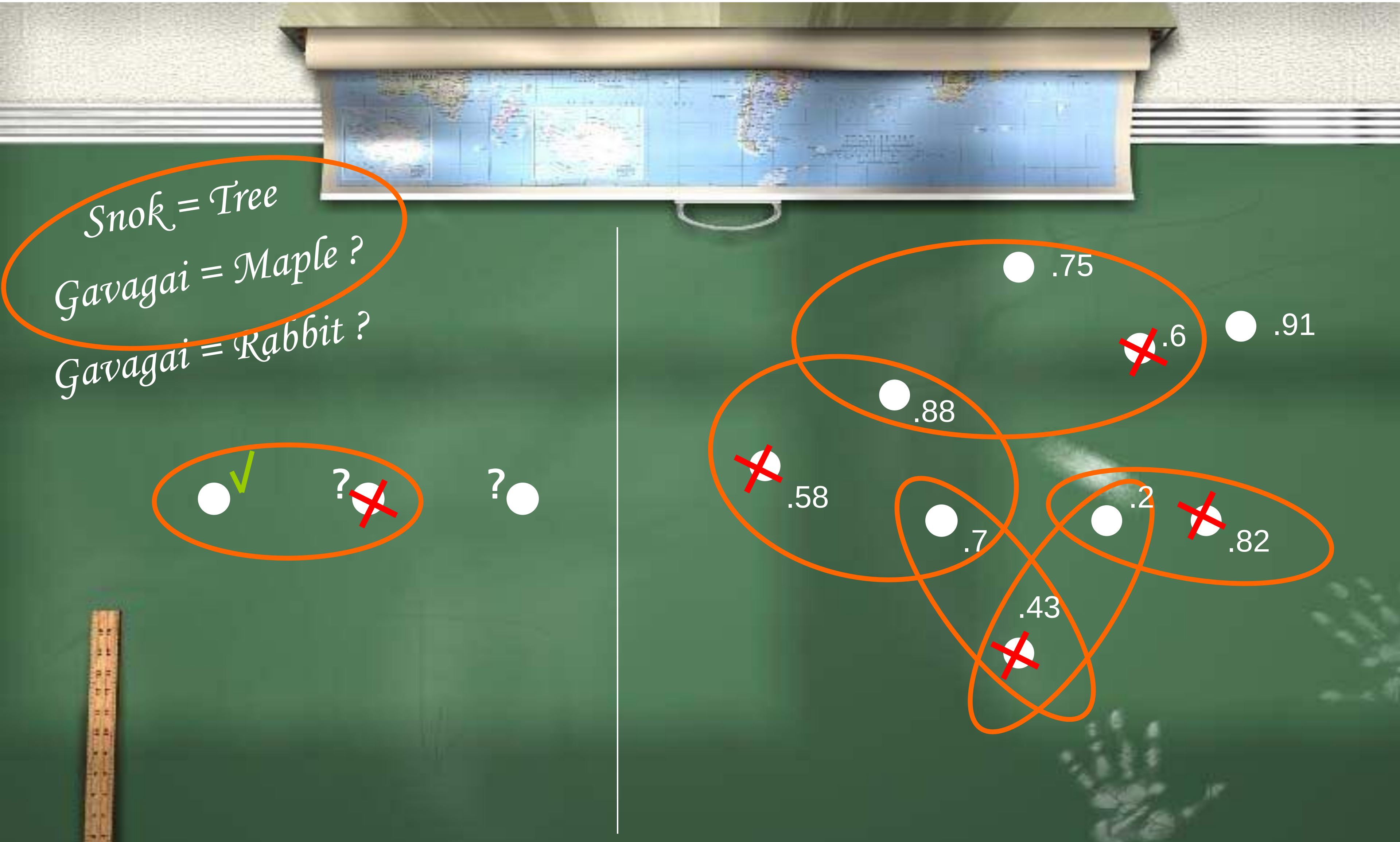
- MIPS = Minimal Incoherence Preserving Subalignment
  - M µ A is a MIPS iff
    - M is incoherent
    - Each M’ ½ M is coherent

Diagnosis

- A subset ¢ µ A of an incoherent alignment A is diagnosis for A (w.r.t. O_1 and O_2) iff
  - A n ¢ is coherent and there exists no ¢' ½ ¢ such that An¢' is coherent

- It follows:
  - ¢ is a minimal hitting set over all MIPS in A

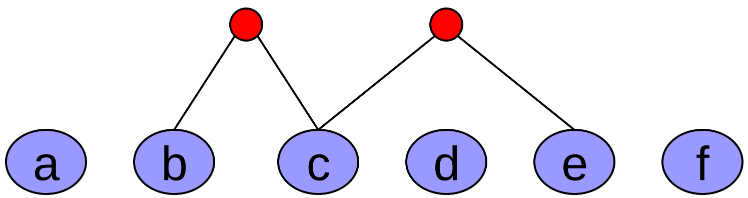
Two Examples



First Idea

- Pick a randomly chosen MIPS and remove ‘worst’ correspondence
- Repeat until no MIPS is left
- **Algorithm fails in constructing minimal hitting set**

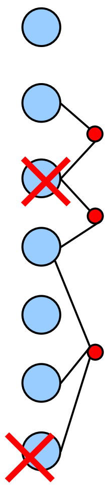
low confidence



high confidence

Local Optimal Diagnosis

high confidence



low confidence

Definition: Accused correspondence

A correspondence $c \in A$ is accused by A iff there exists a MIPS in A with $c \in M$ such that for all $c' \neq c$ in M it holds that

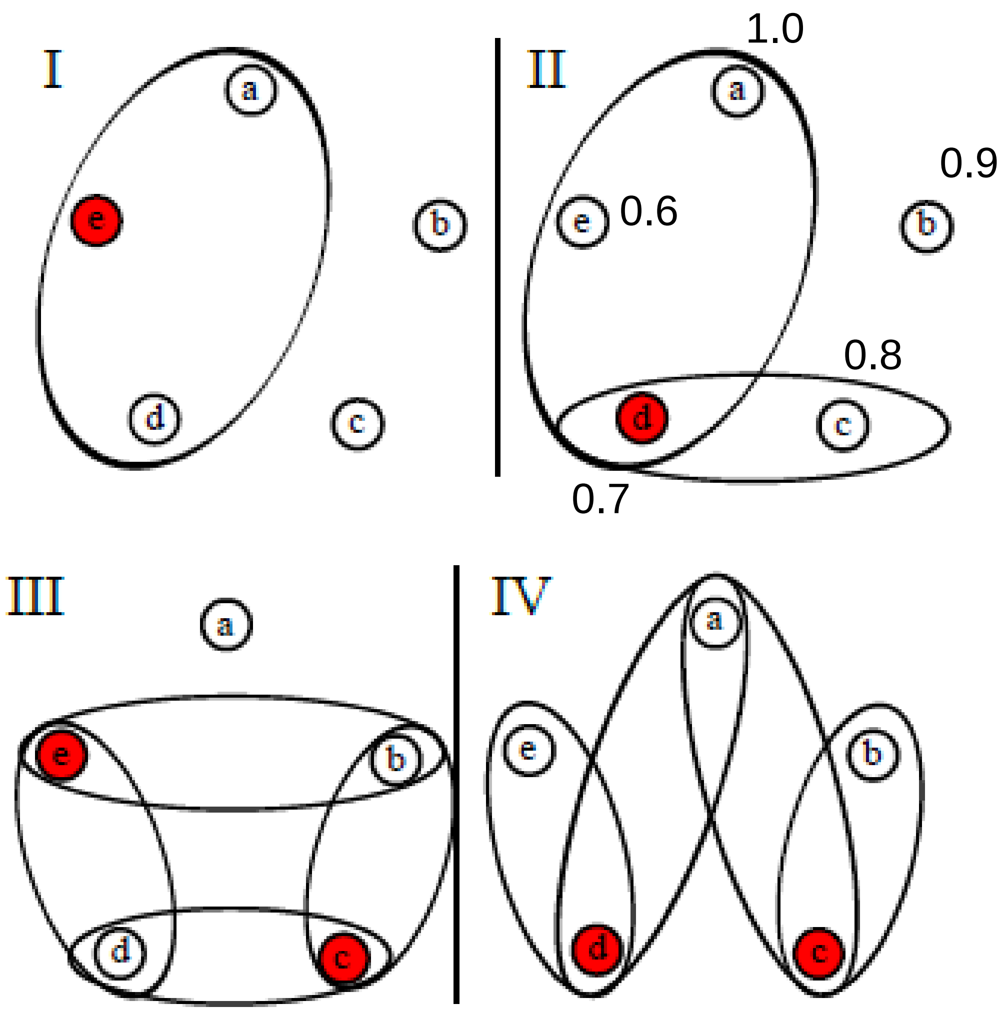
- (1) $®(c') \geq ®(c)$ and
- (2) c‘ is not accused by A.

important!

Definition: Local optimal diagnosis (LOD)

The set of all accussed correspondences is referred to as local optimal diagnosis (LOD).

Local Optimal Diagnosis: Example



Confidences

®(a) = 1.0
®(b) = 0.9
®(c) = 0.8
®(d) = 0.7
®(e) = 0.6

Global Optimal Diagnosis

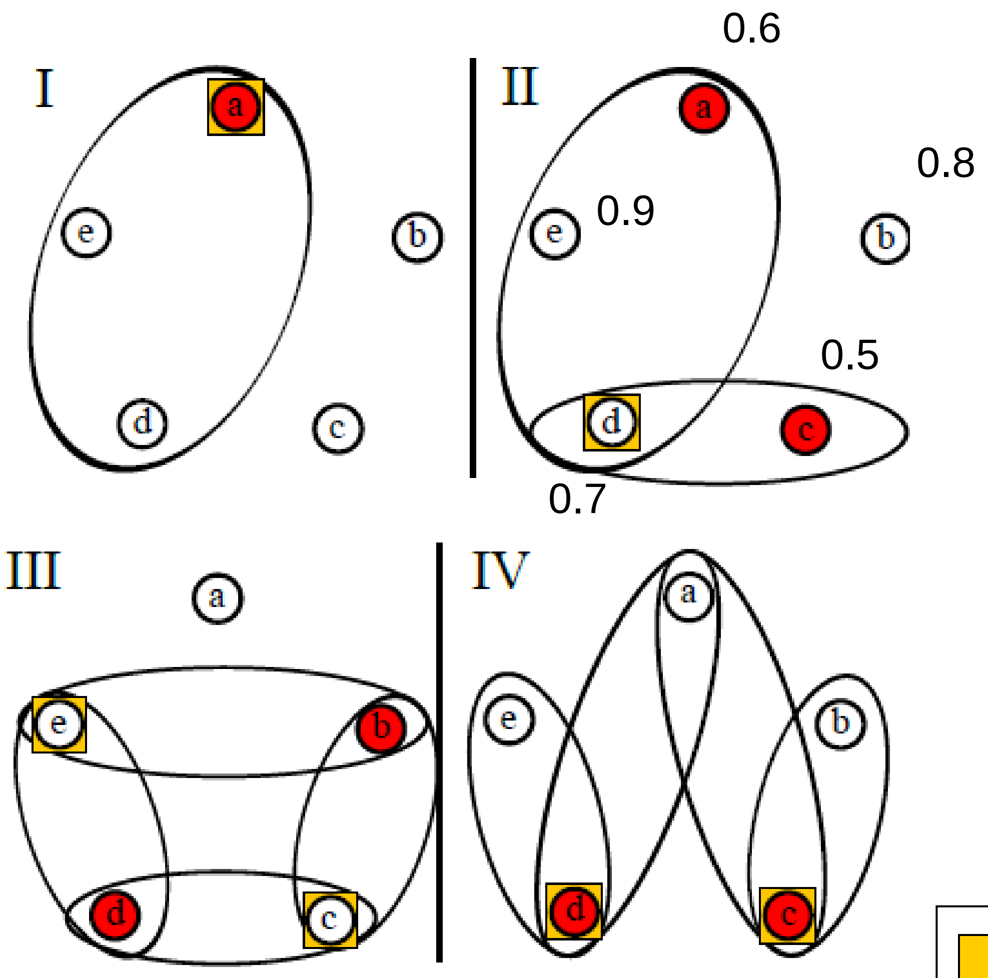
Definition: Global optimal diagnosis

¢ µ A is a global optimal diagnosis for an inocherent alignment A, iff

- A n ¢ is coherent
- there exists no ¢‘ µ A with ∑ $_{c\ 2\ ¢‘}$ ®(c) < ∑ $_{c\ 2\ ¢}$ ®(c) for which A n ¢‘ is coherent

- A diagnosis (= minimal hitting set) which “removes as less confidence as possible”.
- Note: Requires to compute the smallest weighted hitting set

Global Optimal Diagnosis



Confidences

®(a) = 0.6
®(b) = 0.8
®(c) = 0.5
®(d) = 0.7
®(e) = 0.9

Differences between local and global optimal diagnosis occur only if MIPS are overlapping

Outline

- Background & Motivating Example
- Preliminaries/Theory
- **Algorithms**
  - **Reasoning Components**
  - Local Optimal Diagnosis
  - Global Optimal Diagnosis
- Implemented System / Availability
- Experimental Results
- User Support

Incoherence Detection

- Test for incoherence
  - Classify both ontologies and check for unsatisfiable classes C
  - Classify the aligned ontology and check for unsatisfiable classes C_A
  - Compare C and C_A

MIPS Detection

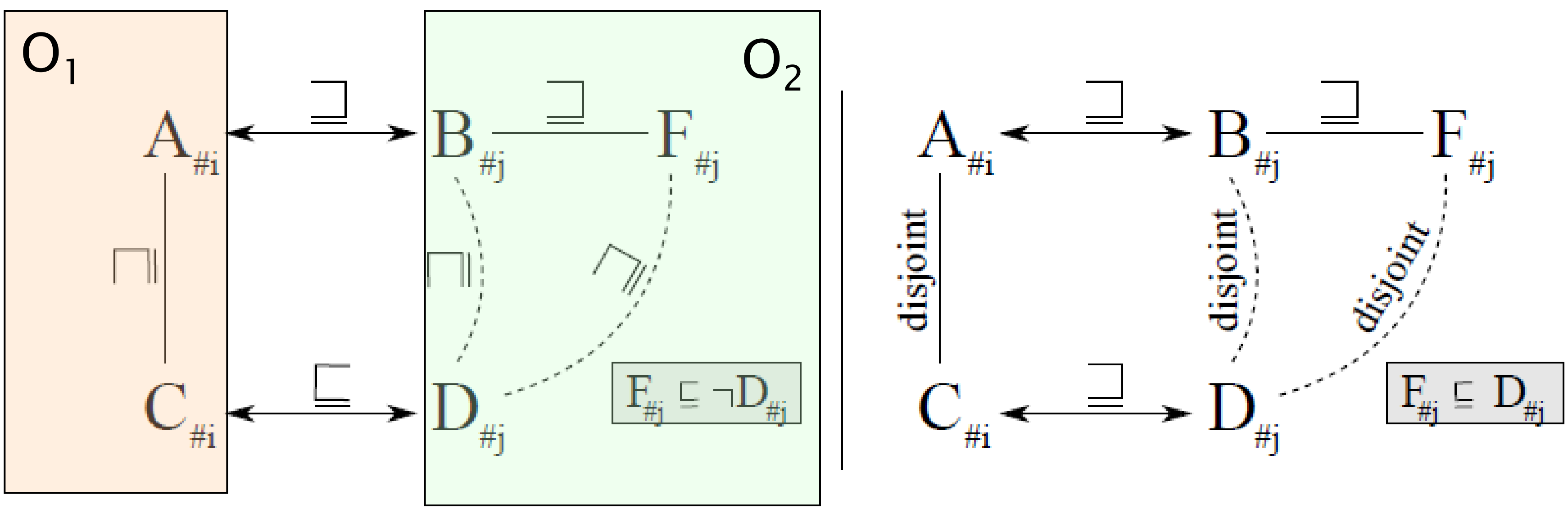
- Expand and shrink algorithm to find one MIPS proposed by Kalyanpur (2006) for ontology debugging
- Iterative procedure that requires |A| times reasoning in ther aligned ontologies in worst case
- Inefficient:
  - Each step in the loop requires to compute unsatisfiable classes in $A_S(O_1, O_2)$ because A has changed
  - Algorithm finds only one MIPS per iteration

Black-Box vs. White-Box

- Approach on last slides are black-box approaches
- White-Box approach works differently:
  - Compute all MUPS for one unsatisfiable concept by single call to reasoner
  - Trace relevant axioms that resulted in unsatifiability in tableau
  - Might be more efficient … but not directly applicable to debugging alignments

Pattern-based Reasoning

- Idea: Use incomplete method for incoherence detection for pairs of correspondences
- First classify O_1 and O_2 once, then check certain patterns for all pairs of correspondences



MIPS Example I

Alignment	
(1) $\langle Acceptance_{\#1}, AcceptedPaper_{\#2}, \equiv \rangle$	
(2) $\langle Paper_{\#1}, Paper_{\#2}, \equiv \rangle$	
Axioms of ontology #1	**Axioms of ontology #2**
(3) $Paper \sqsubseteq Document$	(6) $AcceptedPaper \sqsubseteq EvaluatedPaper$
(4) $Acceptance \sqsubseteq Decision$	(7) $EvaluatedPaper \sqsubseteq AssignedPaper$
(5) $Document \sqsubseteq \neg Decision$	(8) $AssignedPaper \sqsubseteq SubmittedPaper$
	(9) $SubmittedPaper \sqsubseteq Paper$
Entailments	
(10) $Acceptance_{\#1} \equiv AcceptedPaper_{\#2}$	*from (1)*
(11) $Paper_{\#1} \equiv Paper_{\#2}$	*from (2)*
(12) $AcceptedPaper_{\#2} \sqsubseteq Paper_{\#2}$	*from (6), (7), (8) and (9)*
(13) $Acceptance_{\#1} \sqsubseteq Paper_{\#2}$	*from (10) and (12)*
(14) $Acceptance_{\#1} \sqsubseteq Paper_{\#1}$	*from (11) and (13)*
(15) $Acceptance_{\#1} \sqsubseteq Document_{\#1}$	*from (3) and (14)*
(16) $Acceptance_{\#1} \sqsubseteq \neg Decision_{\#1}$	*from (5) and (15)*
(17) $Acceptance_{\#1} \sqsubseteq \bot$	*from (4) and (16)*

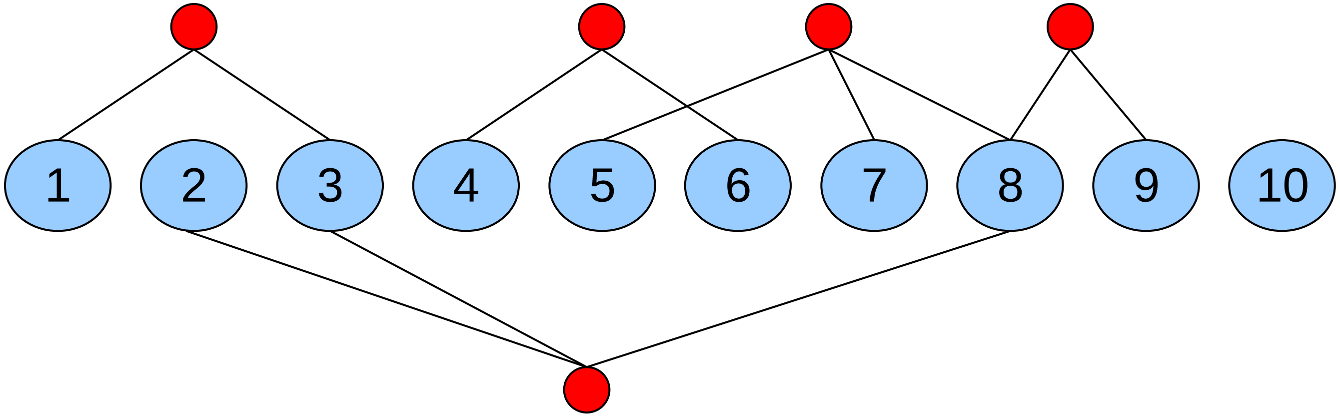
MIPS Example II

Alignment		
(1) $\langle \mathit{rejectPaper}_{\#1}, \mathit{reviewerOfPaper}_{\#2}, \equiv \rangle$		
Axioms of ontology #1	**Axioms of ontology #2**	
(2) $\exists_{\geq 2} \mathit{rejectPaper}^{-1}.\top \sqsubseteq \bot$	(3) $\mathit{reviewerOfPaper} \equiv \mathit{hasReviewer}^{-1}$	
	(4) $\mathit{AssignedPaper} \sqsubseteq \exists_{\geq 3} \mathit{hasReviewer}.\top$	
Entailments		
(5) $\mathit{rejectPaper}_{\#1} \equiv \mathit{reviewerOfPaper}_{\#2}$	*from (1)*	
(6) $\mathit{rejectPaper}^{-1}_{\#1} \equiv \mathit{hasReviewer}_{\#2}$	*from (3) and (5)*	
(7) $\mathit{rejectPaper}_{\#1} \equiv \mathit{hasReviewer}^{-1}_{\#2}$	*from (6)*	
(8) $\mathit{AssignedPaper}_{\#2} \sqsubseteq \exists_{\geq 3} \mathit{rejectPaper}^{-1}_{\#1}.\top$	*from (4) and (7)*	
(9) $\mathit{AssignedPaper}_{\#2} \sqsubseteq \exists_{\geq 2} \mathit{rejectPaper}^{-1}_{\#1}.\top$	*from (8)*	
(10) $\mathit{AssignedPaper}_{\#2} \sqsubseteq \bot$	*from (2) and (9)*	

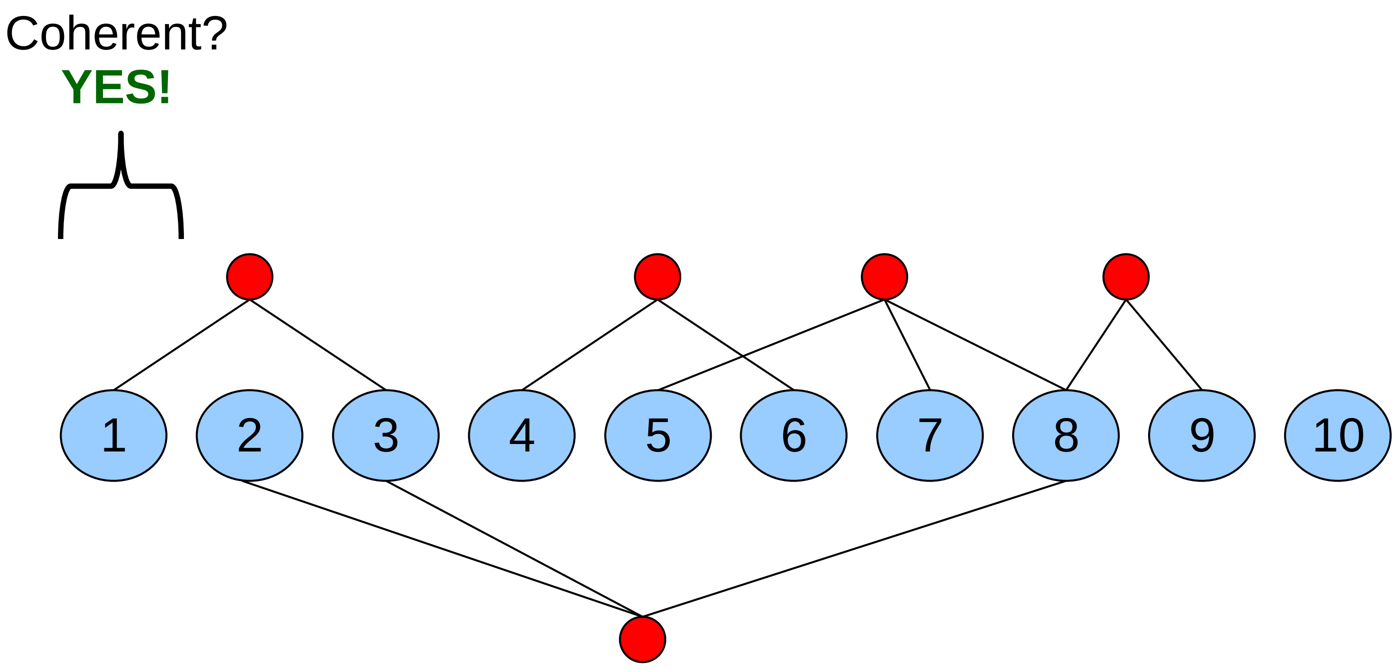
Outline

- Background & Motivating Example
- Preliminaries/Theory
- **Algorithms**
  - Reasoning Components
  - **Local Optimal Diagnosis**
  - Global Optimal Diagnosis
- Implemented System / Availability
- Experimental Results
- User Support

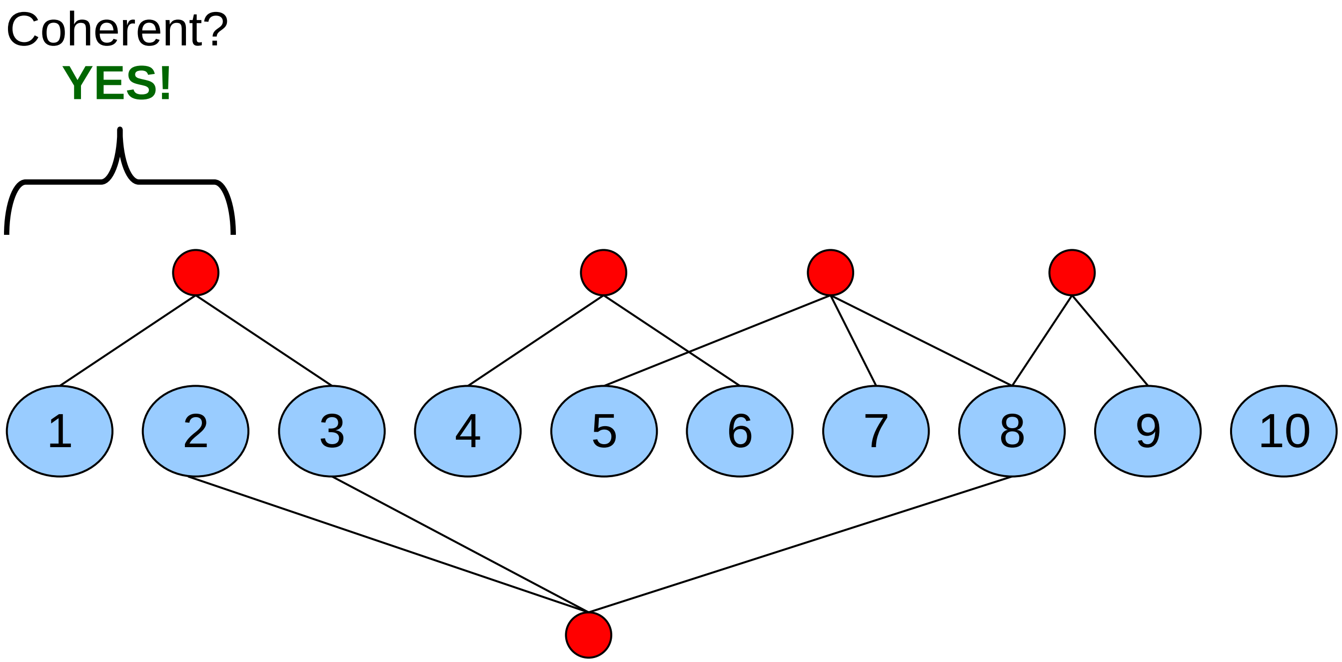
Local Optimal Diagnosis - BF



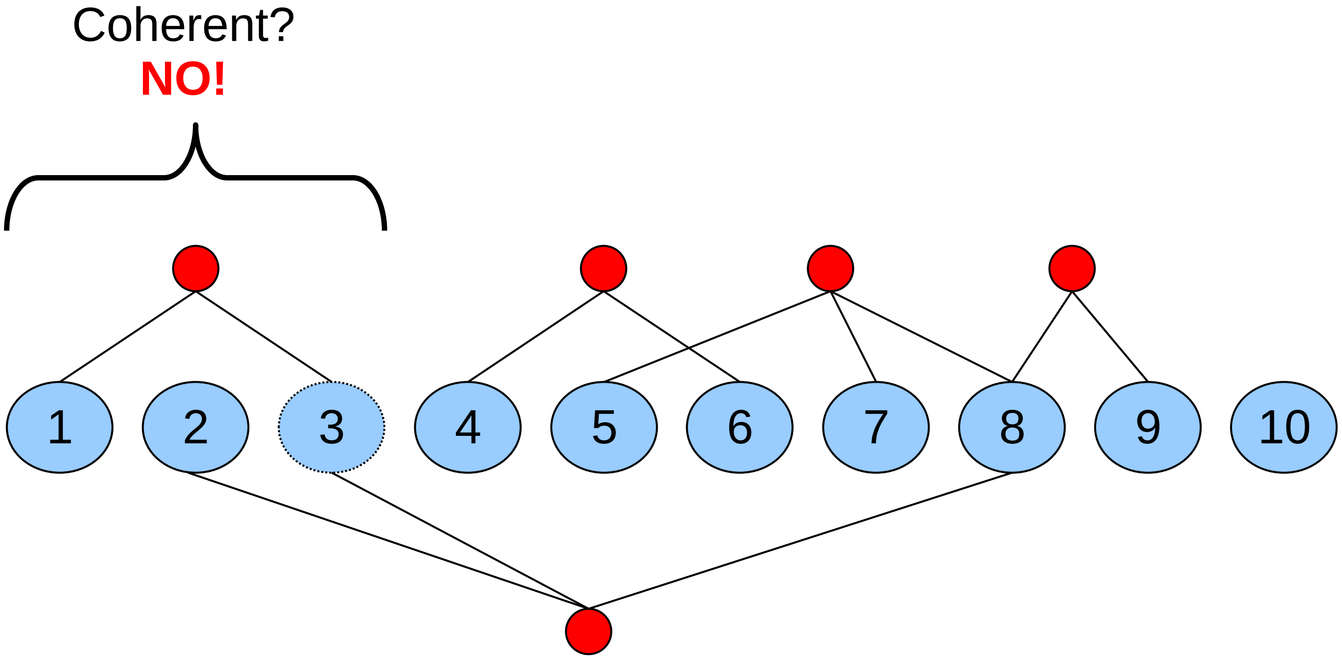
Local Optimal Diagnosis - BF



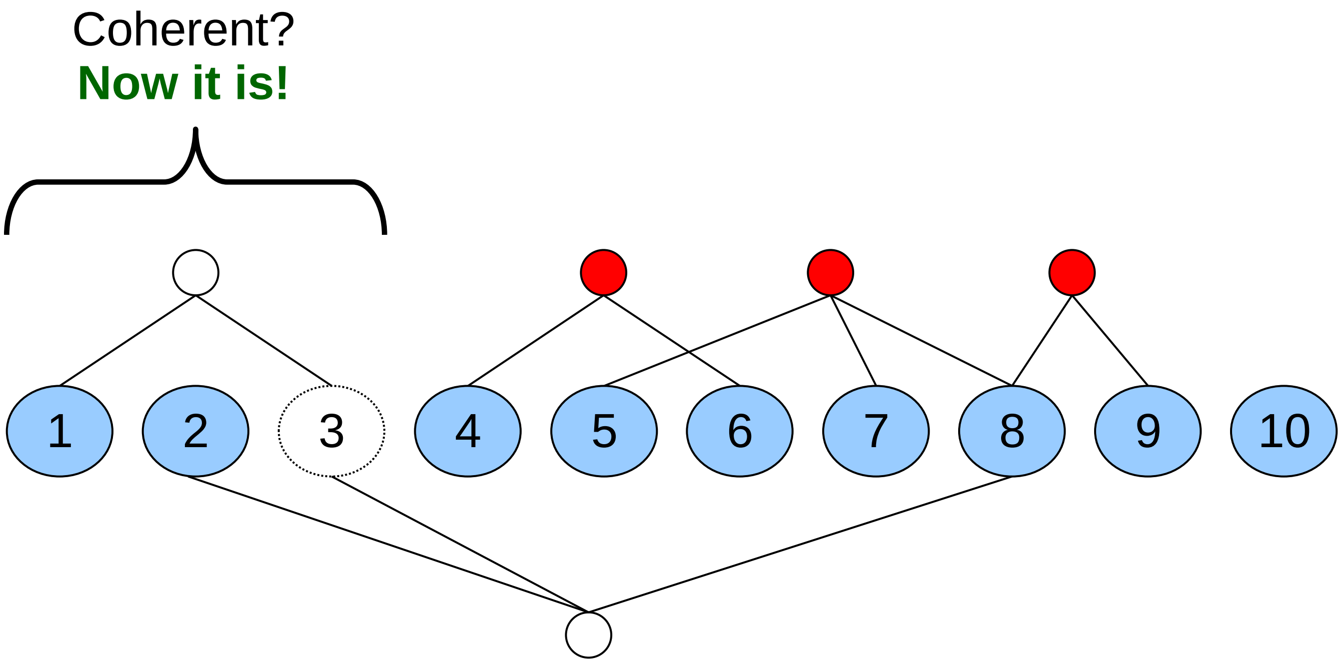
Local Optimal Diagnosis - BF



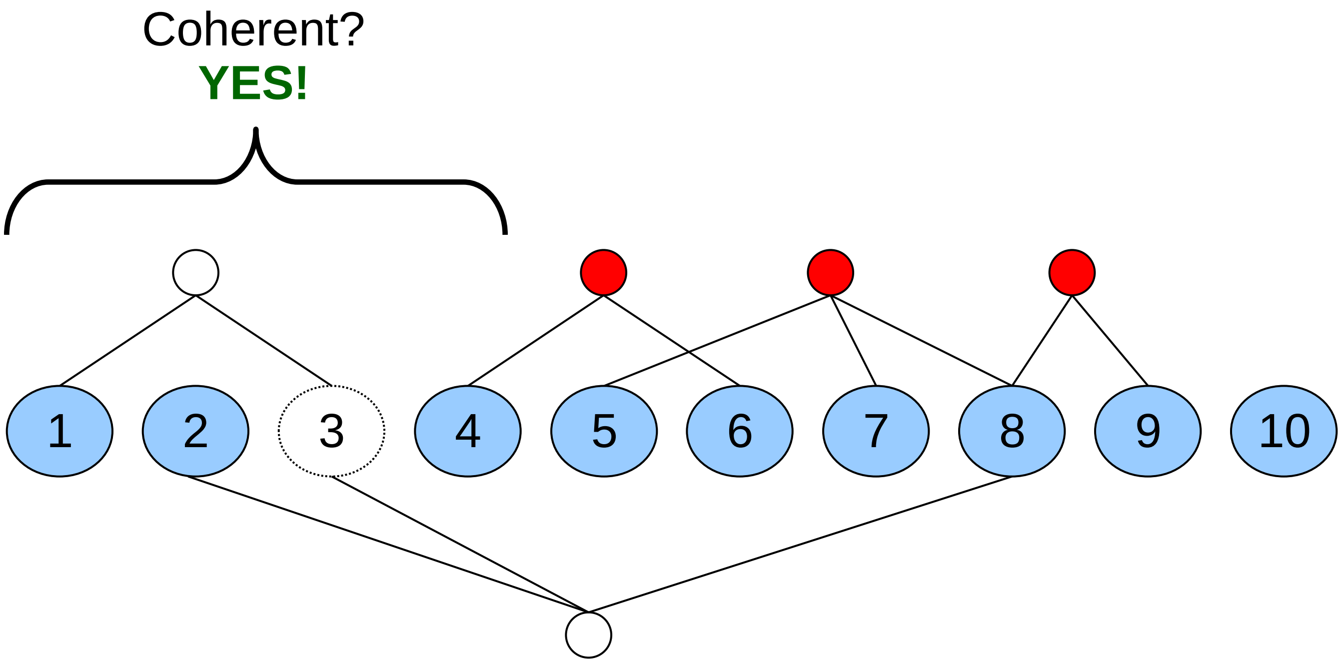
Local Optimal Diagnosis - BF



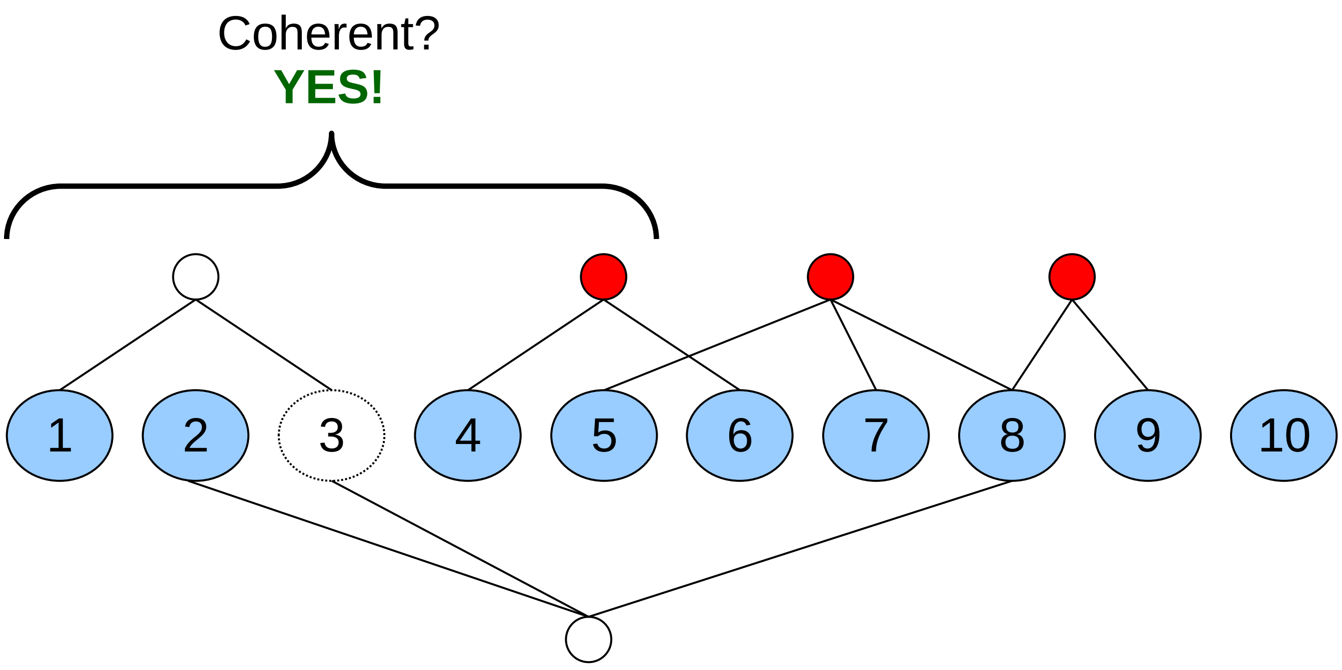
Local Optimal Diagnosis -BF



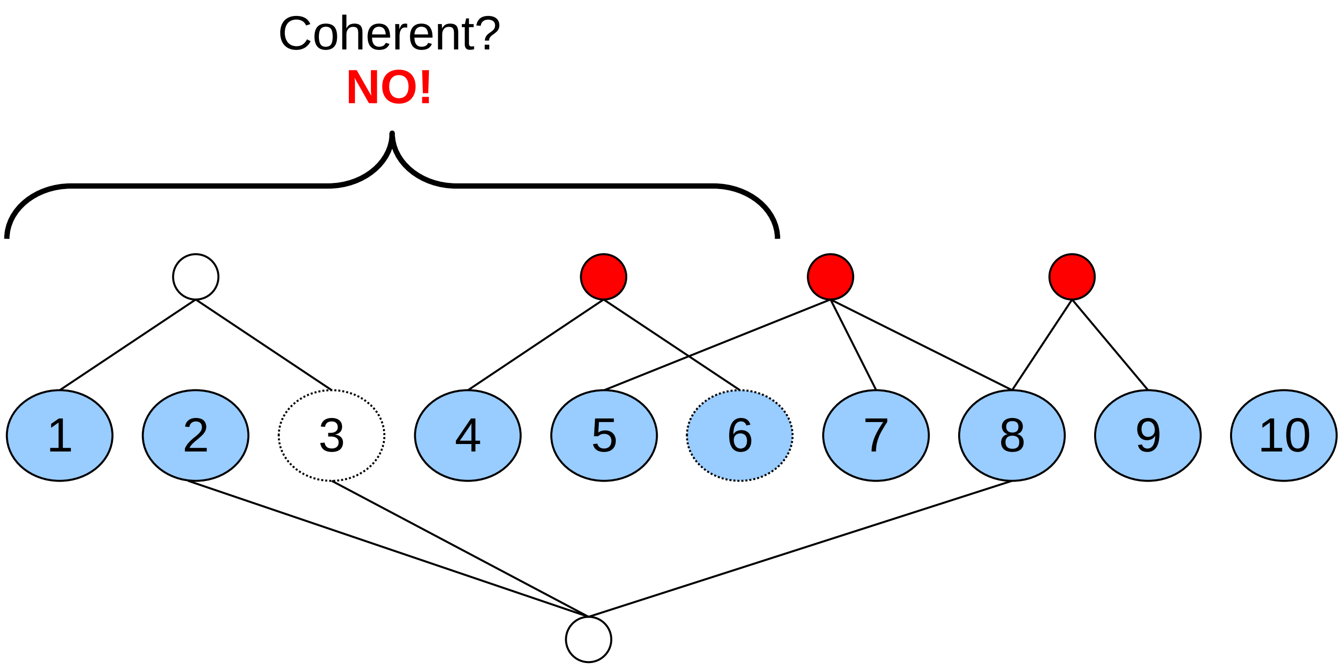
Local Optimal Diagnosis -BF



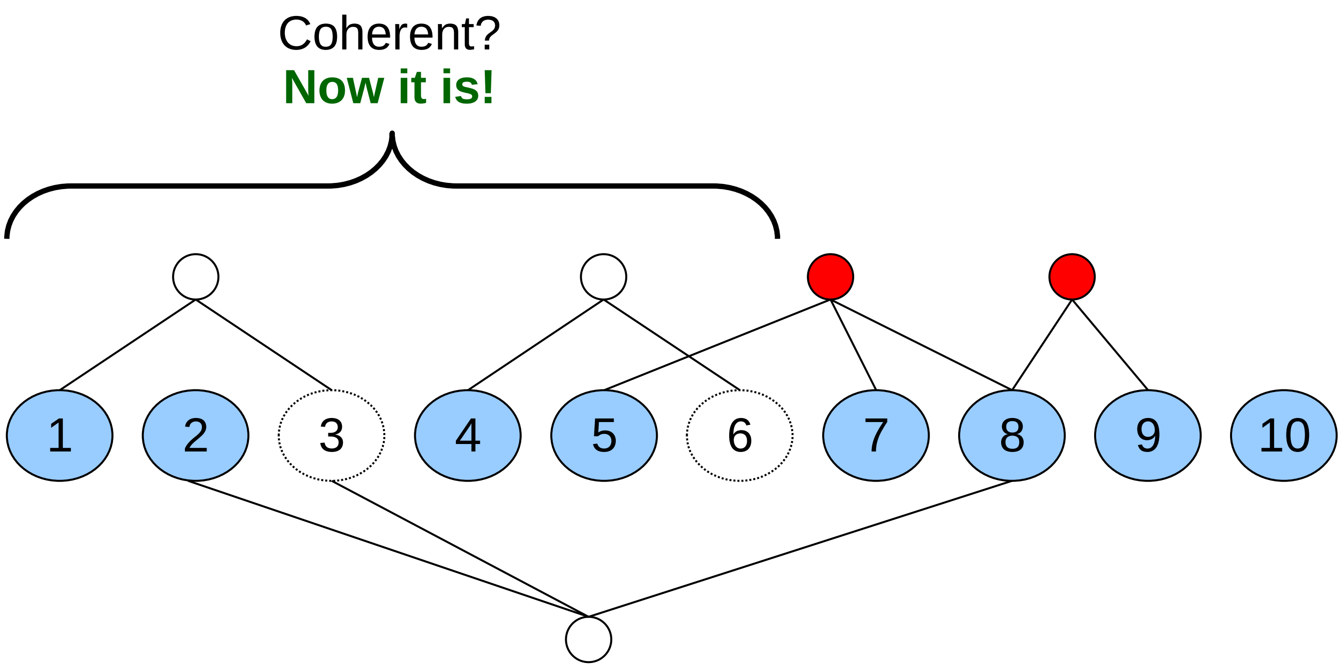
Local Optimal Diagnosis - BF



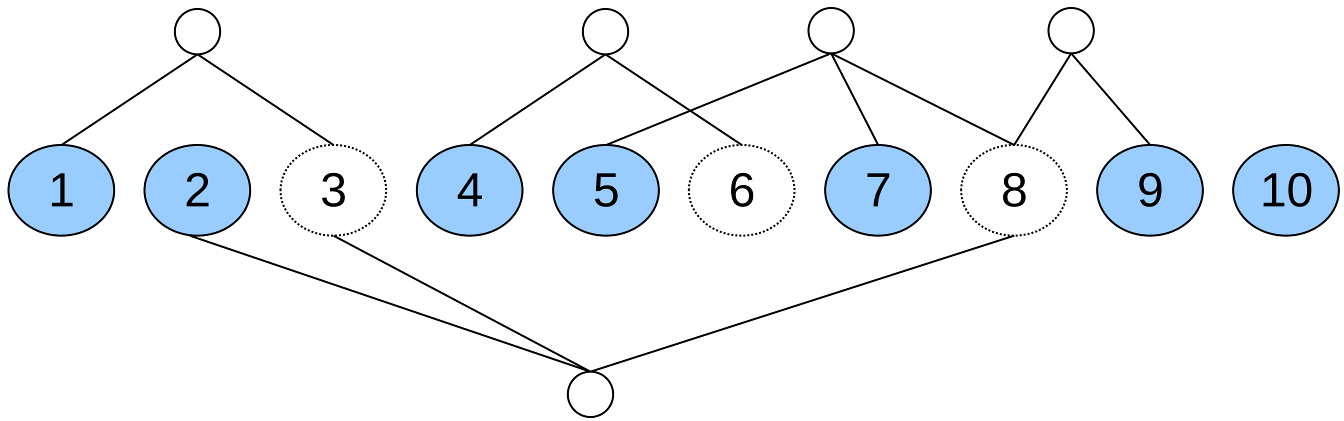
Local Optimal Diagnosis - BF



Local Optimal Diagnosis - BF



Local Optimal Diagnosis - BF



… final result is the Local Optimal Diagnosis {3,6,8}

Improvements

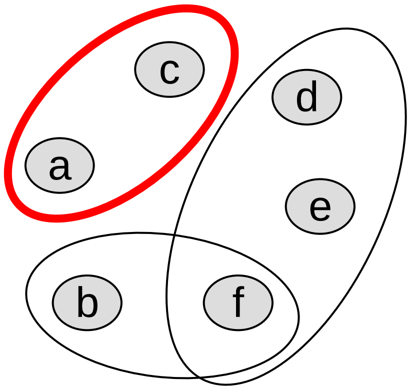
- Only simple brute force variant presented here
- Speed up with pattern-based reasoning in:
  - Meilicke, Stuckenschmidt. An efficient method for computing alignment diagnoses. RR-2009.

- Main Idea: Use pattern-based reasoning as above, check correctness afterwards and fix MIPS that have been missed out

Outline

- Background & Motivating Example
- Preliminaries/Theory
- **Algorithms**
  - Reasoning Components
  - Local Optimal Diagnosis
  - **Global Optimal Diagnosis**
- Implemented System / Availability
- Experimental Results
- User Support

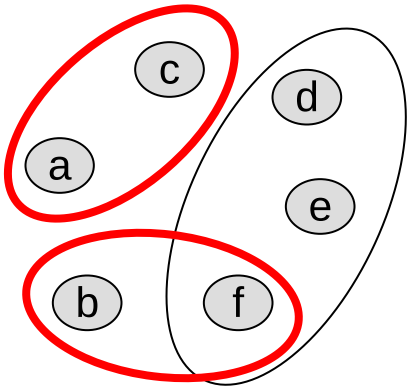
Uniform-Cost-Search (Bruteforce)

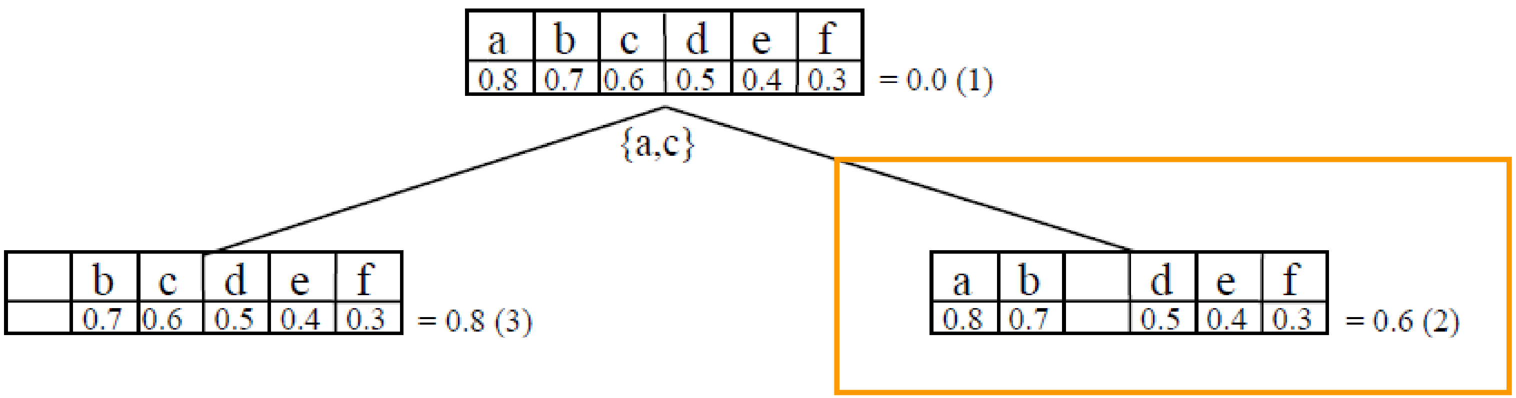


a	b	c	d	e	f
0.8	0.7	0.6	0.5	0.4	0.3

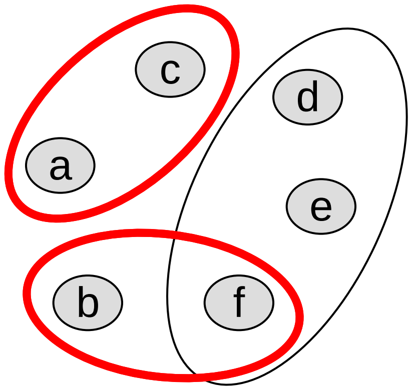
= 0.0 (1)

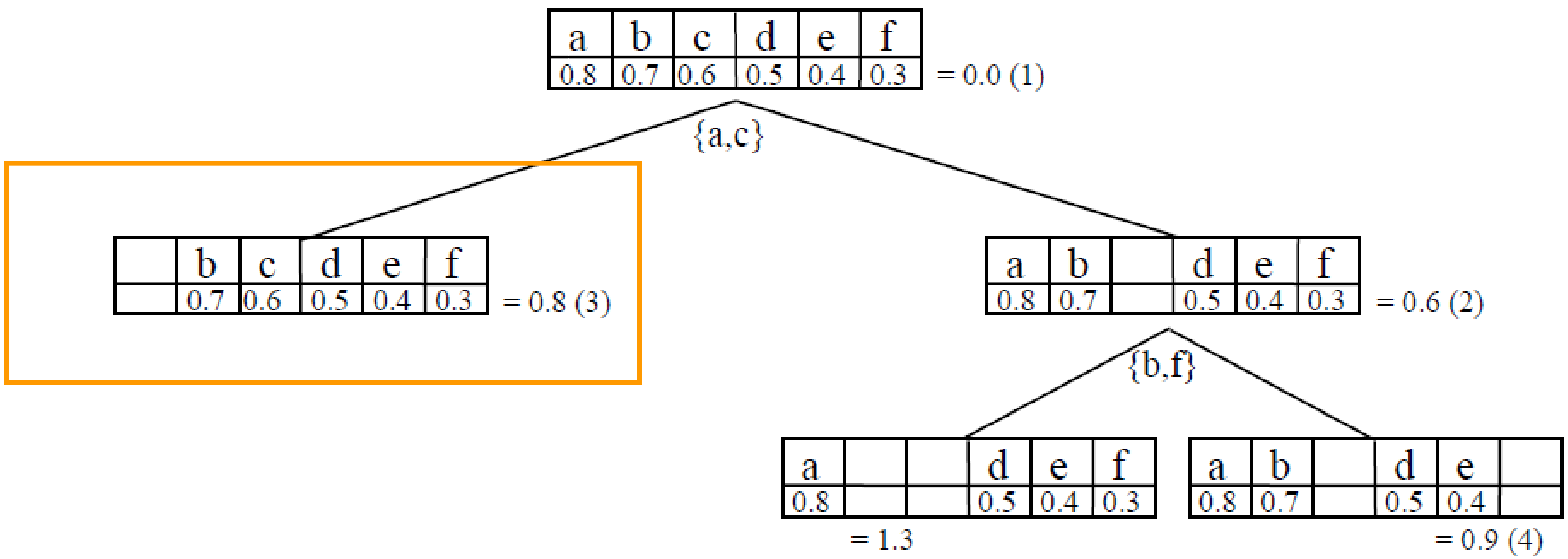
Uniform-Cost-Search (Bruteforce)



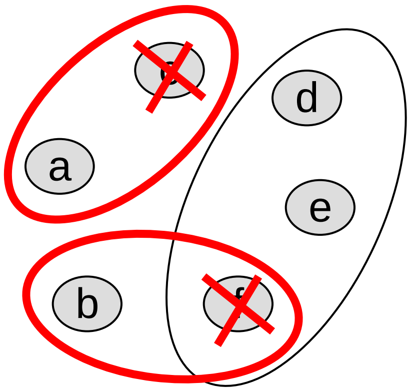


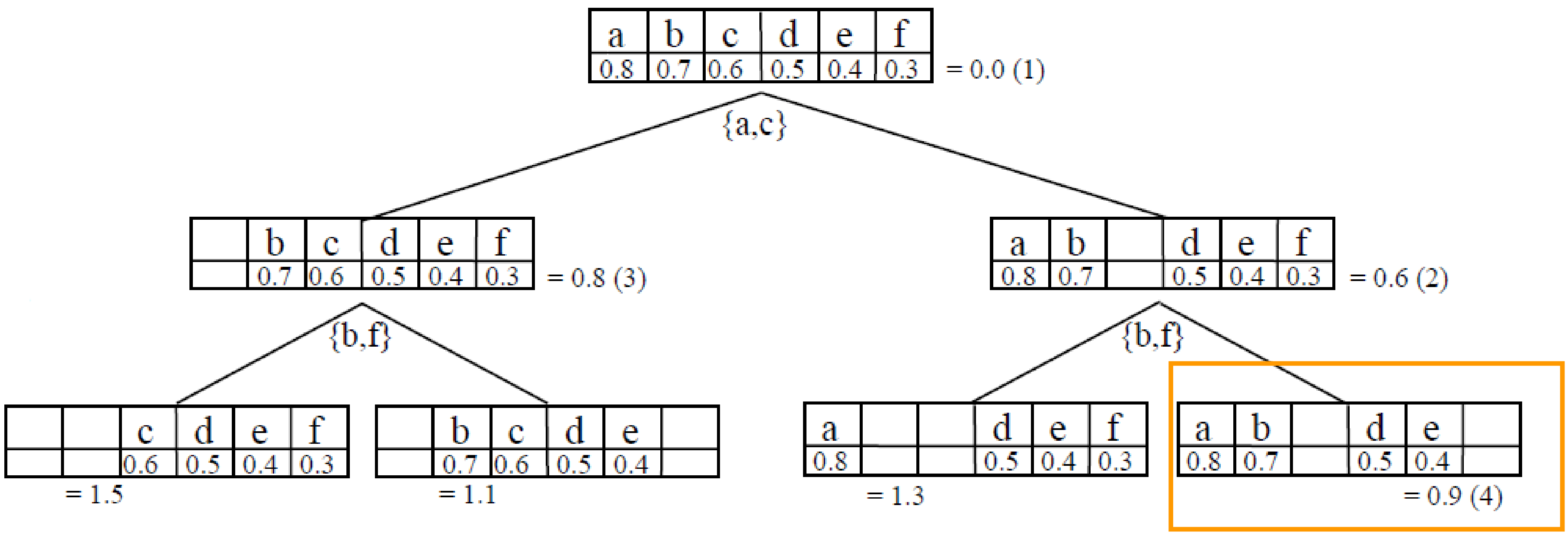
Uniform-Cost-Search (Bruteforce)





Uniform-Cost-Search (Bruteforce)

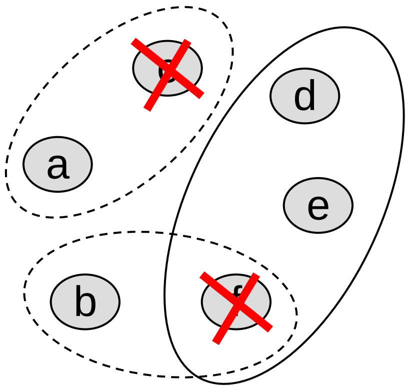


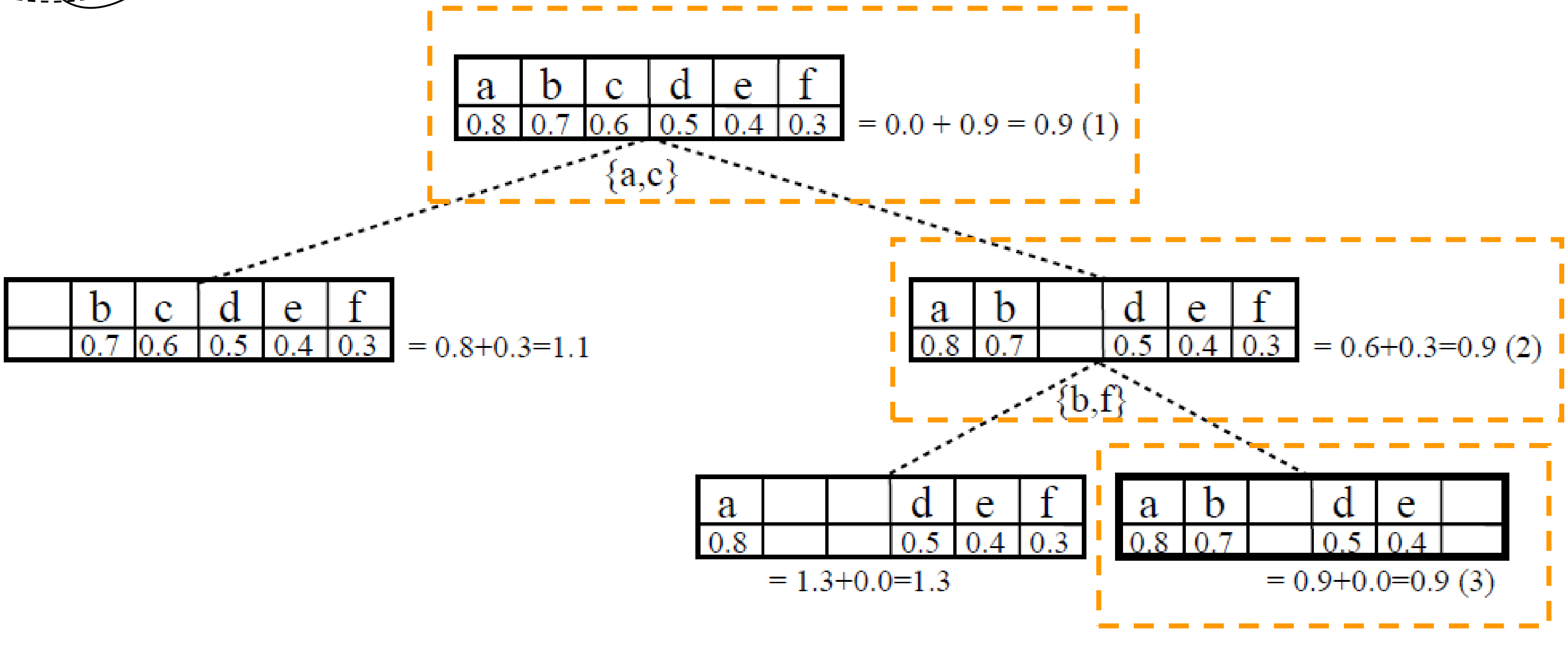


Patternbased Reasoning

- Idea: Use incomplete method for incoherence detection for pairs of correspondences in preprocessing step
- Use found MIPS for branching
- Use fullfledged reasoning only, when all previously found MIPS are resolved

A*-Search -Patternbased Reasoning





Outline

- Background & Motivating Example
- Preliminaries/Theory
- Algorithms
  - Reasoning Components
  - Local Optimal Diagnosis
  - Global Optimal Diagnosis
- **Implemented System / Availability**
- Experimental Results
- User Support

Short Demo

- http://web.informatik.uni-mannheim.de/alcomo/

Outline

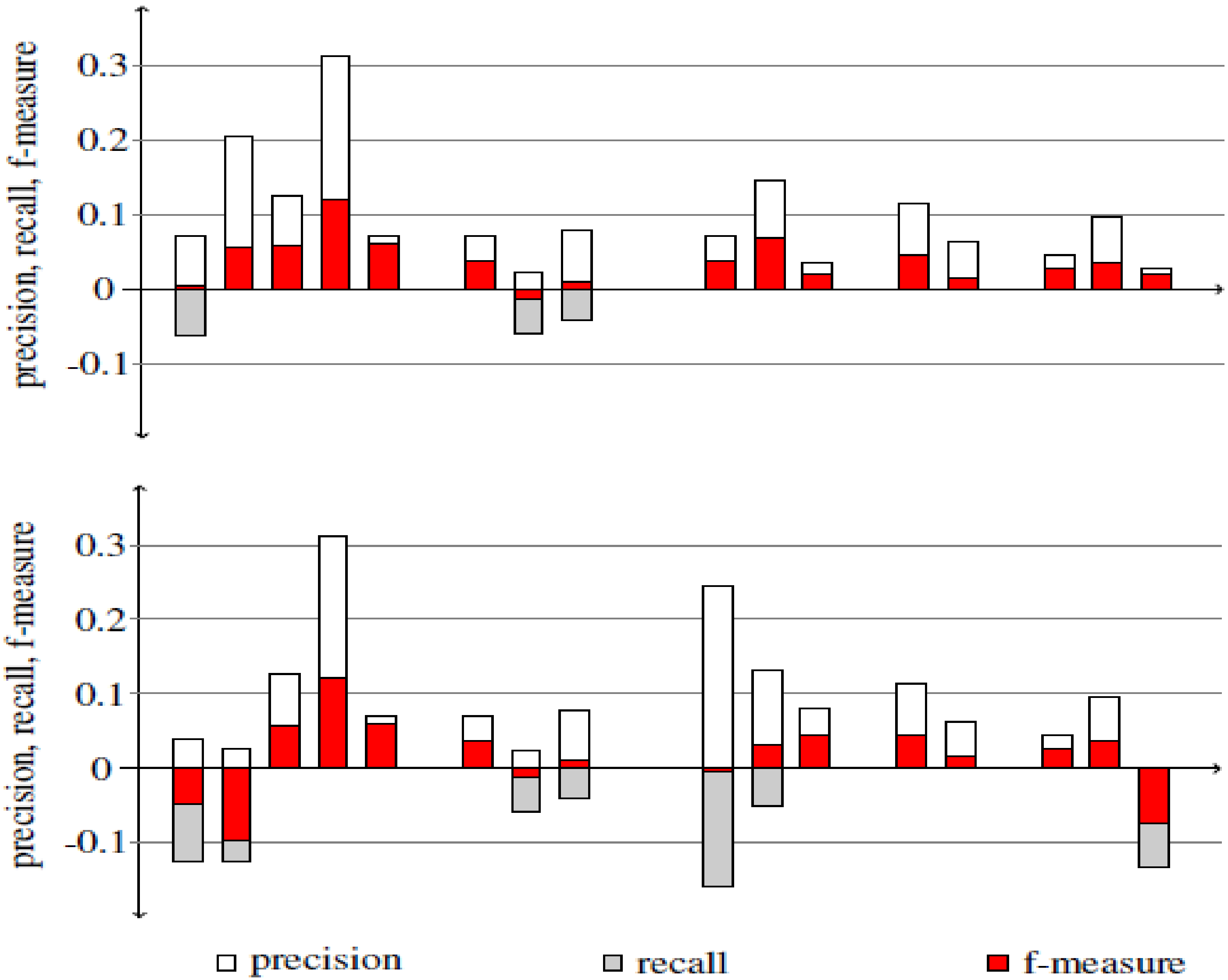
- Background & Motivating Example
- Preliminaries/Theory
- Algorithms
  - Reasoning Components
  - Local Optimal Diagnosis
  - Global Optimal Diagnosis
- Implemented System / Availability
- **Experimental Results**
- User Support

OAEI

- Ontology Alignment Evaluation Initiative
- Yearly campaign that offeres several evaluation tracks for ontology matching systems
- Takes place tomorrow together with OM workshop

- Rich source for testcases
  - Ontologie pairs and reference alignments
  - Automtically generated alignments

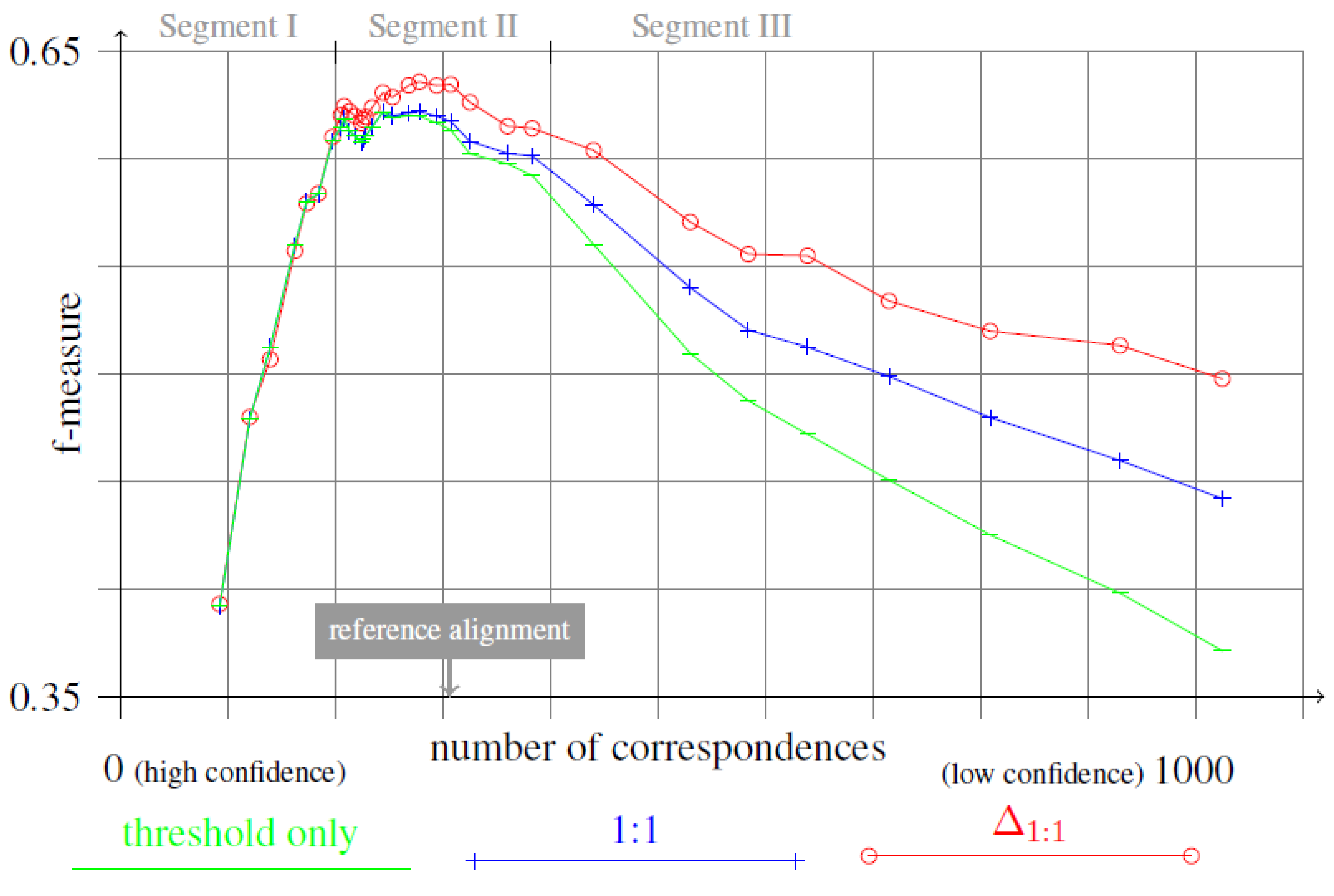
Global vs. Local



Aggregated Results - Global

	Input			Repaired			Comparison		
Matcher	*pre*	*f*	*rec*	*pre*	*f*	*rec*	*pre*	*f*	*rec*
$AgrMaker_{10}$	0.493	0.559	0.647	0.55	0.58	0.614	+0.057	+0.021	-0.033
$ASMOV_{10}$	0.348	0.469	0.719	0.381	0.496	0.709	+0.033	+0.027	-0.01
$Ef2Match_{10}$	0.487	0.549	0.627	0.53	0.565	0.605	+0.043	+0.016	-0.022
$Falcon_{10}$	0.583	0.578	0.572	0.659	0.607	0.562	+0.076	+0.029	-0.01
$GeRMeSMB_{10}$	0.328	0.397	0.503	0.352	0.402	0.467	+0.024	+0.005	-0.036
$SOBOM_{10}$	0.282	0.384	0.603	0.337	0.412	0.531	+0.055	+0.028	-0.072
$AgrMaker_{09}$	0.404	0.478	0.585	0.484	0.513	0.546	+0.08	+0.035	-0.039
$AgrMakerE_{09}$	0.282	0.381	0.585	0.316	0.384	0.49	+0.034	+0.003	-0.095
$Aroma_{09}$	0.352	0.409	0.487	0.411	0.435	0.461	+0.059	+0.026	-0.026
$ASMOV_{09}$	0.374	0.392	0.412	0.382	0.396	0.412	+0.008	+0.004	+/-0
$ASMOV_{08}$	0.312	0.379	0.484	0.344	0.393	0.458	+0.032	+0.014	-0.026
$Lily_{08}$	0.406	0.457	0.523	0.443	0.464	0.487	+0.037	+0.007	-0.036
Average	0.388	0.453	0.562	0.432	0.471	0.528	+0.044	+0.018	-0.034

Extracting Coherent Alignments



Outline

- Background & Motivating Example
- Preliminaries/Theory
- Algorithms
  - Reasoning Components
  - Local Optimal Diagnosis
  - Global Optimal Diagnosis
- Implemented System / Availability
- Experimental Results
- **User Support**

Motivation

- Automatically generated alignments are not perfect
- Manually generating alignments takes lots of efforts

- Idea: Automatically generate an (maybe imprecise) alignment with high recall and revise it manually!

- Can alignment incoherence be used to support the process?

Main Idea

- User makes decision given list of correspondences
  - User can ACCEPT a correspondence
  - or REJECT a correspondence.
- (Manually) undecided correspondences are
  - Involved in some conflict
  - Not involved in some conflict
  - Have to be rejected because of an ACCEPT

User Support: Demo

- http://web.informatik.uni-mannheim.de/alcomo/revision/asmov/

- http://web.informatik.uni-mannheim.de/alcomo/revision/dssim/

Other Approaches

- Previous approach was minimalistic
  - Reduce overload for the user

- Alternative approach: ContentMap
  - Jimenez-Ruiz, et al.: Ontology integration using mappings: Towards getting the right logical consequences.
  - Can show all justifications (MIPS) and repair plans (diagnoses)

References

- Christian Meilicke: Alignment Incoherence in Ontology Matching. Thesis, University of Mannheim, Chair of Artificial Intelligence, 2011.
- Guilin Qi, Qiu Ji, Peter Haase: A Conflict-Based Operator for Mapping Revision. In Proceedings of the 8th International Semantic Web Conference (ISWC'09), 521-536, 2009.

- Related Topics at ISWC
  - On Monday: OM-Workshop
  - On Thursday 14:00: Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau: LogMap: Logic-based and Scalable Ontology Matching

Debugging the missing is-a structure of taxonomies

Outline

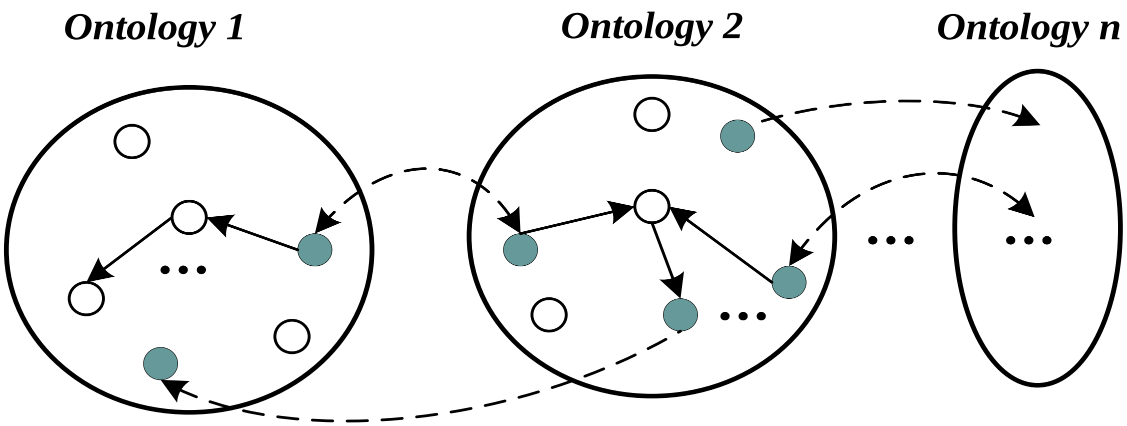
- Background
- Definitions
- Debugging approach
- Implemented system
- Experiments
- Future Work

Outline

- **Background**
- Definitions
- Debugging approach
- Implemented system
- Experiments
- Future Work

Taxonomy networks

A **taxonomy network** consists of a set of **taxonomies** and sets of **mappings** between these taxonomies.



Defects in ontologies

- Syntactic defects
  - eg. wrong tags or incorrect format
- Semantic defects
  - eg. unsatisfiable concepts or inconsistent ontologies
- **Modeling defects**
  - **eg. wrong or missing relations**

  → Solution requires domain knowledge.

Missing is-a relations

- In 2008 Ontology Alignment Evaluation Initiative (OAEI) Anatomy track, task 4
  - Ontology MA : Adult Mouse Anatomy Dictionary (2744 concepts)
  - Ontology NCI-A : NCI Thesaurus - anatomy (3304 concepts)
  - Partial reference alignment between them (988 mappings)
    - 121 missing is-a relations in MA
    - 83 missing is-a relations in NCI-A

Influence of missing structure

- Ontology-based querying.

Medical Subject Headings (MeSH)

All MeSH Categories
- **Diseases Category**
  - **Eye Diseases**
    - **Scleral Diseases**
      - **Scleritis**

...

return 1363 articles

Influence of missing structure

- Incomplete results from ontology-based queries

Medical Subject Headings (MeSH)

All MeSH Categories

- **Diseases Category**
  - **Eye Diseases**
    - **Scleral Diseases**
      - **Scleritis**

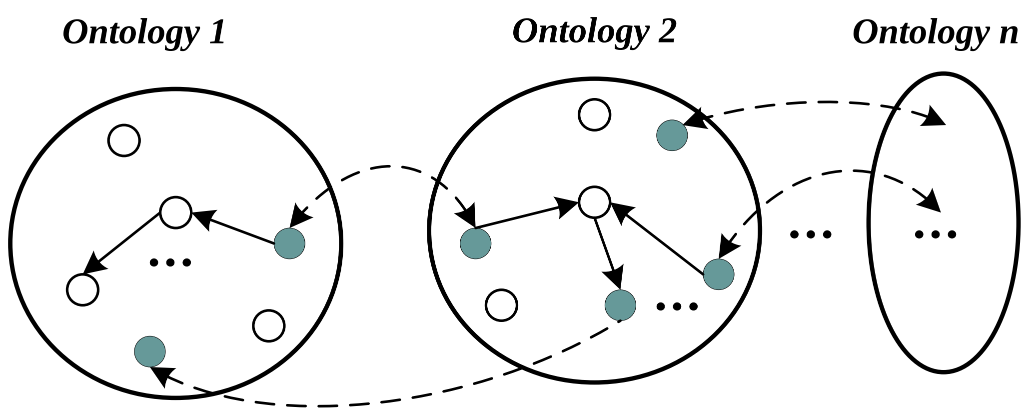
...

return 1363 articles

return 613 articles

55% results are missed !

Assumptions and scope



- We focus on **taxonomies**,

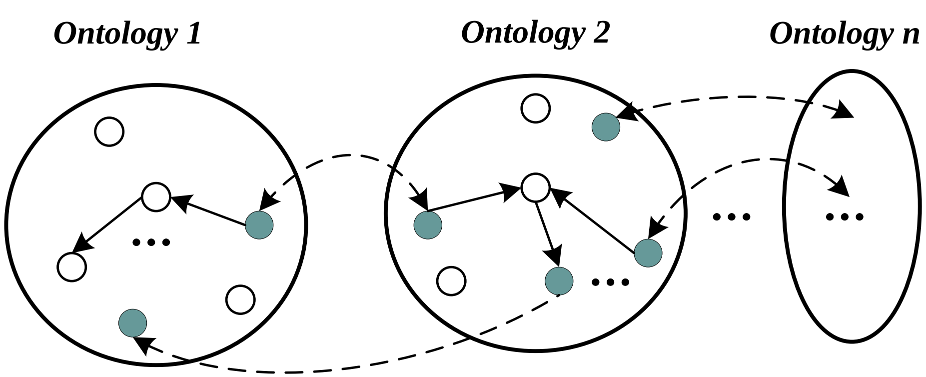
  → *named concepts* and *is-a relations*.
- We assume that **all the existing structure in the taxonomies is correct**.

Assumptions and scope

- We assume that all **the existing mappings** in the taxonomy network are **correct**.
- The mappings represent equivalence and subsumption.

> ***Partial Reference Alignment (PRA)*** *– is a set of correct mappings between two ontologies.*

- The existing correct mappings are called **PRA mappings.**
- Concepts in PRA mappings are called **PRA concepts.**



Debugging missing is-a structure in taxonomy networks

Given a set of taxonomies networked by sets of **correct** mappings, how to **detect and repair the missing is-a relations in these networked taxonomies**?

Outline

- Background
- **Definitions**
- Debugging approach
- Implemented system
- Experiments
- Future Work

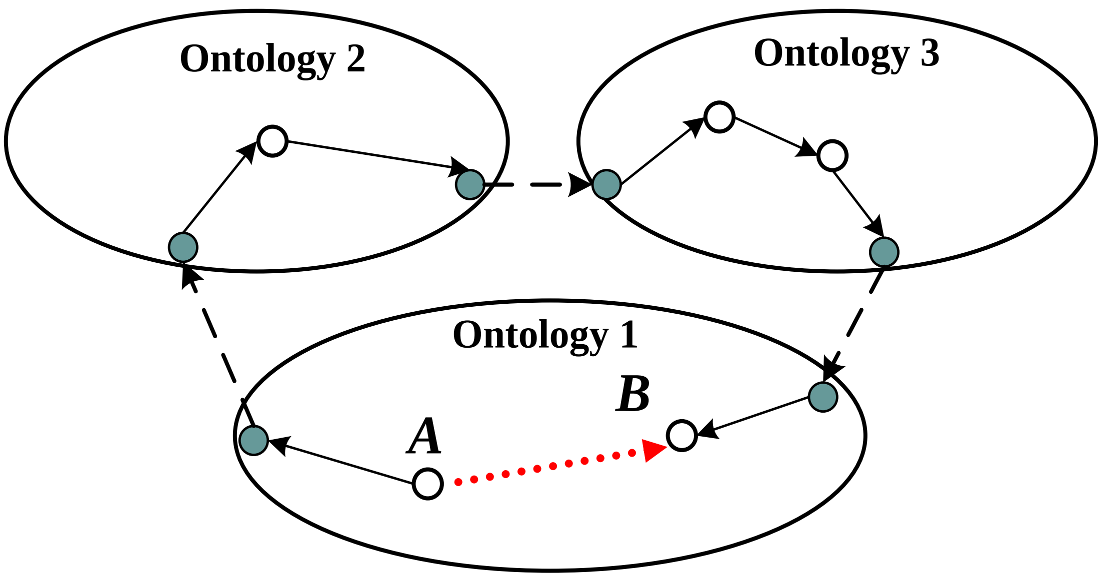
Detecting missing is-a relations

- Using external knowledge
  - Ontology learning
  - Discovery of subsumption relations (Hearst patterns)

- Using knowledge inherent in the network

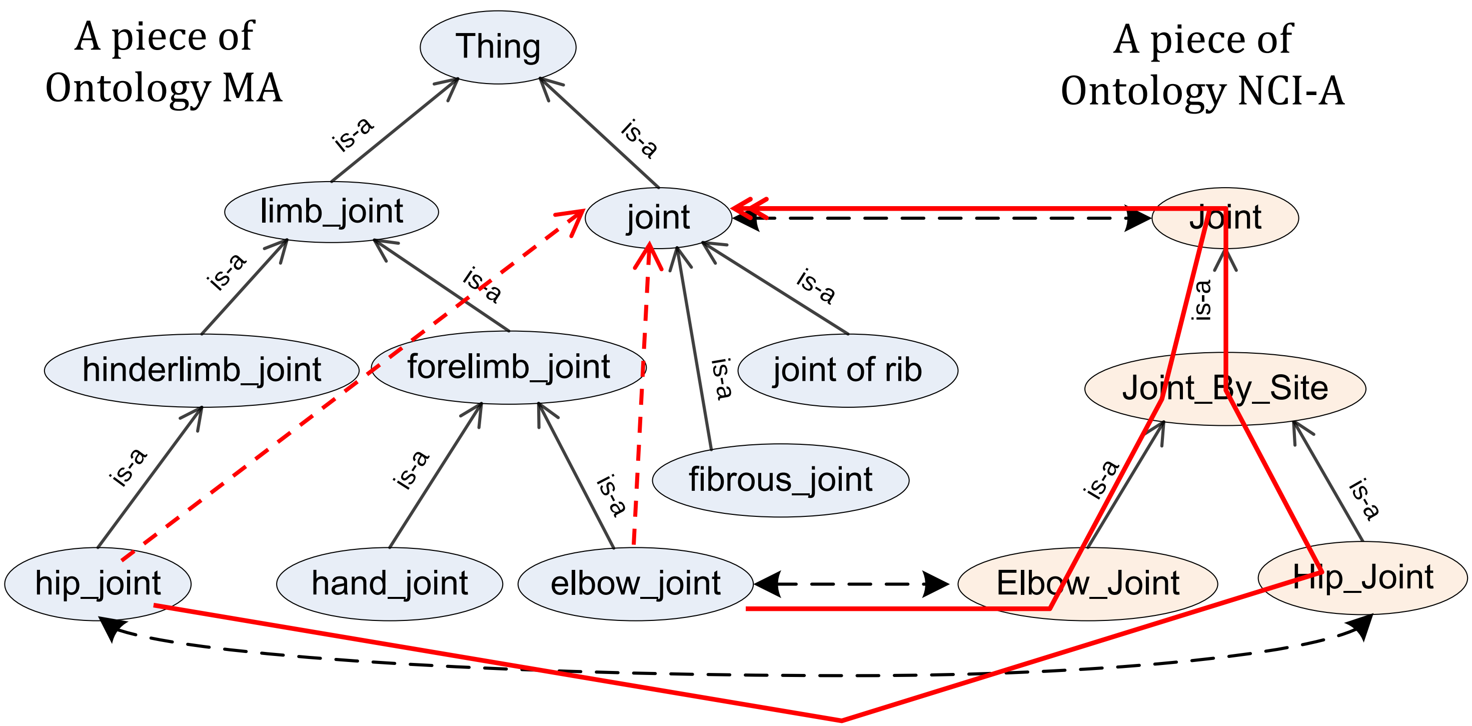
Missing is-a relations

Given two concepts A and B in a taxonomy O in the network. If “A is-a B” is **logically derivable from the taxonomy network,** but **not from the taxonomy O alone**, then “A is-a B” is a **missing is-a relation.**



Example of missing is-a relations

- Two small pieces of MA and NCI-A, both about concept "joint", and 3 equivalence mappings.



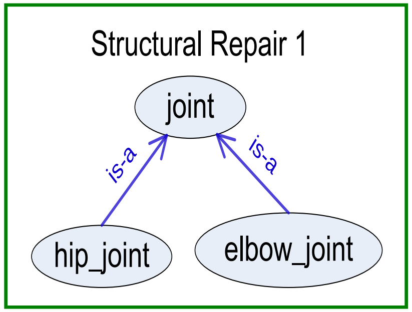
Repairing missing is-a relations

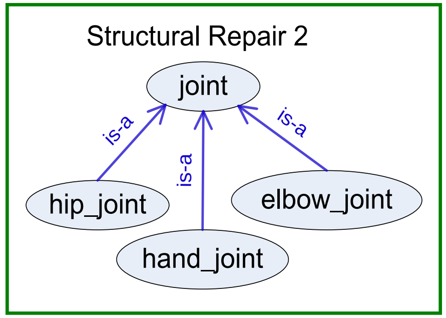
Repair the original taxonomies by adding a set of is-a relations (called **structural repair**) to each taxonomy, such that the missing is-a relations can be derived from the extended taxonomy.

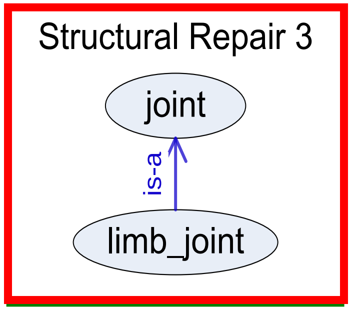
- Structural repair
  - The is-a relations within the structural repair are called ’repairing actions’.

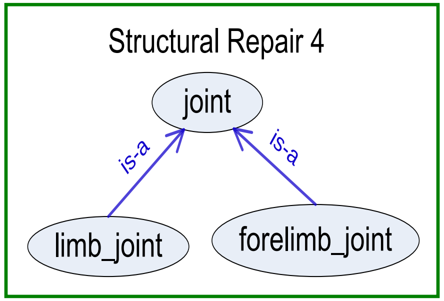
→ The set of missing is-a relations themselves is a structural repair, but it is not always the only nor the best choice.

Example





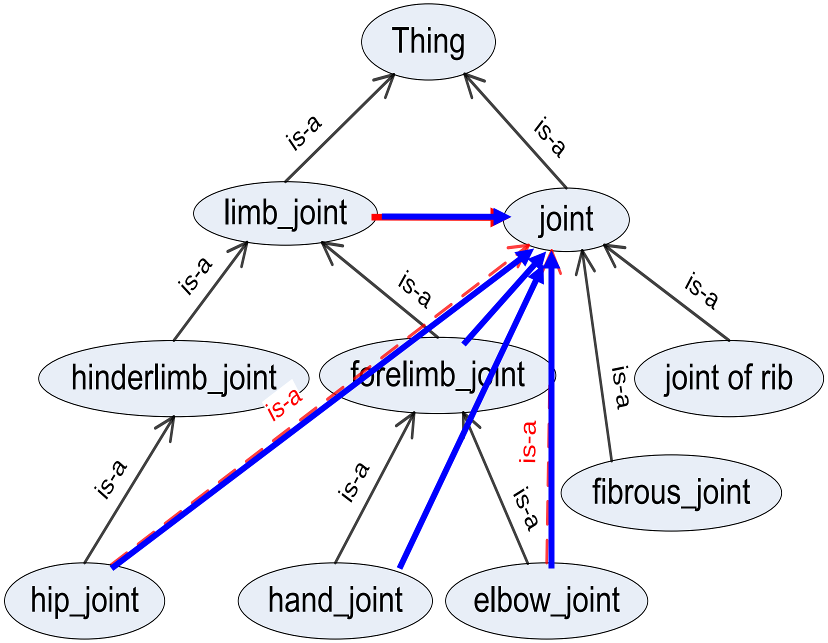




Question:

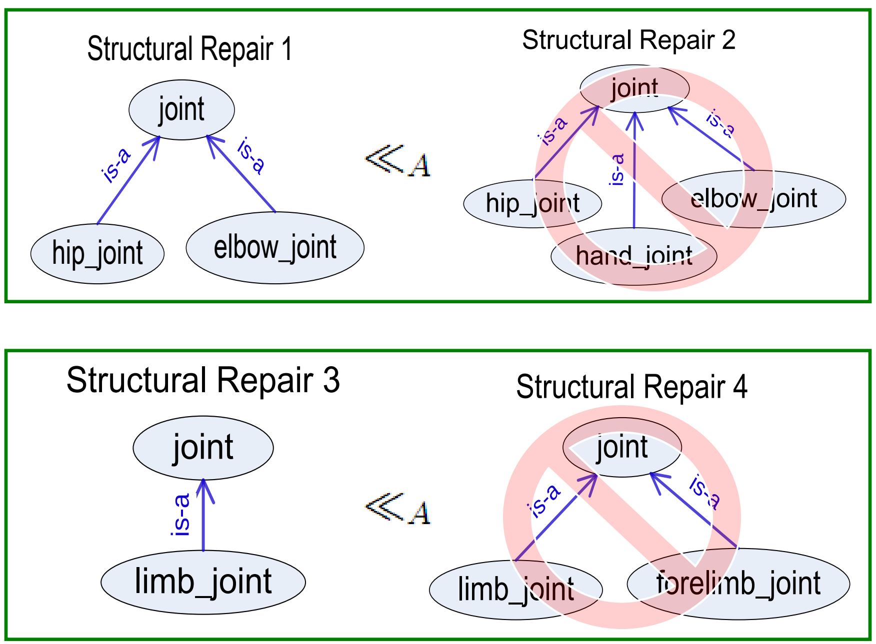
How can we recognize structural repairs that are interesting for a domain expert?

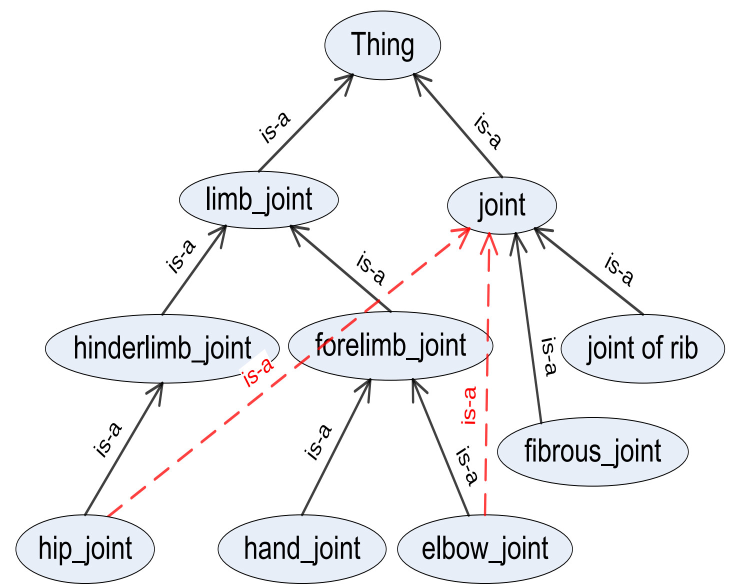
→ heuristics.



Axiom-based Heuristic

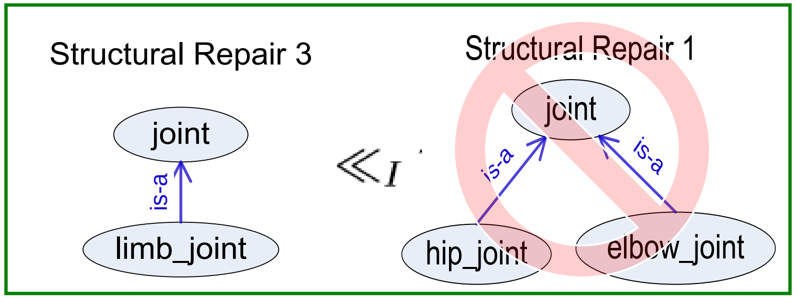
Prefer to use structural repair **without non-contributing** repairing actions.

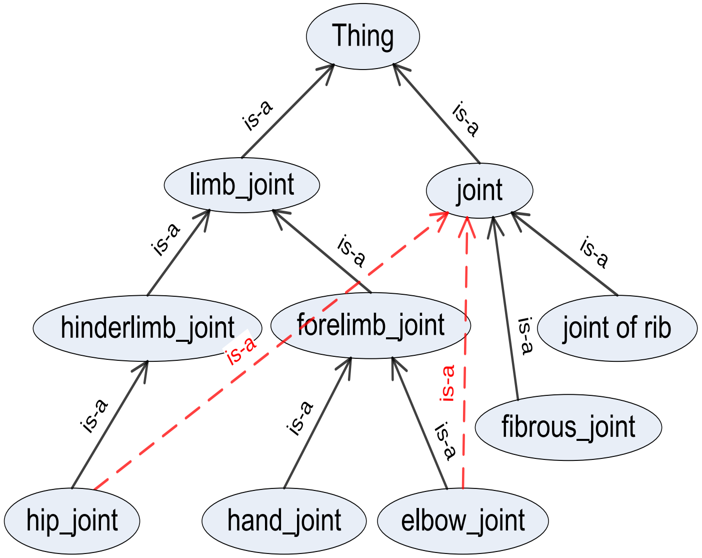




Information-based heuristic

Prefer to use structural repair with **more informative** repairing actions.

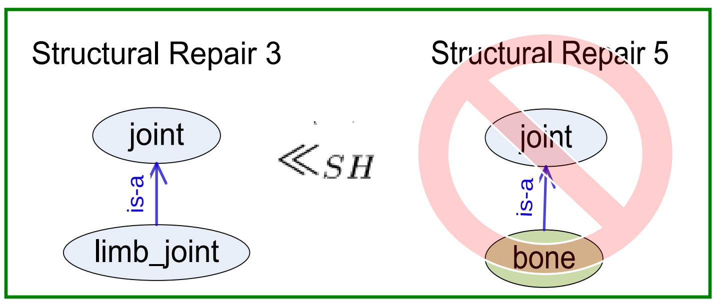




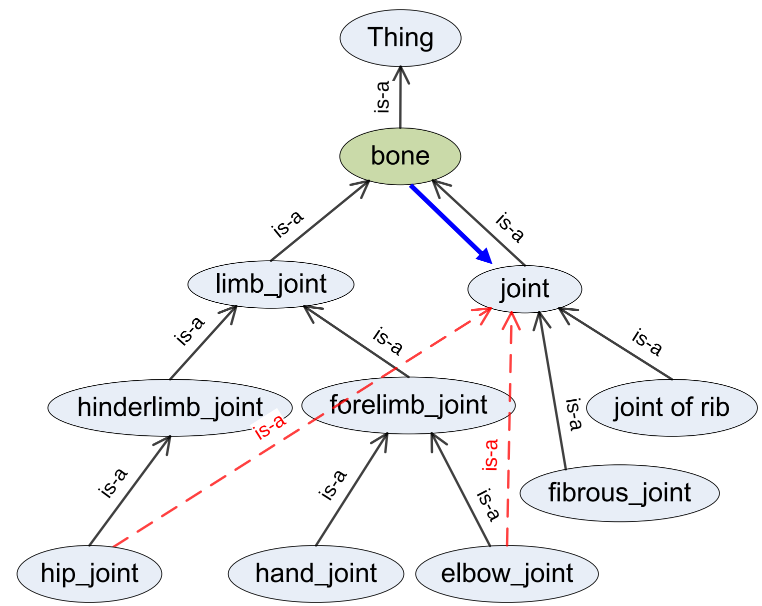
(limb_joint, joint) is more informative than **(hip_joint, joint)** and **(elbow_joint, joint)**

Strict hierarchy heuristic

Prefer to use structural repair which **does not change the existing is-a relations in the original ontology into equivalence relations.**



(bone, joint) will introduce an equivalence relation between ’**joint**’ and ’**bone**’.



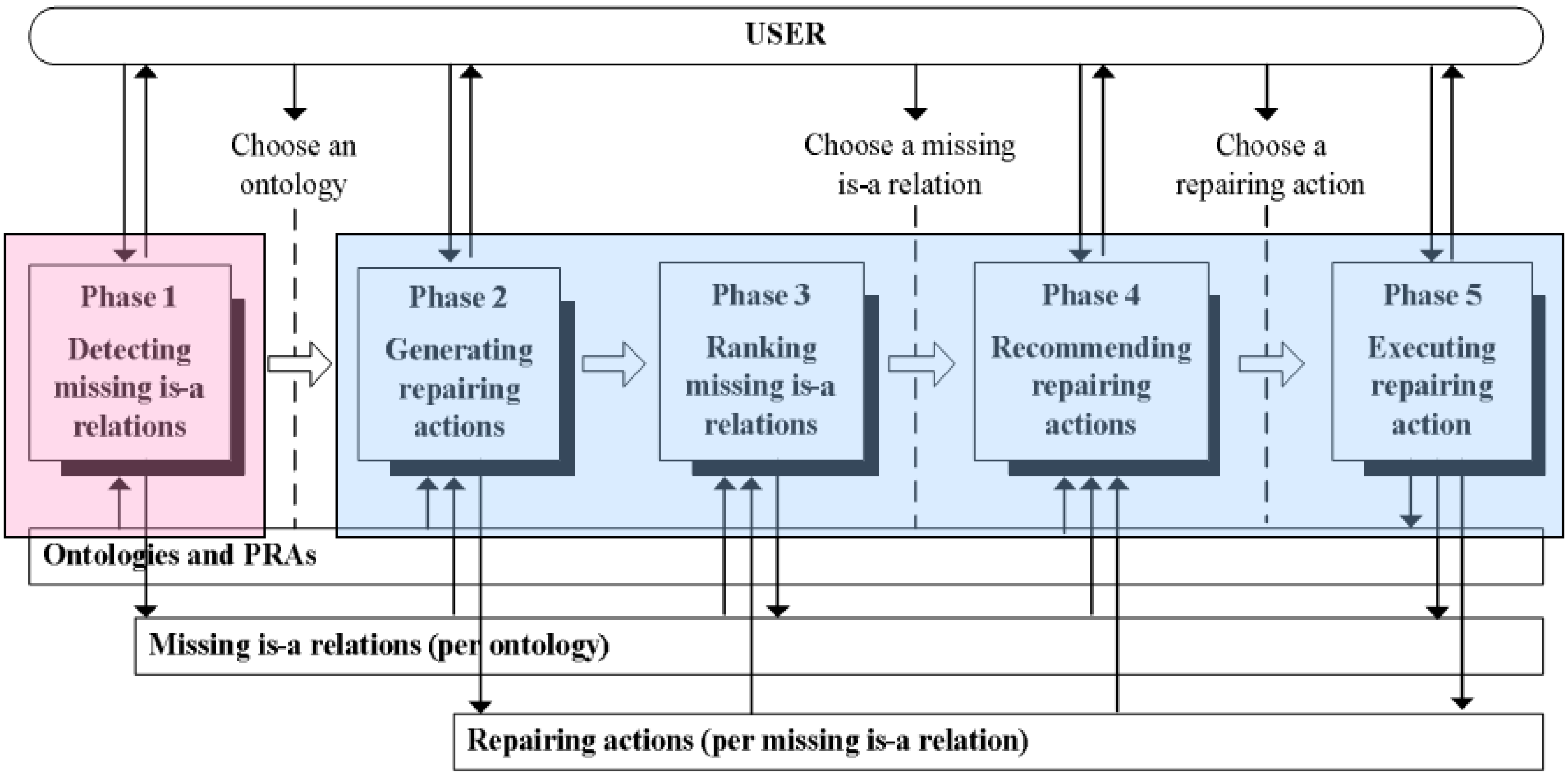
Single relations heuristic

- Assume that it is more likely that domain experts have missed a single relation than a chain of relations
  - *Assume it is more likely that*
    (ankle_joint, limb_joint)
    is missing than
    (ankle_joint, x_1) and (x_1,x_2), and … and (x_{k-1}, x_k)
    and (x_k, limb_joint).

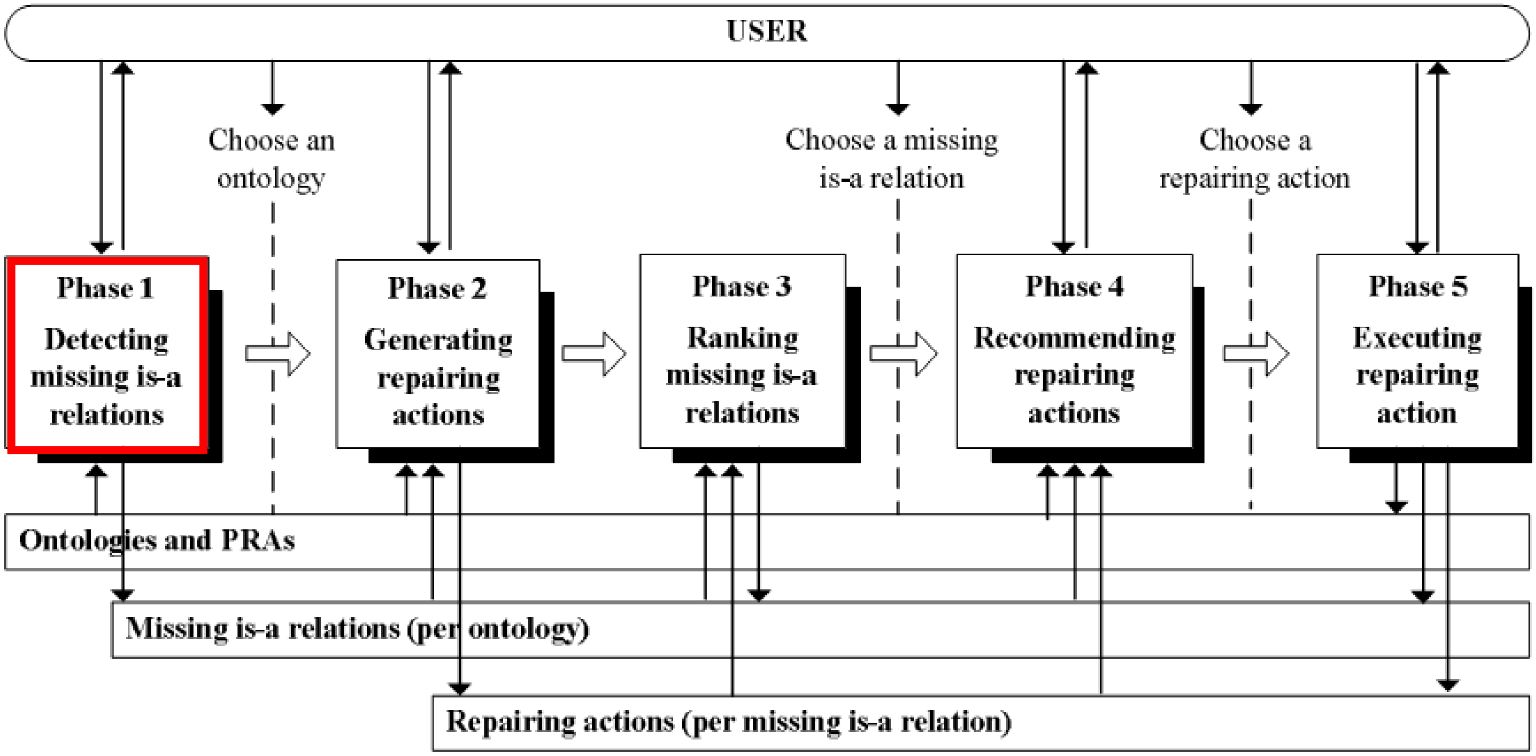
Outline

- Background
- Definitions
- **Debugging approach**
- Implemented system
- Experiments
- Future Work

Overview of debugging approach



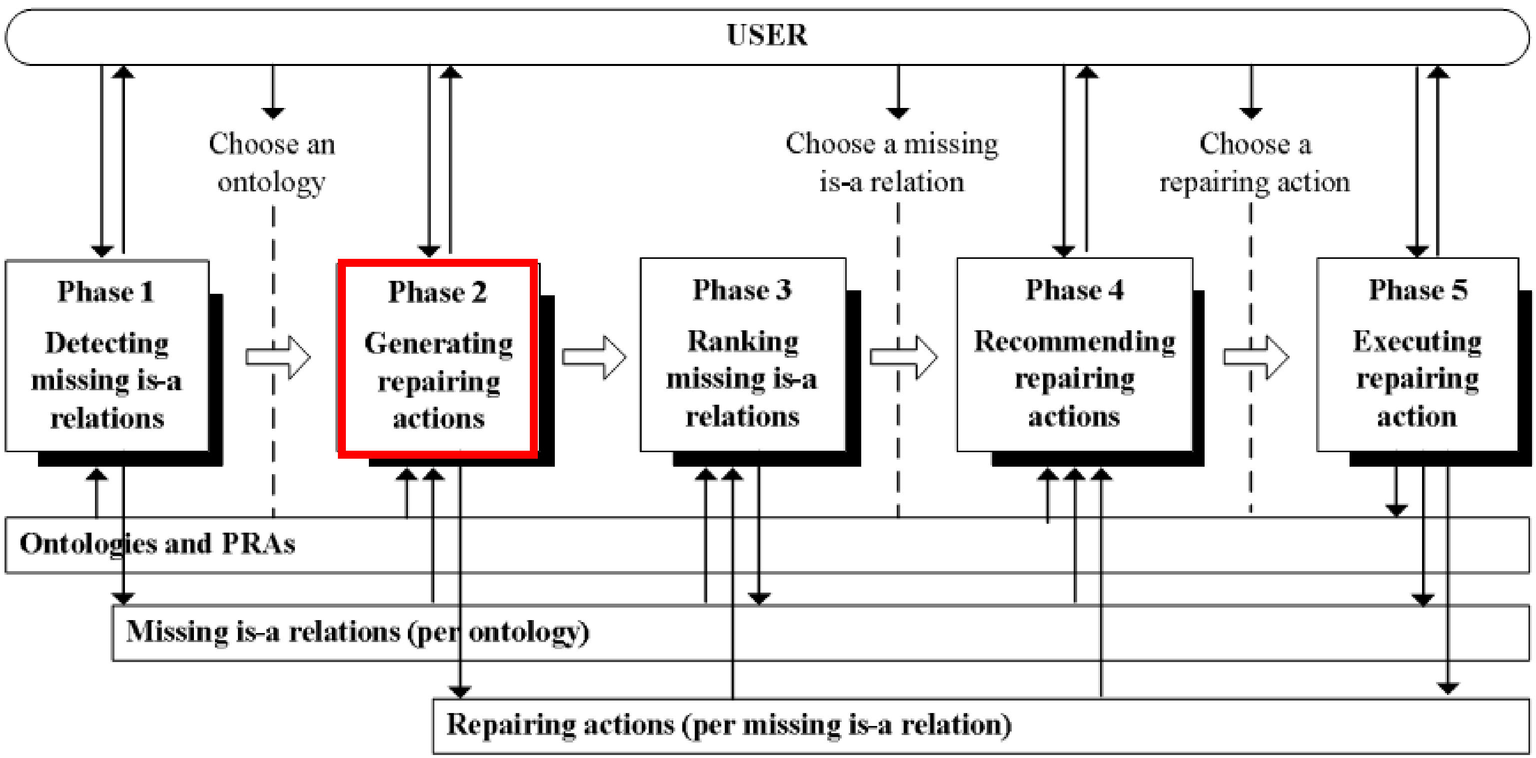
Phase 1.
Detecting missing is-a relations



Detecting missing is-a relations

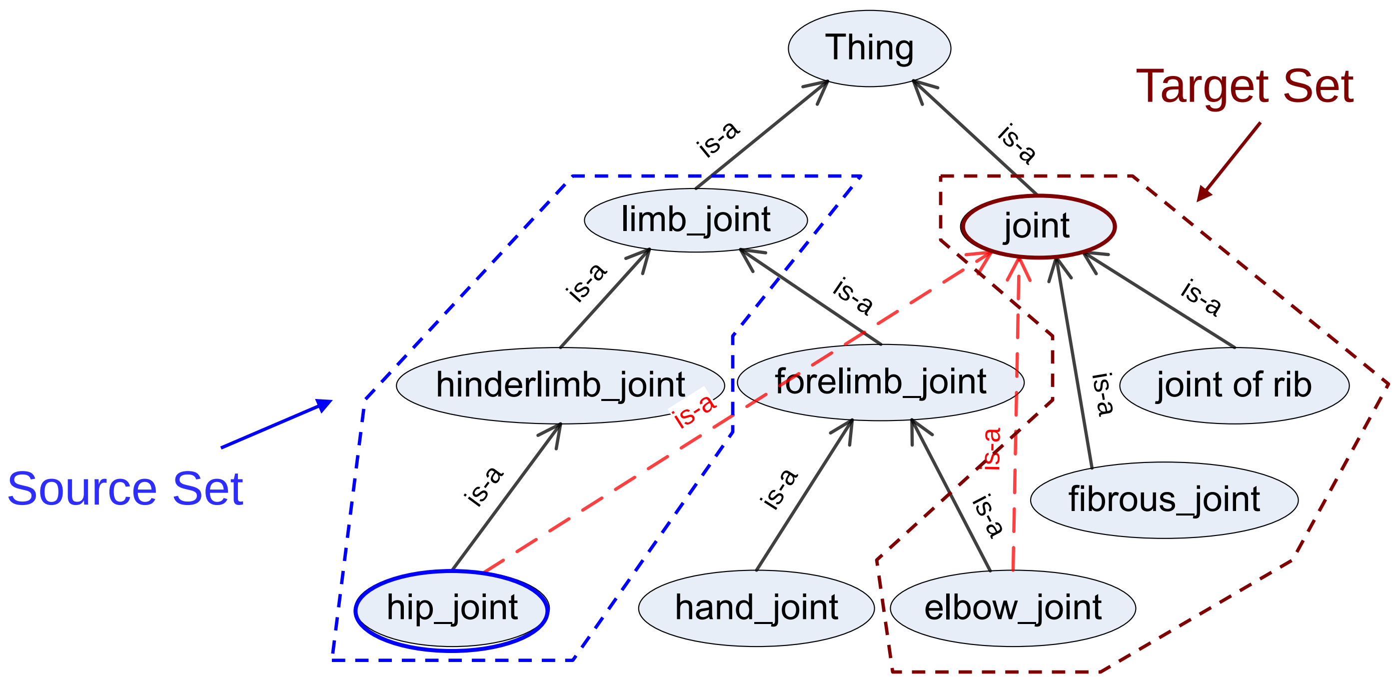
- Based on definition
- Only need to detect missing is-a relations between PRA concepts

Phase 2.
Generating repairing actions



Example

For missing is-a relation **(hip_joint, joint)**, we generate two sets of concepts representing 3×4 repairing actions using algorithm 1.



Algorithm 1 - basic algorithm

- Intuition
  - Given a set of missing is-a relations, find possible repairing actions taking into account that all missing is-a relations will be repaired.

Input
The ontology under repair O, its set of missing is-a relations M.
Output
Repairing actions.
Algorithm
1. Initialize KB with ontology;
2. For every missing is-a relation $(a,b) \in$ M: add the axiom $a \rightarrow b$ to the KB;
3. For each $(a,b) \in$ M:
   $Source(a,b)$:= super-concepts(a) − super-concepts(b);
   $Target(a,b)$:= sub-concepts(b) − sub-concepts(a);
4. Missing is-a relation (a,b) can be repaired by choosing an element from $Source(a,b) \times Target(a,b)$.

Figure 4.5: The basic algorithm for generating repairing actions.

Algorithm 1 - basic algorithm

- Conforms to the heuristics
  - For a repairing action **(s, t)** regarding missing is-a relation **(a, b)**, it is guaranteed that
    - since **a** $\rightarrow$ **s** and **t** $\rightarrow$ **b**
      - **(s, t)** is relevant for repairing **(a, b)** *Axiom-based heuristic*
      - **(s, t)** is at least as informative as **(a, b)** *Information-based heuristic*
    - **(a, t)** and **(s, b)** will not introduce equivalence relations, where in the original taxonomy we have only is-a relations *Strict-hierarchy heuristic*
    - For each missing is-a relation one is-a relation is selected for repairing *Single relation heuristic*

3. For each $(a,b) \in \mathrm{M}$:
   $Source(a,b) :=$ super-concepts(a) $-$ super-concepts(b);
   $Target(a,b) :=$ sub-concepts(b) $-$ sub-concepts(a);
4. Missing is-a relation (a,b) can be repaired by choosing an element from $Source(a,b) \times Target(a,b)$.

Algorithm 2 - extended algorithm

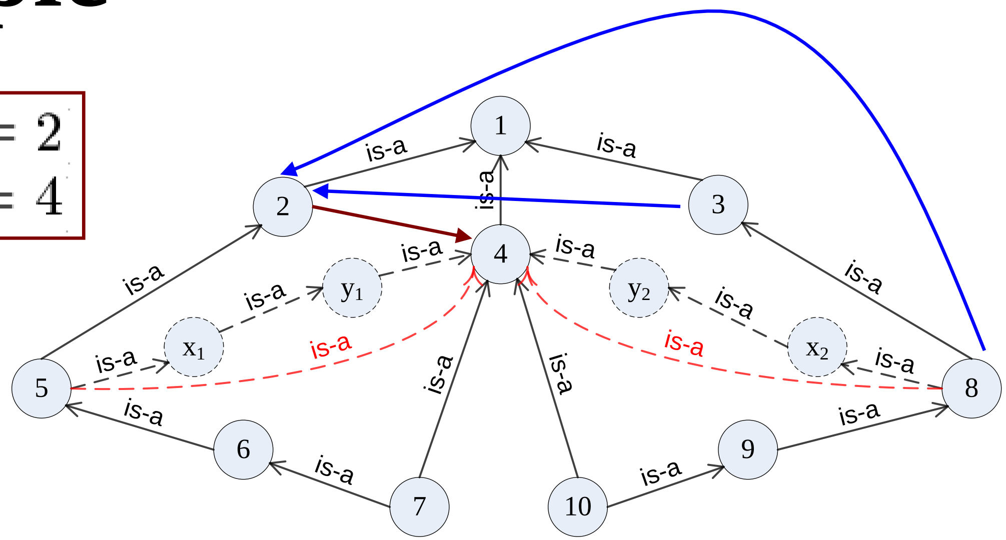
- Intuition:
  - Taking into account influence of other missing is-a relations that are common to all possible choices for repairing actions of other missing is-a relations.

Input
The ontology under repair O, its set of missing is-a relations M.
Output
Repairing actions.
Algorithm
1. Initialize KB with ontology ;
2. For every missing is-a relation $(a,b) \in$ M:
   Create two new concepts x and y in the KB;
   Add the axioms $a \rightarrow x$, $x \rightarrow y$, $y \rightarrow b$ to the KB;
3. For each $(a,b) \in$ M:
   $Source\text{-}ext(a,b)$:= super-concepts(a) $-$ super-concepts(x);
   $Target\text{-}ext(a,b)$:= sub-concepts(b) $-$ sub-concepts(y);
4. Missing is-a relation (a,b) can be repaired by choosing an original ontology element from $Source\text{-}ext(a,b)$ and an original ontology element from $Target\text{-}ext(a,b)$.

Figure 4.7: The extended algorithm for generating repairing actions.

Example

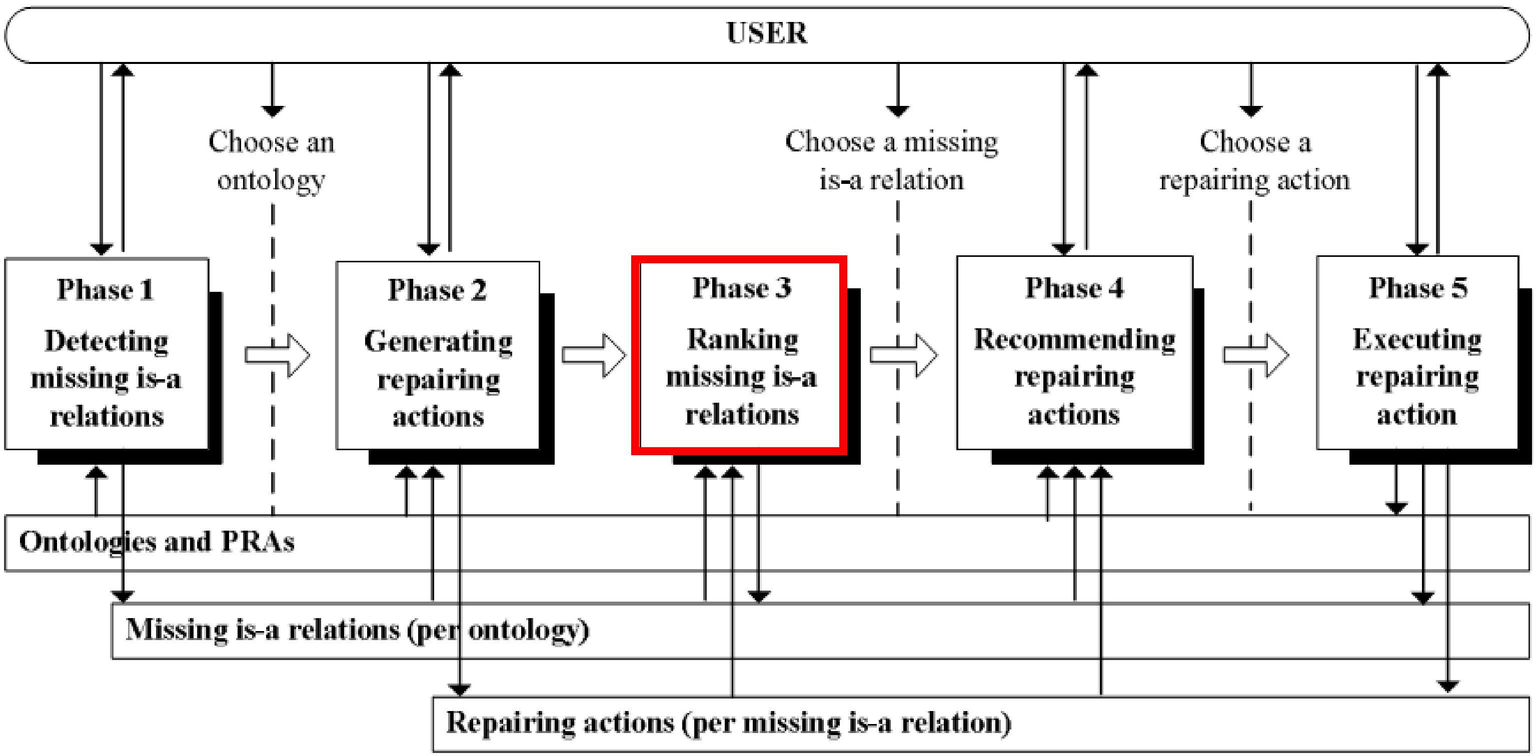
$$\boxed{\begin{aligned} x_1 &= 2 \\ y_1 &= 4 \end{aligned}}$$



$$\begin{aligned}
Source\text{-}ext(5,4) &= \{5,4,1,2,x_1,y_1\} - \{4,1,x_1,y_1\} = \{5,2\} \\
Target\text{-}ext(5,4) &= \{4,8,9,10,5,6,7,x_1,y_1,x_2,y_2\} - \{5,6,7,x_1,y_1\} \\
&= \{4,8,9,10,x_2,y_2\} \\
Source\text{-}ext(8,4) &= \{8,4,1,3,x_2,y_2\} - \{4,1,x_2,y_2\} = \{8,3\} \\
Target\text{-}ext(8,4) &= \{4,8,9,10,5,6,7,x_1,y_1,x_2,y_2\} - \{8,9,10,x_2,y_2\} \\
&= \{4,5,6,7,\boxed{2}\}
\end{aligned}$$

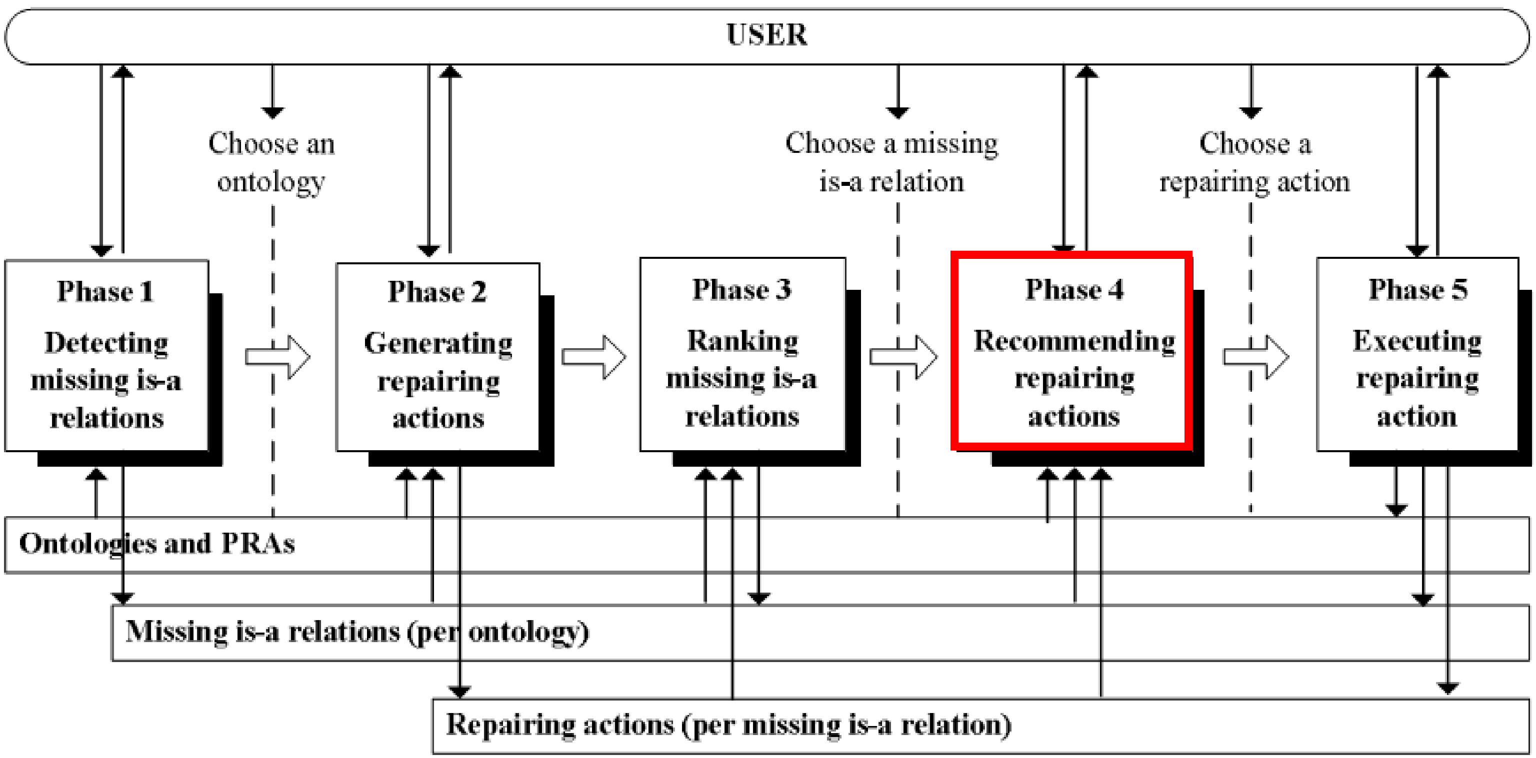
For instance, if we choose repairing action $(2,4)$ for missing is-a relation $(5,4)$, which means x_1 and y_1 will become equivalent to 2 and 4 respectively, the influence is that concept 2 will become a new element in $Target\text{-}ext(8,4)$

Phase 3.
Ranking missing is-a relations



- Rank the missing is-a relations with respect to the number of possible repairing actions.

Phase 4.
Recommending repairing actions



- Recommend repairing actions based on external domain knowledge, such as WordNet and UMLS.

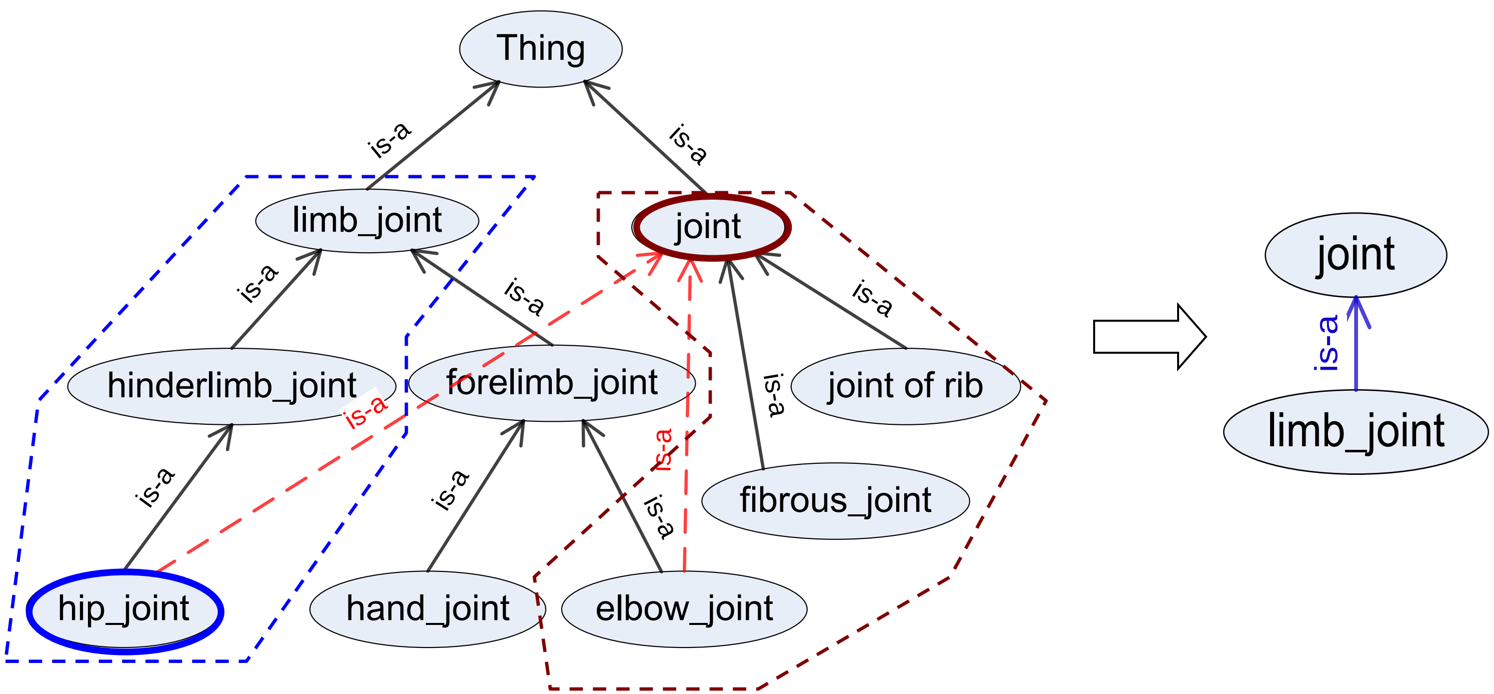
Recommendation algorithm

- We assume that we can query the external domain knowledge regarding subsumption of concepts
  - General thesauri
    - e.g. WordNet
  - Specialized domain-specific sources
    - e.g. UMLS (Unified Medical Language System)

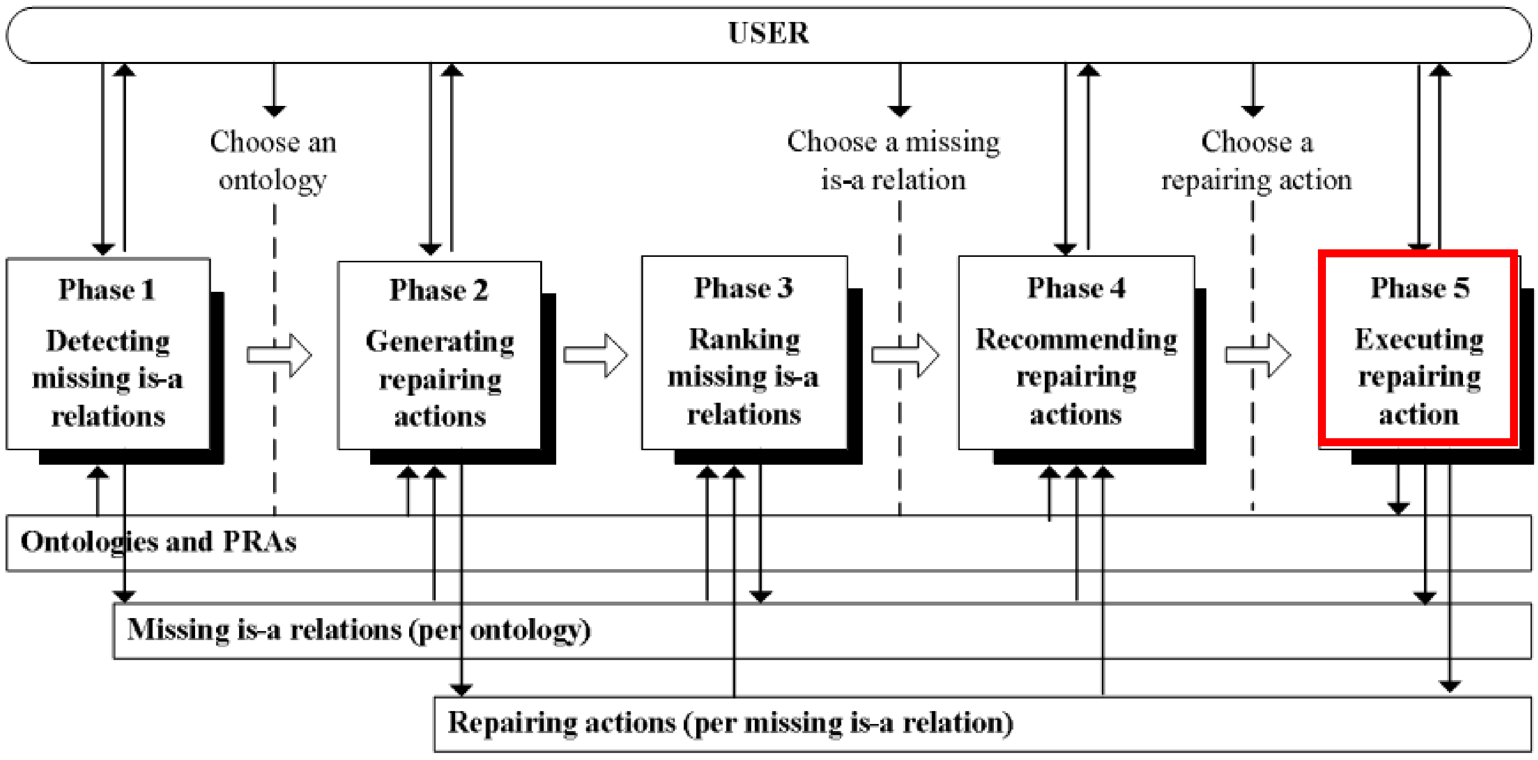
- Algorithm
  - Given a missing is-a relation with already generated repairing actions, among those, recommend the most informative repairing actions that are supported by evidence in the domain knowledge.

Example

For missing is-a relation **(hip_joint, joint)**, the recommendation algorithm suggests from the previously generated repairing actions the use of **(limb_joint, joint).**



Phase 5.
Executing repairing actions



Executing repairing actions

- Intuition
  - Every time a repairing action is chosen and executed, the repairing actions for the other missing is-a relations need to be recomputed based on the taxonomy extended with the chosen repairing action.
  - In order to optimize the update process, keep track of the influences.

Outline

- Background
- Definitions
- Debugging approach
- **Implemented system**
- Experiments
- Future Work

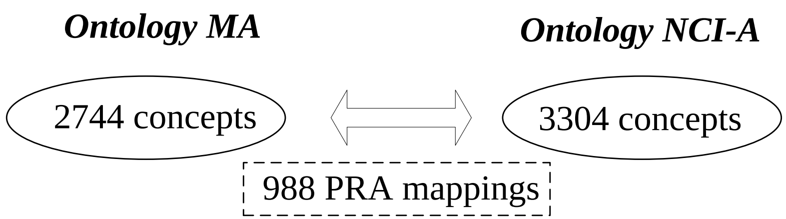

RepOSE
RepOSE
Repair of Ontological Structure Environment
Ontologies networked by PRAs
Load Ontologies and PRAs
Ontologies : 2
joint_mouse_anatomy.owl
Show Ontology
Missing is-a relations
Detect Missing IS-A Relations
Missing IS-A Relations : 7
MA_0000460:wrist joint -> MA_0000319:joint
Repair
Generate Repairing Actions
useExtendedAlg
Recommend Repairing Actions
WordNet
UMLSK
Recommendations : 1
MA_0000691->MA_0000319
Source : 3
MA_0000460: wrist joint
MA_0000614: forelimb joint
MA_0000691: limb joint
Repair
Save
Target : 26
MA_0001499: gomphosis
MA_0001505: joint of girdle
MA_0000453: metacarpophalangeal joint
MA_0000459: shoulder joint
MA_0000322: synovial joint
MA_0000319: joint
MA_0001514: joint of vertebral body
MA_0000451: elbow joint
MA_0001513: joint of vertebral arch
MA_0001509: costochondral joint
MA_0000471: knee joint
MA_0001508: joint of rib
MA_0001512: sternocostal joint
MA_0001511: interchondral joint
MA_0001501: synchondrosis

Outline

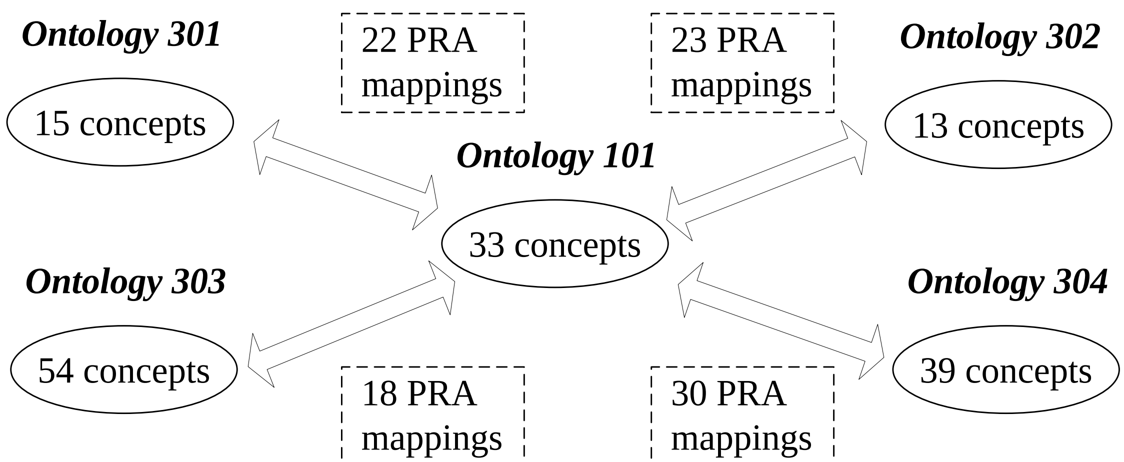
- Background
- Definitions
- Debugging approach
- Implemented system
- **Experiments**
- Future Work

Datasets

- Anatomy dataset (2008 OAEI Anatomy)



- Bibliography dataset (2010 OAEI Benchmark)



Experiment result

- Bibliography Dataset – 1 network
  - Initially, we found
    - 22 missing is-a relations in ontology 101 (of which 12 redundant)
    - 20 in ontology 304 (of which 13 redundant + 1 becoming redundant during repair)
    - 1 in each of the others.
  - During the repairing
    - We found 3 additional missing is-a relations in ontology 304.
  - The whole debugging process took about 5 minutes.

Experiment result

- Bibliography Dataset – 4 small networks
  - Initially, we found
    - For 101-301: 1 missing is-a relation for each ontology
    - For 101-302: 17 missing is-a relations (of which 11 redundant) for 101 and 1 for 302
    - For 101-303: 1 missing is-a relation for 303
    - For 101-304: 4 missing is-a relations for 101 and 5 (of which 1 redundant) for 304
  - During the repairing, no additional missing is-a relations were found.
  - The whole debugging process took less than 5 minutes.

Experiment result

- Bibliography Dataset – comparison

- 301, 302, 303: same results in both scenarios
- More missing is-a relations found and repaired in the scenario with 1 network

Experiment result

- Anatomy Dataset
  - Initially, we found 199 missing is-a relations in MA and 167 in NCI-A.
  - During the repairing
    - We found 6 additional missing is-a relations in MA, and 10 in NCI-A.
    - For 25 missing is-a relations in MA and 11 in NCI-A, the repairing actions changed.
    - In most cases, the ranking and recommendations seemed useful.
    - Most source and target sets are small enough to allow a good visualization.
  - Extended algorithm: influences for most missing is-a relations; clusters
  - The whole debugging process took about 3 hours.

Experiment result

- Recommending repairing actions
  - We use WordNet as domain knowledge.
  - The running time for generating recommendations for all missing is-a relations was
    - Circa 4 minutes for MA
    - Circa 2 minutes for NCI-A
  - Number of recommendations
    - MA: 19 receive 1; 12 receive 2; 2 receive 3.
    - NCI-A: 5 receive 1.

Experiment result

- Anatomy Dataset

	total initially	equivalence initially	redundant initially	to repair initially
MA	199	6	78	115
NCI-A	167	3	84	80

Figure 17: Scenario 1 - Initially detected missing is-a relations.

	total during	equivalence during	redundant during	to repair during
MA	6	0	1	5
NCI-A	10	0	3	7

Figure 29: Scenario 1 - Additionally detected missing is-a relations during whole debugging session.

	total	explicitly repaired	by others	obvious self	obvious non-self	ask recommendation
MA	120	101	19	28	0	73
NCI-A	87	87	0	7	0	80

Figure 25: Scenario 1 - Repaired missing is-a relations.

	total	use rec self	use rec non-self	not use rec self	not use rec non-self
MA	73	52	16	3	2
NCI-A	80	73	6	0	1

Figure 28: Scenario 1 - Recommendations.

Outline

- Background
- Definitions
- Repairing the structure of an ontology
- Implemented system
- Experiments
- **Future Work**

Extension

- Debugging *wrong* and missing is-a structure within networked taxonomies
  → demo session

Experiment on Anatomy dataset (2010 OAEI Anatomy)

MA: 2744 concepts, 1807 asserted is-a relations

NCI-A: 3304 concepts, 3761 asserted is-a relations

PRA: 986 equivalence relations, 1 subsumption

→

new is-a relations: 107 for MA, 64 for NCI-A

removed is-a relations: 3 from MA, 12 from NCI-A

total: 5 hours debugging time (almost all time on validation)

Future work

- Debugging is-a structure within networked ontologies
  - ontologies in more expressive knowledge representation languages

- Investigate the interaction and integration of ontology alignment and ontology debugging process

References

- Lambrix P, Liu Q, Tan H, Repairing the missing is-a structure of ontologies, *Proceedings of the 4th Asian Semantic Web Conference - ASWC09*, LNCS 5926, 76-90, 2009.

- Lambrix P, Liu Q, RepOSE: A system for debugging is-a structure in networked taxonomies, Demo at the *10th International Semantic Web Conference – ISWC11*, 2011.