Christian Meilicke – University Mannheim

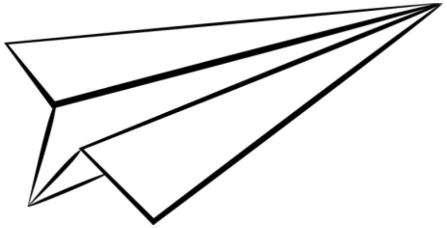
From Alignment Coherence towards a New Model for Ontology Matching

IN OMNIBUS VERITAS

UNIVERSITÄT MANNHEIM

About me

- Christian Meilicke
  - 1997-2003 studied philosophy / educational science in Mannheim
  - 2003-2006 bachelor computer science in Mannheim
  - 2007 researcher at the Chair of Prof. Stuckenschmidt
  - End of 2011 received Phd
  - Postdoc at Prof. Stuckenschmidt



What I did

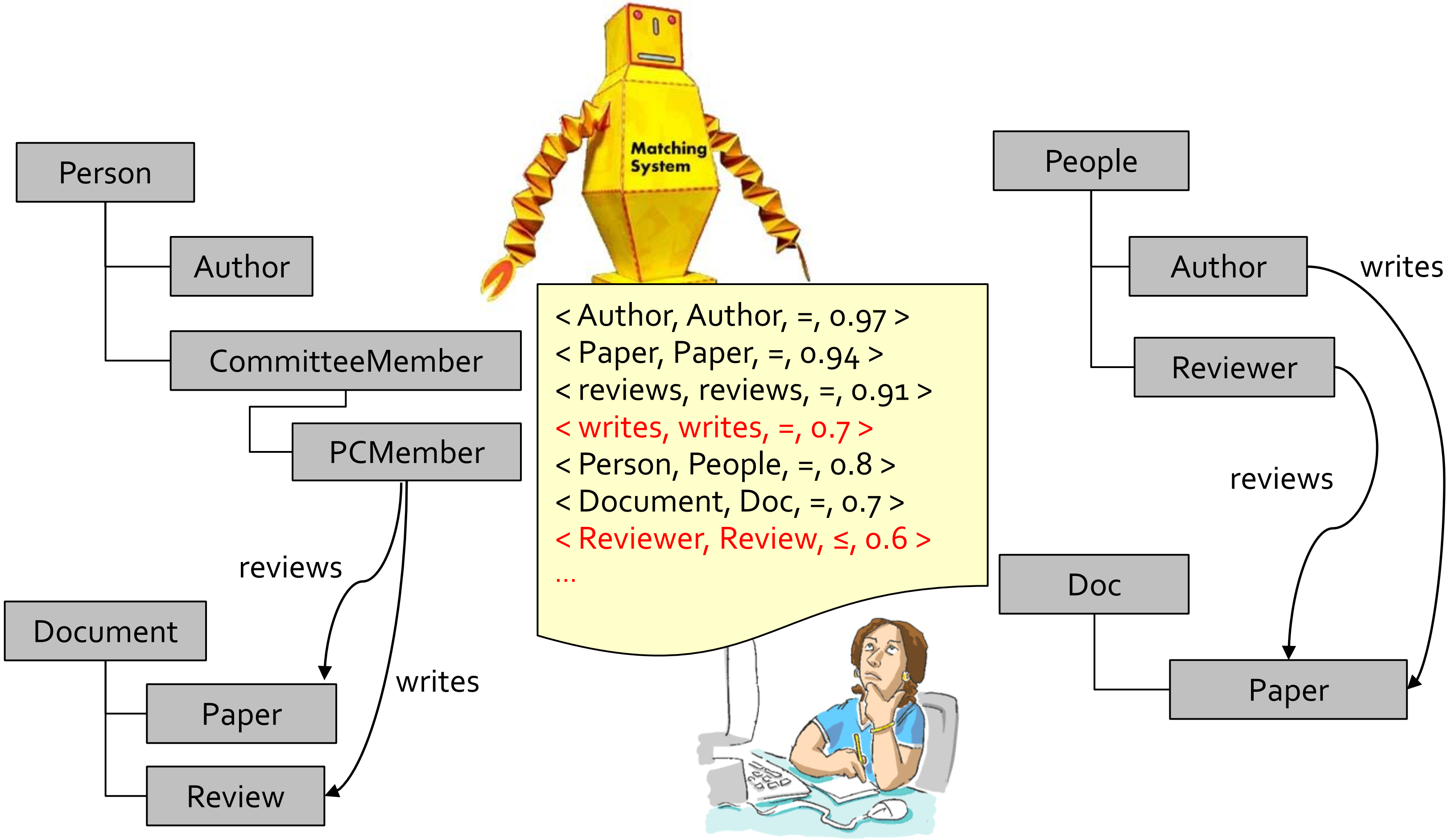
- Research Topic: Alignment (In)Coherence
  - Reasoning
  - Optimization
- OAEI
  - Ontology Alignment Evaluation Initiative
  - SEALS = Semantic Evaluation at Large Scale
  - Automation of Evaluation Process

Outline

- PART I: Alignment Incoherence
  - Preliminaries & Motivating Example
  - Algorithms
  - Experimental Results
- PART II: Matching as Optimization
  - Implemented in CODI at OAEI 2011 (and 2012)
- PART III: A new approach towards Ontology Matching
  - submitted as project proposal to DFG

- PART I: Alignment (In)coherence
  - … some things I did in my thesis

Ontology Matching



< Author, Author, =, 0.97 >
< Paper, Paper, =, 0.94 >
< reviews, reviews, =, 0.91 >
< writes, writes, =, 0.7 >
< Person, People, =, 0.8 >
< Document, Doc, =, 0.7 >
< Reviewer, Review, ≤, 0.6 >
…

Alignment Incoherence

- In the context of reductionistic alignment semantic S, the aligned ontology $A_S(O_1,O_2)$ is defined as $O_1 \cup O_2 \cup X$

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

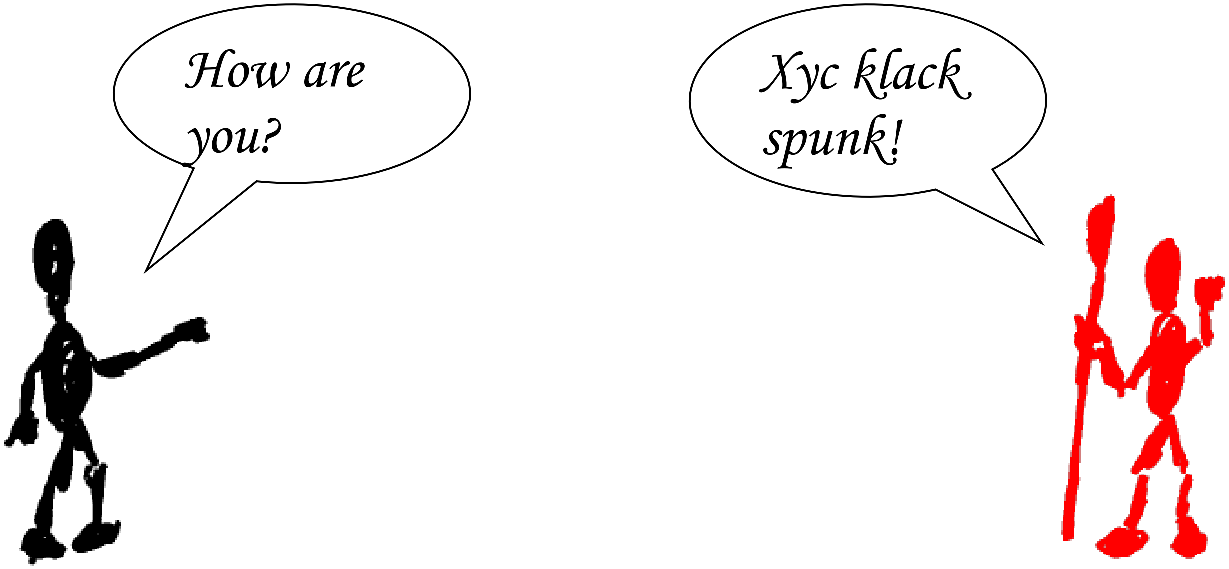
- An alignment A is incoherent iff $A_S(O_1,O_2)$ is incoherent, i.e. iff $A_S(O_1,O_2)$ contains an unsatisfiable concept or property

OAEI 2012 – Conference Track

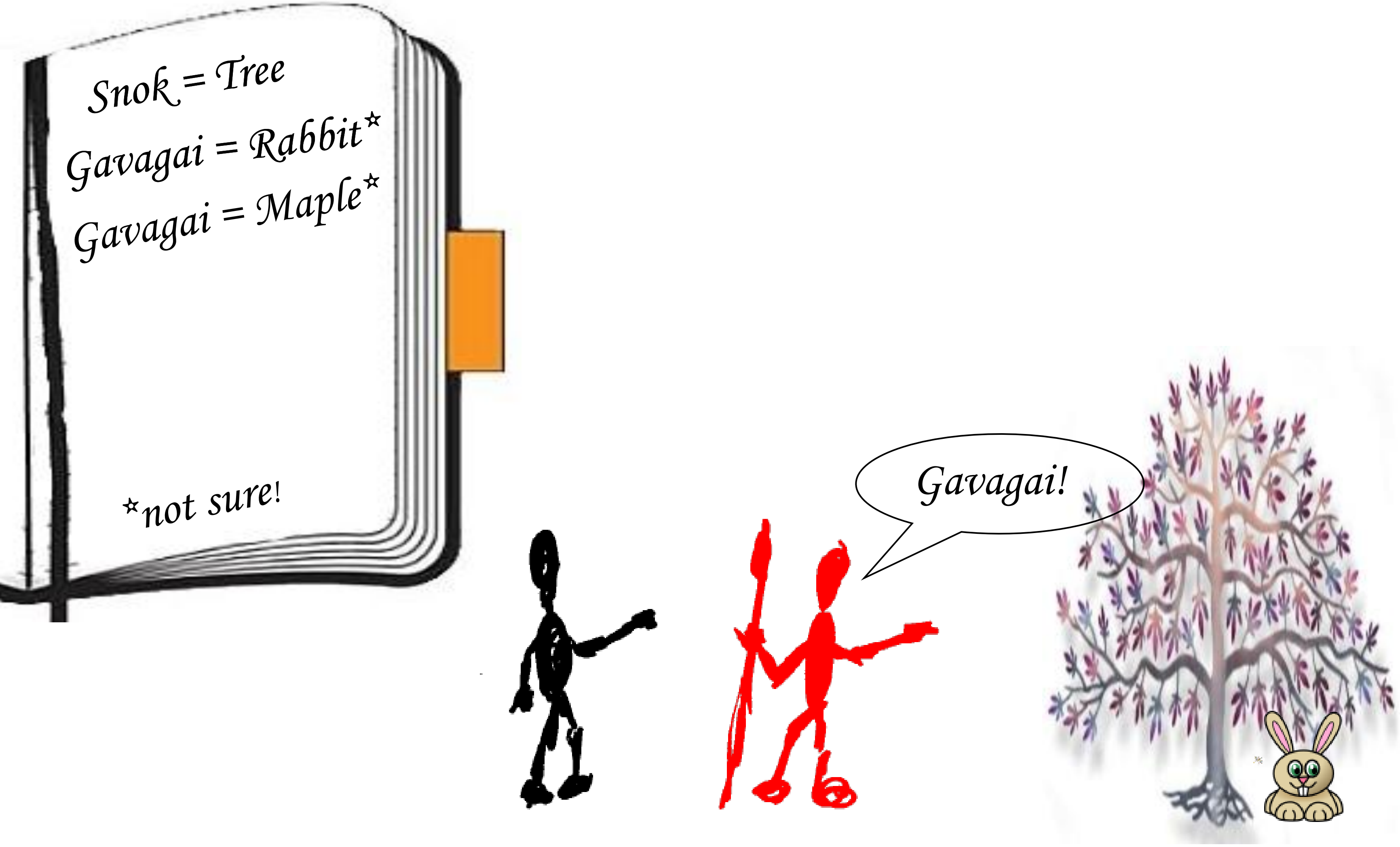
- Only 4 of 16 systems generate coherenct alignment
  - LogMap (uses specific reasoning techniques)
  - CODI (details later)
  - YAM (uses ALCOMO)
  - ServoMapLt (very small alignments)
- All other systems are still incoherent
- In average ~10% of all correspondences have to be removed to have a coherent alignment

Motivating Example

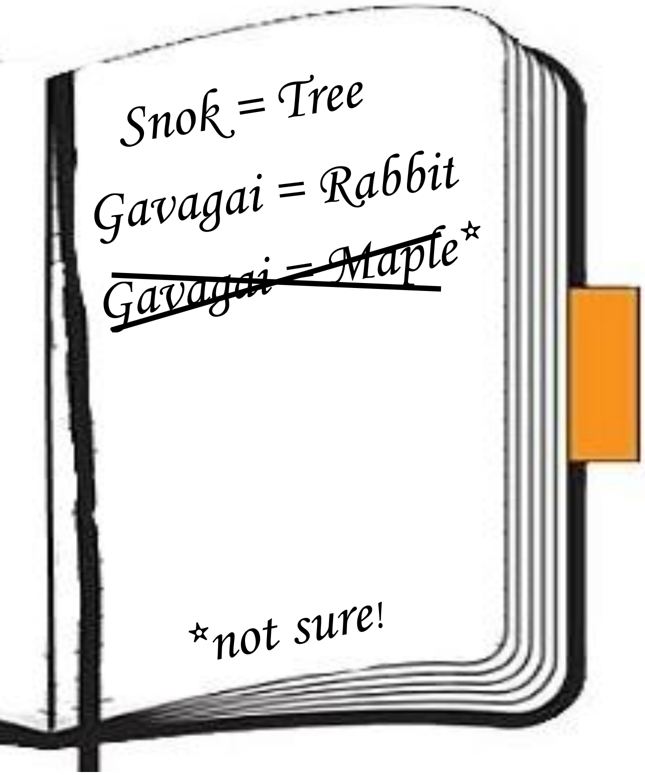
- Translating between English and an unknown language

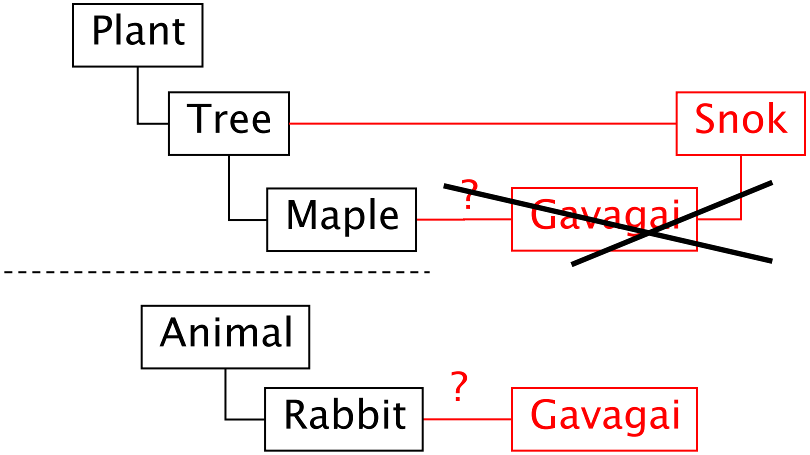


Motivating Example



Motivating Example





Plant

Tree

Maple

Snok

?

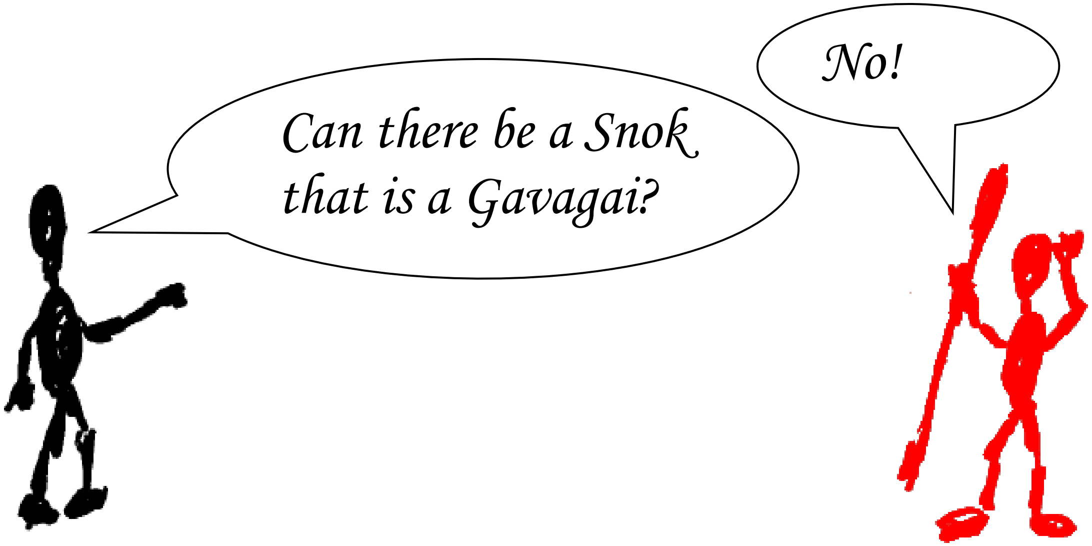
~~Gavagai~~

Animal

Rabbit

?

Gavagai



Example in Description Logics

$O_1 = \{$

$\text{Maple} \sqsubseteq \text{Tree} \sqsubseteq \text{Plant}$

$\text{Rabbit} \sqsubseteq \text{Animal}$

$\text{Animal} \sqsubseteq \neg\, \text{Plant}$

$\}$

$A = \{$

$\text{Tree} \equiv \text{Snok}$

$\text{Maple} \equiv \text{Gavagai}$

$\}$

$O_2 = \{$

$\text{Gavagai} \sqsubseteq \neg\, \text{Snok}$

$\}$

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

$A_S(O_1, O_2) \models \text{Gavagai} \sqsubseteq \neg\, \text{Snok}$

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

Diagnosis

- 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.

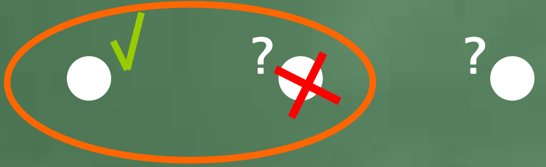
- A subset $\Delta \subseteq A$ is a diagnosis for A (w.r.t. O_1 and O_2) iff
  - $A \setminus \Delta$ is coherent and there exists no $\Delta' \subset \Delta$ such that $A \setminus \Delta'$ is coherent

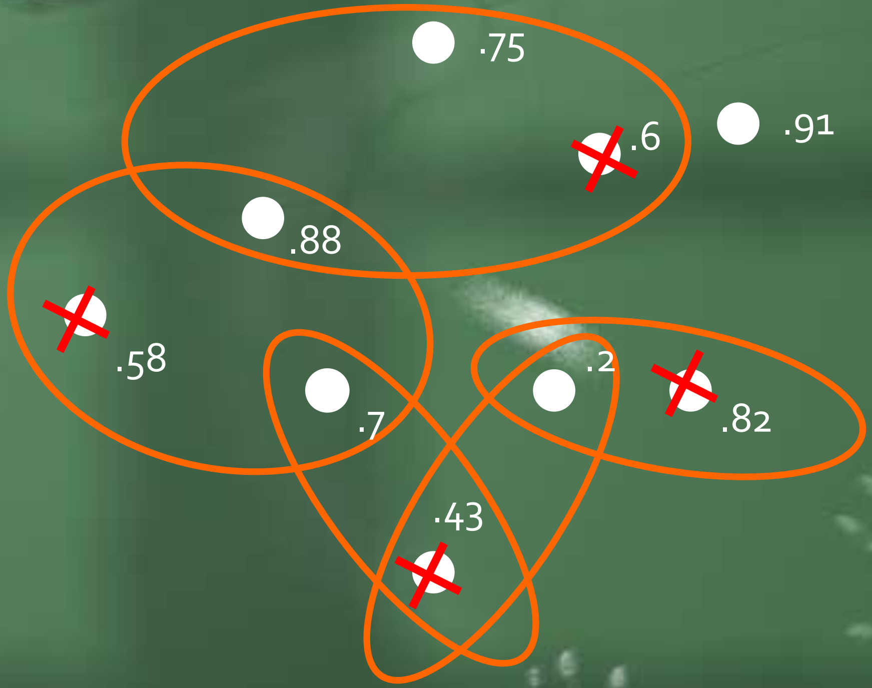
Which Diagnosis?

Snok = Tree

Gavagai = Maple ?

Gavagai = Rabbit ?





Global Optimal Diagnosis

A subset $\Delta \subseteq$ A of an incoherent alignment A is diagnosis for A (w.r.t. O1 and O2) iff

- Δ is a diagnosis and
- there exists no Δ' such that $\sum_{c \in \Delta'} < \sum_{c \in \Delta}$.

The diagnosis with minimal total of confidence values

Two challenges

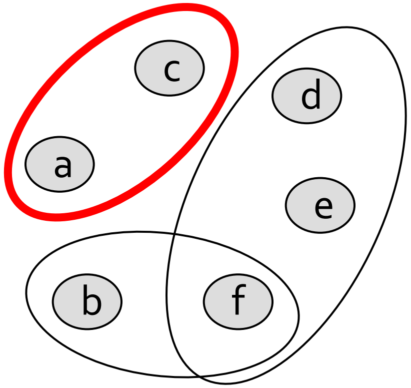
- Determine the conflict sets („orange sets“, also called MIPS)
  - **M**inimal **I**ncoherence **P**reserving **S**ub-alignment
  - Requires specific reasoning techniques
  - Number of MIPS can be very high
- Solve the optimization problem
  - Weighted Hitting Set Problem
    - Related decision problem is NP-complete
  - Can be done with different methods
  - E.g. simple search algorithm

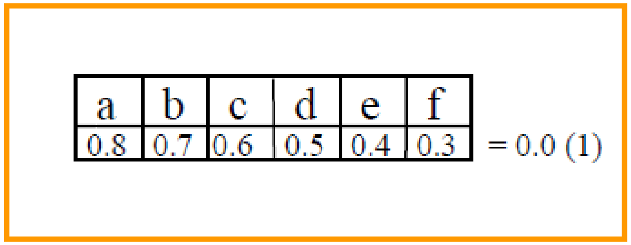
Reasoning

- Not main topic of this talk … suppose we have two algorithms :
  - Pattern-based algorithm that finds nearly all MIPS in short time
  - Expensive algorithms using full-fledged reasoning that finds a single MIPS

Details can be found in:
Christian Meilicke: Alignment Incoherence in Ontology Matching. University Mannheim 2011

Uniform-Cost-Search

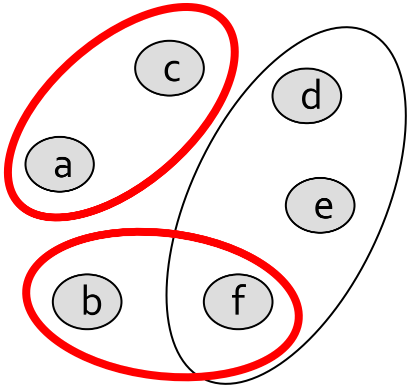


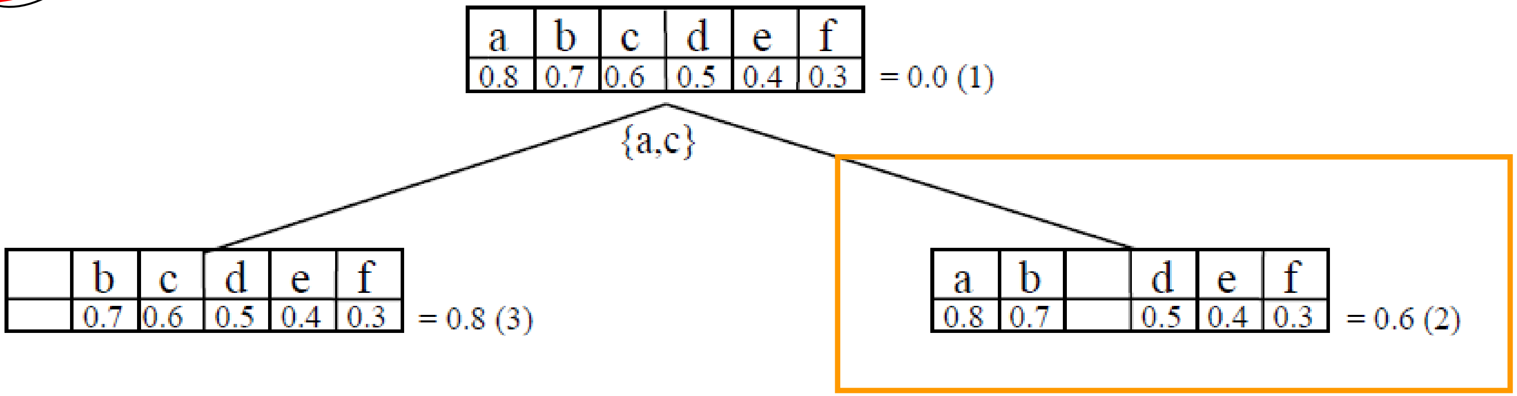


a	b	c	d	e	f
0.8	0.7	0.6	0.5	0.4	0.3

= 0.0 (1)

Uniform-Cost-Search





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

= 0.0 (1)

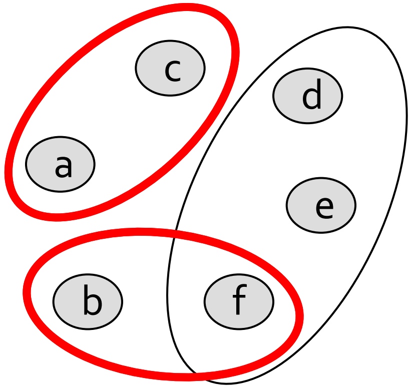
	b	c	d	e	f
	0.7	0.6	0.5	0.4	0.3

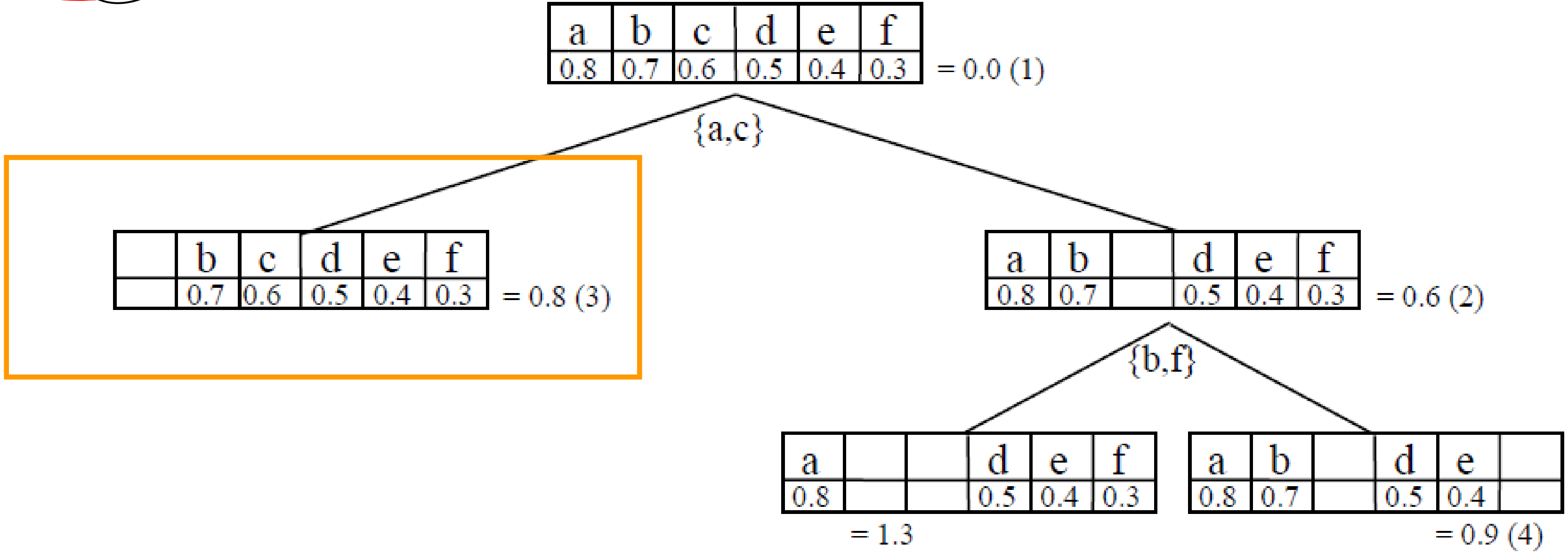
= 0.8 (3)

a	b		d	e	f
0.8	0.7		0.5	0.4	0.3

= 0.6 (2)

Uniform-Cost-Search





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

= 0.0 (1)

{a,c}

	b	c	d	e	f
	0.7	0.6	0.5	0.4	0.3

= 0.8 (3)

a	b		d	e	f
0.8	0.7		0.5	0.4	0.3

= 0.6 (2)

{b,f}

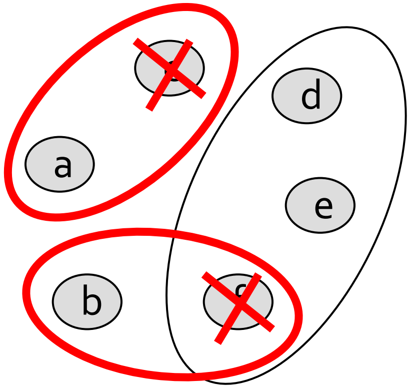
a			d	e	f
0.8			0.5	0.4	0.3

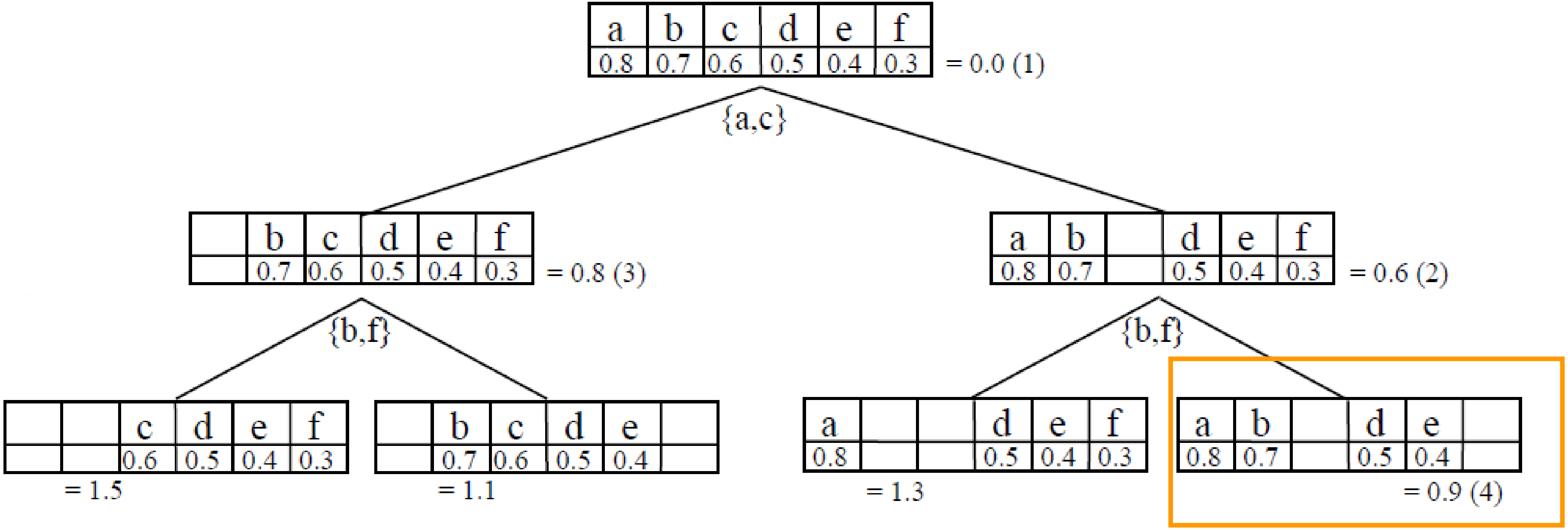
= 1.3

a	b		d	e	
0.8	0.7		0.5	0.4	

= 0.9 (4)

Uniform-Cost-Search





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

= 0.0 (1)

	b	c	d	e	f
	0.7	0.6	0.5	0.4	0.3

= 0.8 (3)

a	b		d	e	f
0.8	0.7		0.5	0.4	0.3

= 0.6 (2)

		c	d	e	f
		0.6	0.5	0.4	0.3

= 1.5

	b	c	d	e	
	0.7	0.6	0.5	0.4	

= 1.1

a			d	e	f
0.8			0.5	0.4	0.3

= 1.3

a	b		d	e	
0.8	0.7		0.5	0.4	

= 0.9 (4)

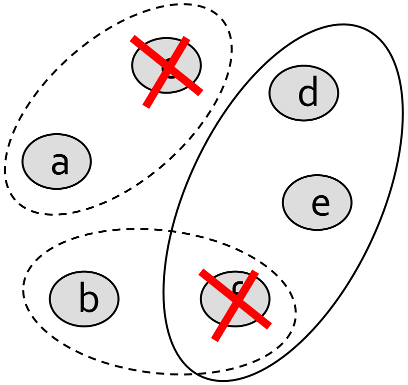
Patternbased Reasoning

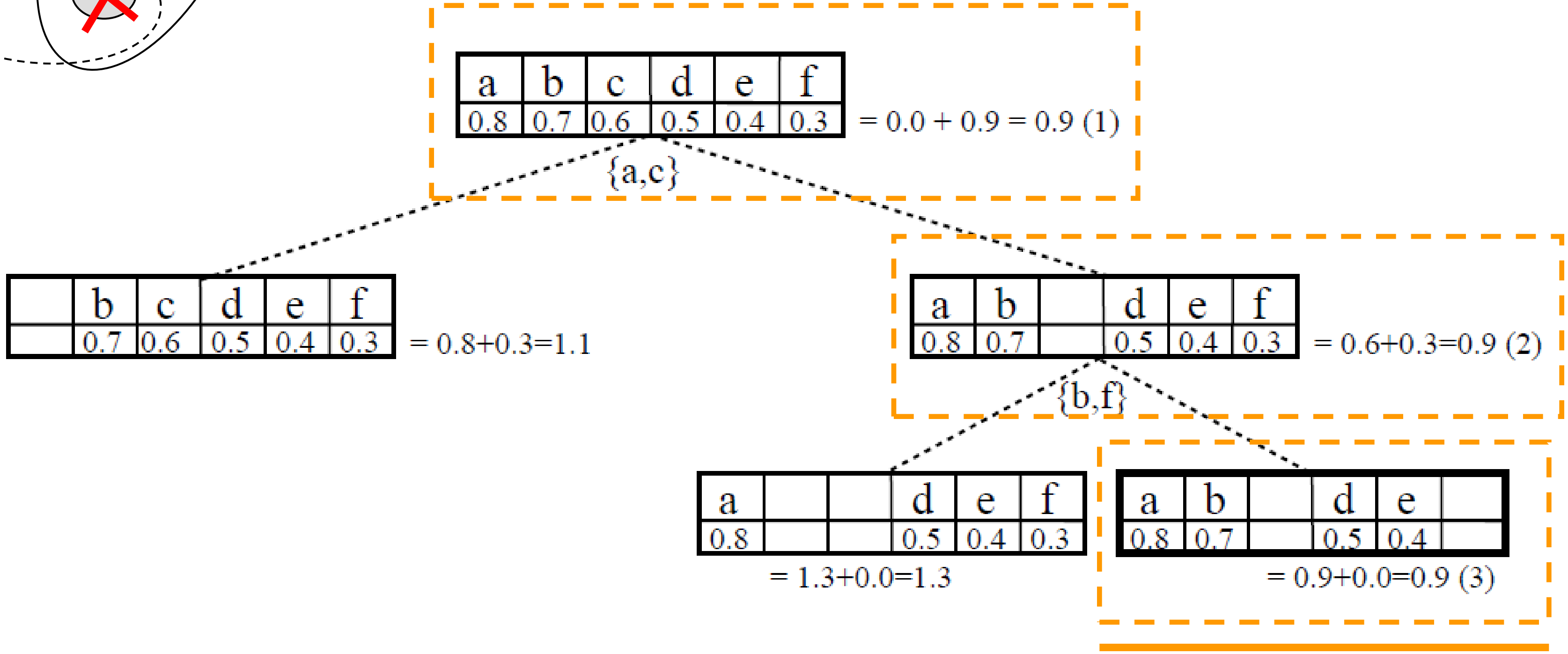
- Idea: Use incomplete method for incoherence detection for pairs of correspondences in preprocessing step

- Use MIPS available after preprocessing for branching in the upper levels of the tree

- Use fullfledged reasoning only, when all previously found MIPS are resolved

A*-Search with Patternbased Reasoning





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

= 0.0 + 0.9 = 0.9 (1)

	b	c	d	e	f
	0.7	0.6	0.5	0.4	0.3

= 0.8+0.3=1.1

a	b		d	e	f
0.8	0.7		0.5	0.4	0.3

= 0.6+0.3=0.9 (2)

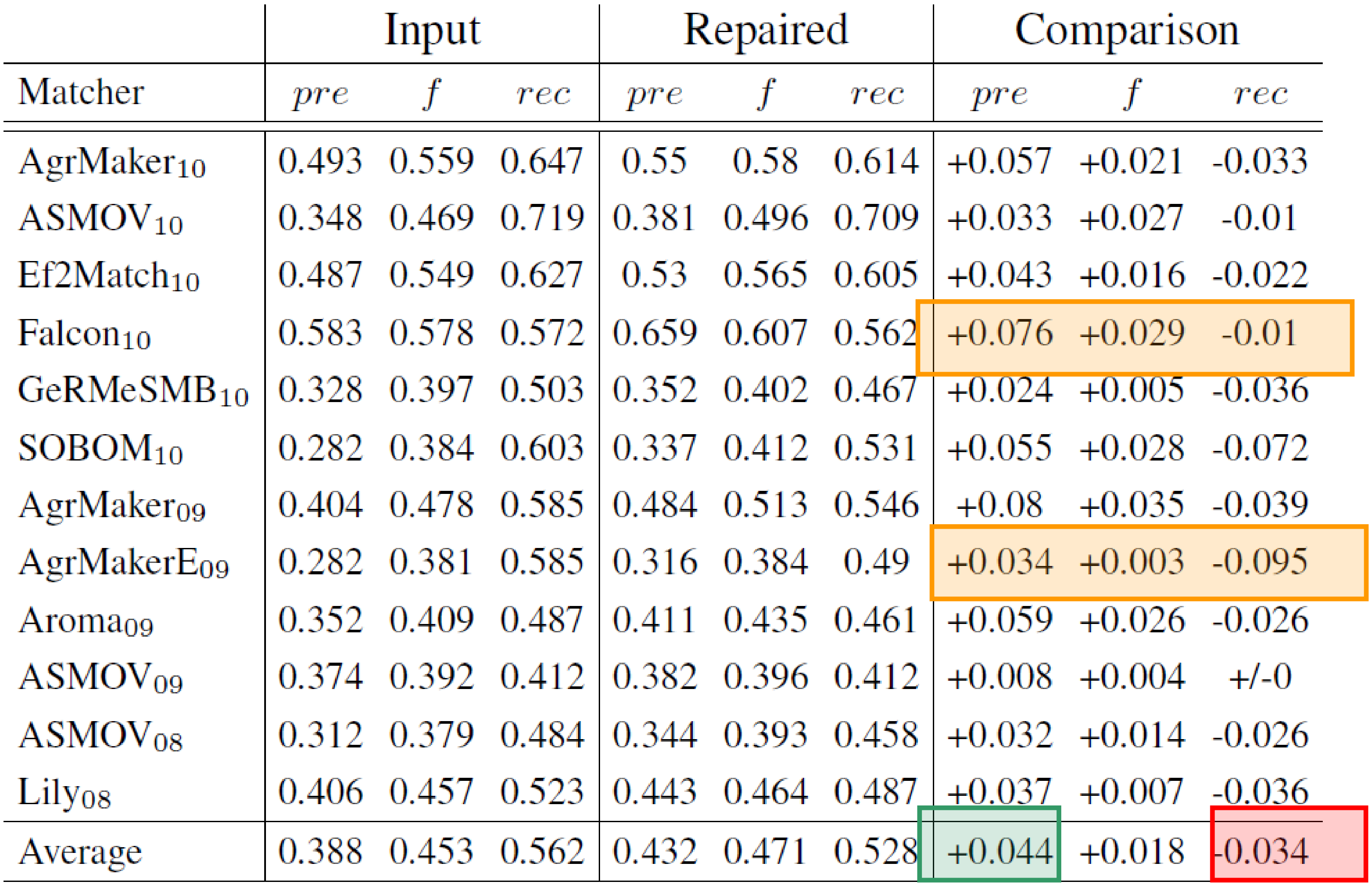
a			d	e	f
0.8			0.5	0.4	0.3

= 1.3+0.0=1.3

a	b		d	e	
0.8	0.7		0.5	0.4	

= 0.9+0.0=0.9 (3)

Debugging Matching Systems



	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

Conclusion

- Can be applied to the outcome of any matching system as post-processing step
- Search algorithms to find global optimal solution
  - For larger problems not efficient
  - No method will be efficient for very large problems
- Improvement in precision, small loss in recall
  - Relatively small improvement of overall quality in terms of F-measure

- PART II: Matching as Optimization
  - more generic and extendable
  - CODI = Combinatorial Optimization for Data Integration

Matching Process

1. Similarities are computed
   - String based similarity measures
   - WordNet or other external resources

2. Similarities are refined
   - Similarity flooding
   - Other structural measures

3. Alignment is extracted
   - One-to-one constraint
   - Coherence constraint

Matching Process

1. Similarities are computed
   - String based similarity measures
   - WordNet or other external resources

2. Similarities are refined
   - Similarity flooding
   - Other structural measures

NO MORE SEARCHING

3. Alignment is extracted
   - One-to-one constraint
   - Coherence constraint

OPTIMIZATION PROBLEM

Markov Logic

- Analyze Ontologies and Labels
  - Markov Logic formulae that describe structure
  - Mappings as weighted Markow Logic formulae
- Define general constraints
  - Hard 1:1 and coherency constraints
  - Soft stability constraints
- Compute MAP state
  - The state with maximum a-posteriori likelyhood
  - Translate to ILP and use GUROBI to solve it
  - Retranslate solution to MAP state
  - Retranslate MAP state to alignment

Structure of the Ontology

```
subsumes1(1#Person, 1#Author)
subsumes1(1#Author, 1#FirstAuthor)

disjoint1(1#Document, 1#Person)

domainsub1(1#writes, 1#Author)
rangesub1(1#writes, 1#Paper)

...
```

!

Mapping Hypothesis

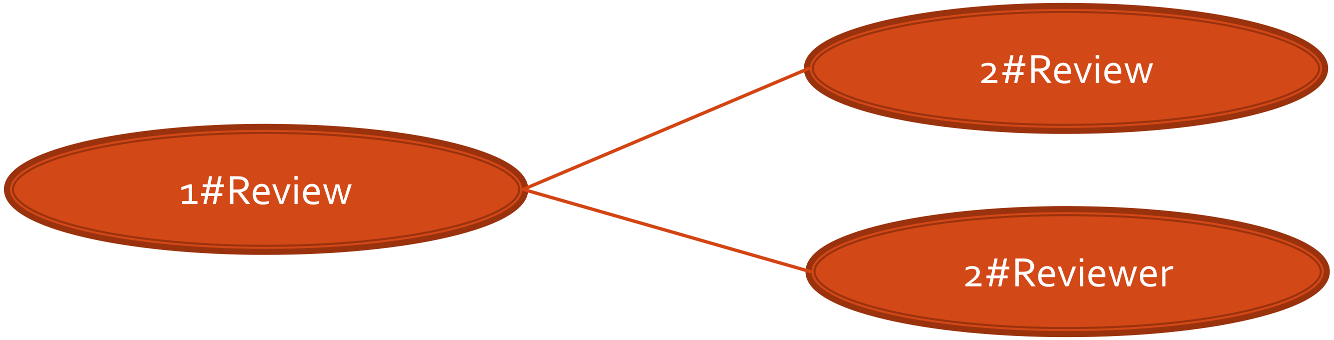
```
cmap(1#Person, 2#Person), 0.98
cmap(1#Review, 2#Reviewer), 0.76

pmap(1#writes, 2#writesPaper), 0.66
...
```

$$\sum$$

1:1 Constraints

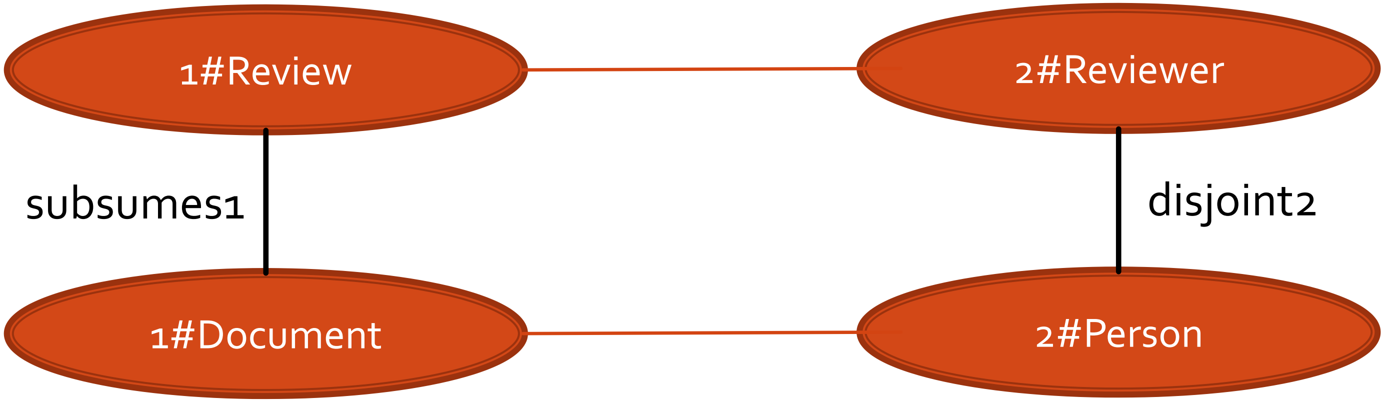
```
|c2| cmap(c1, c2) <= 1.
|c1| cmap(c1, c2) <= 1.
|p2| pmap(p1, p2) <= 1.
|p1| pmap(p1, p2) <= 1.
```

!



Coherence Constraints

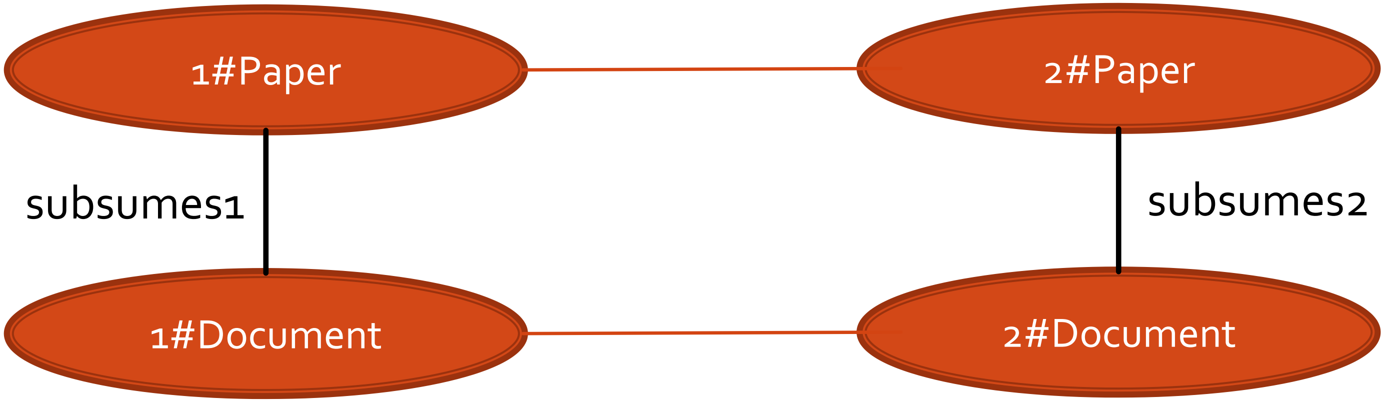
```
subsumes1(c1, b1) AND disjoint2(c2, b2) AND cmap(c1, c2) => !cmap(b1, b2).
subsumes2(c2, b2) AND disjoint1(c1, b1) AND cmap(c1, c2) => !cmap(b1, b2).
domainsub1(p1, c1) AND domaindis2(p2, c2) AND cmap(c1, c2) => !pmap(p1, p2).
...
```



Stability Constraints

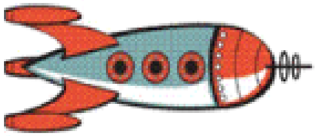
```
0.25  subsumes1(c1, b1) AND subsumes2(c2, b2)
=> cmap(c1, c2) n cmap(b1, b2)
...
```

∑



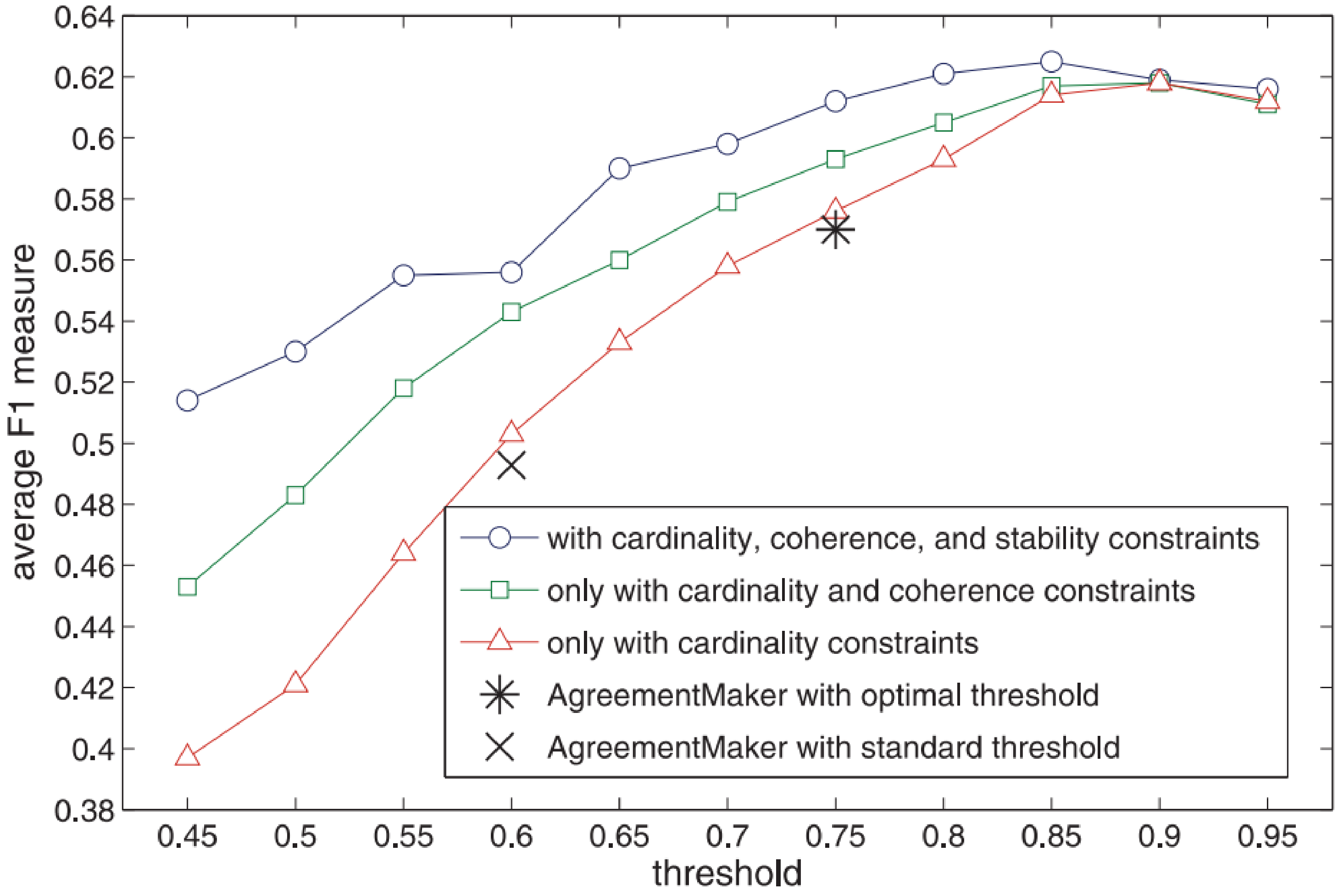
CODI

- **Complete description** of CODI matching system
  - Details on similarity measures not presented
  - Not all constraints related to properties shown
- Translation to ILP based in Jan Nößners ROCKIT system
  - https://code.google.com/p/rockit/



- Reasoning about coherency
  - Coherence rules are equivalent to pattern-based reasoning
  - CODI is sometimes incoherent

Benefits of an integrated approach

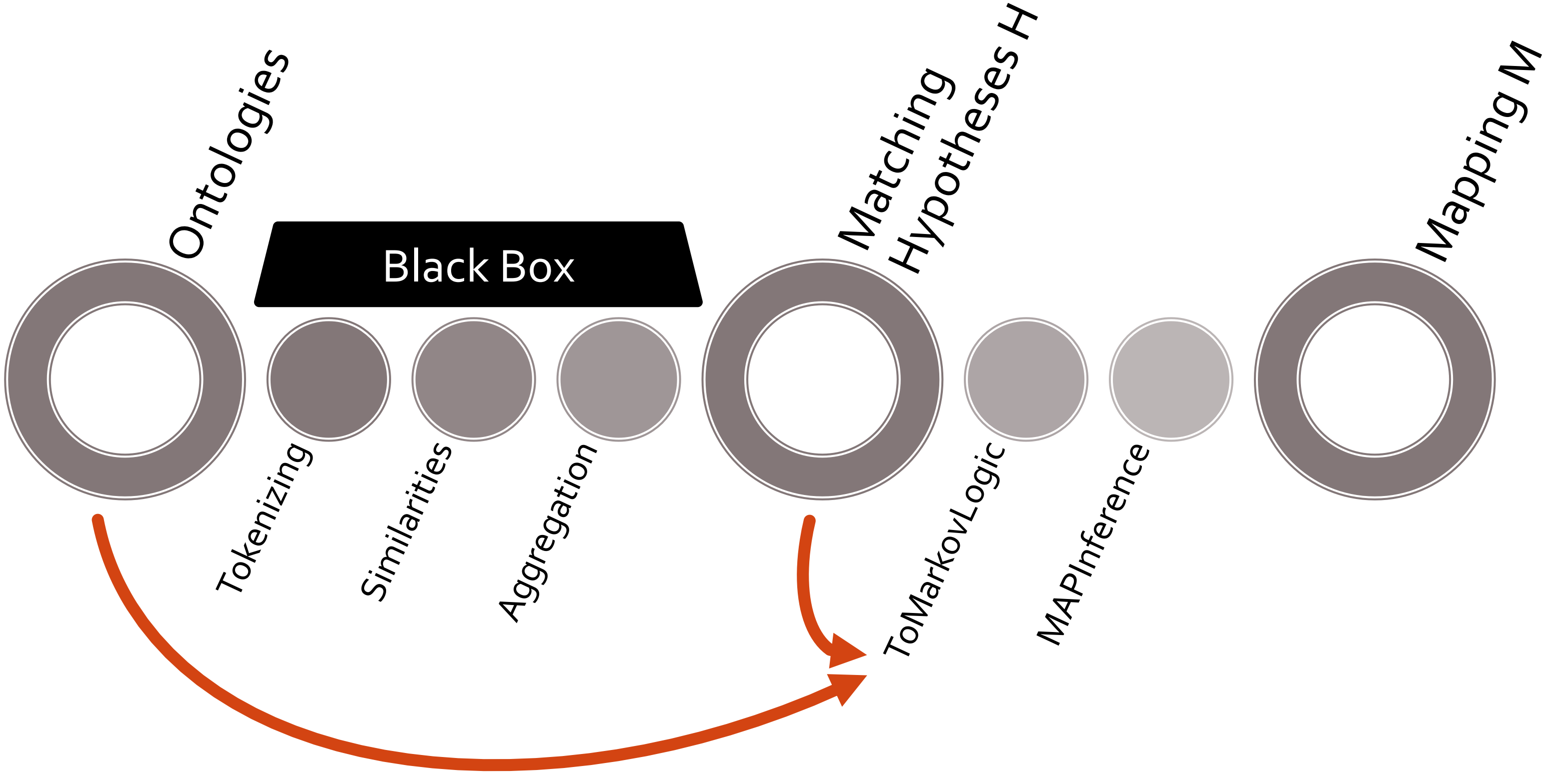


Conclusion

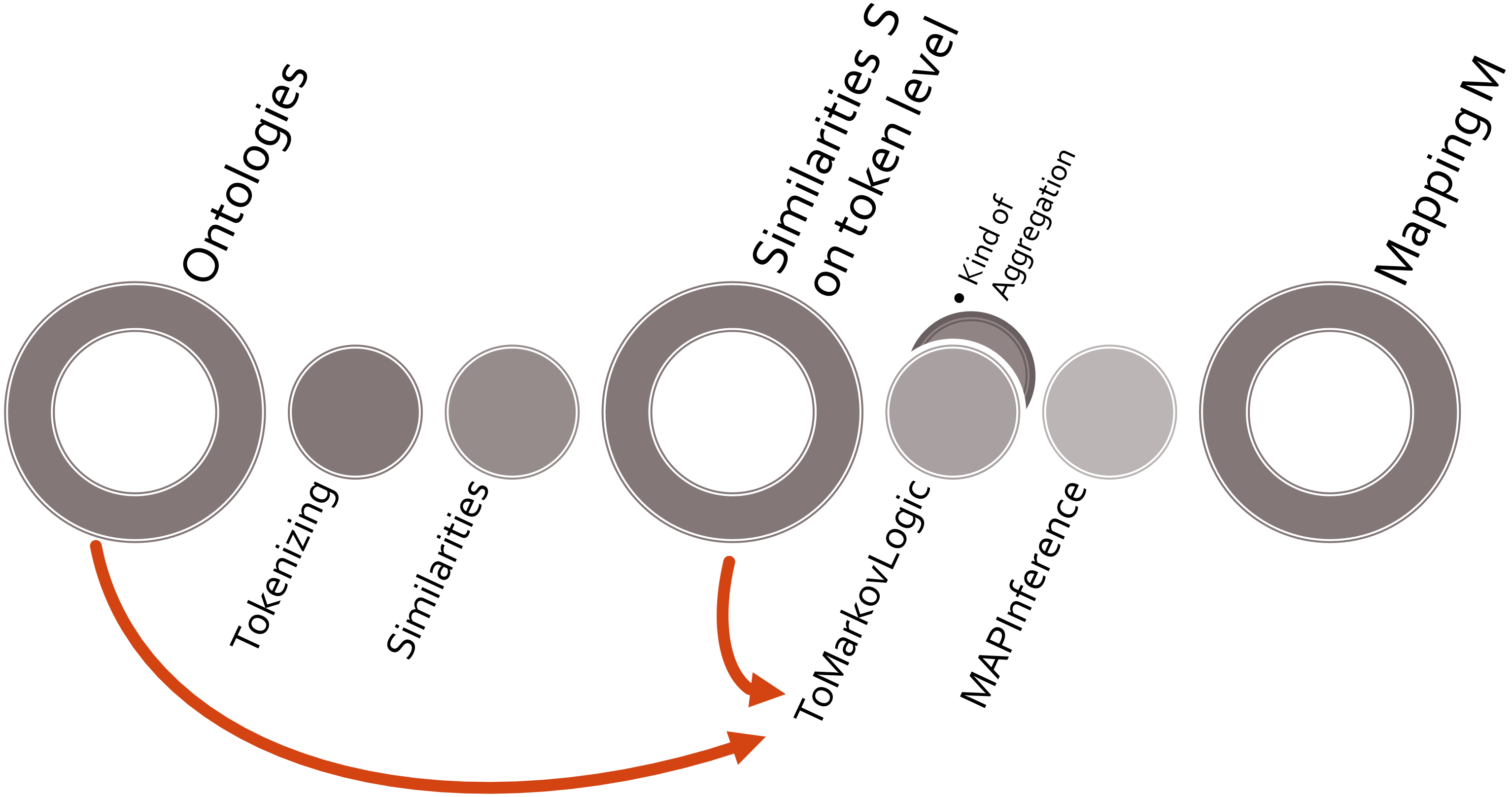
- Clear way to define the matching process
  - You just write down what you want as result
- Stability constraints help to improve the results slightly
- Much more effcient way to solve the optimization problem
  - … compared to a selfmade search algorithm

- PART III: A new approach towards Ontology Matching
  - *to be is to be the value of a variable (Quine)*
  - labels become part of the optimization problem
  - beneficial for complex matching

Overall Matching Process



A minor modification …



Notation

- 1#AcceptedPaper
  - denotes an entity (concept) from ontology 1
- 1:Accepted
  - denotes a label attached to an entity from ontology 1

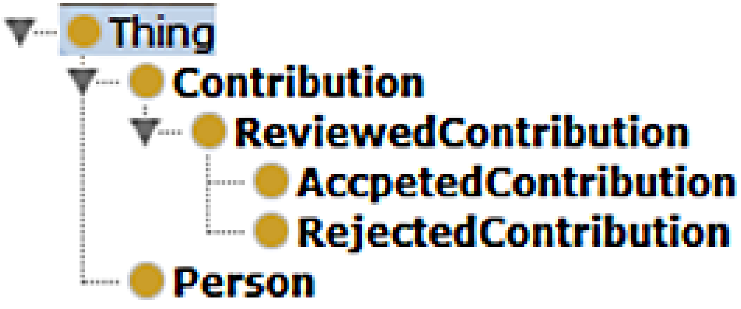
Modelling two mapping levels

- Mappings on entity level
  - `cmap(1#AcceptedPaper, 2#AcceptedContribution)`
  - `pmap(1#writesPaper, 2#writtenBy)`
- Mappings on token level
  - `tmap(1:Accepted, 2:Accepted), 0.5`
  - `tmap(1:Paper, 2:Contribution), -0.31`
- Linking entities and token
  - `headnoun(1#AcceptedPaper, 1:Paper)`
  - `modifier(1#AcceptedPaper, 1:Accepted)`

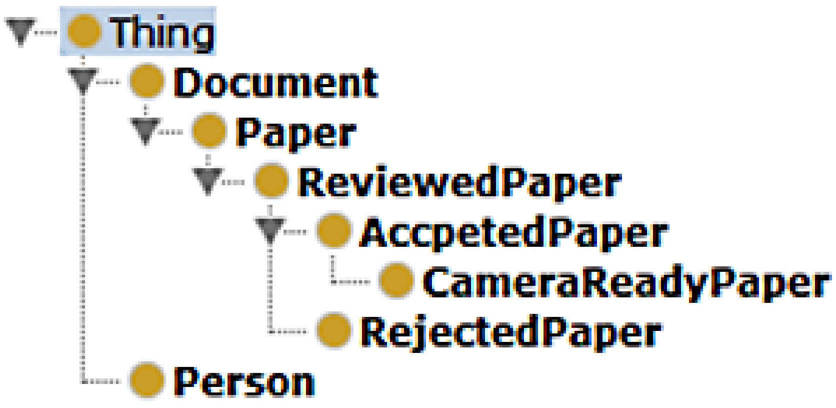
A toy example

- Using ROCKIT to solve the MAP inference problem

- Tiny example to illustrate the effects

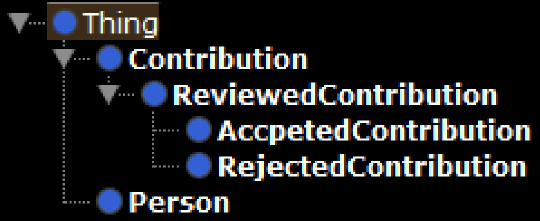


- Thing
  - Contribution
    - ReviewedContribution
      - AccpetedContribution
      - RejectedContribution
  - Person

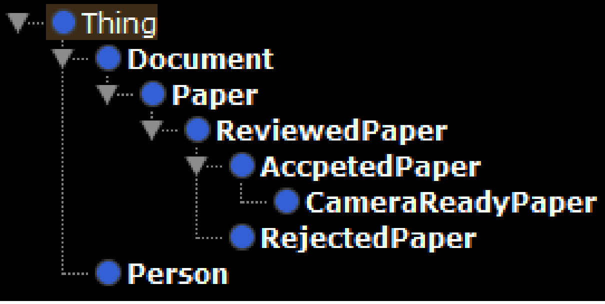


- Thing
  - Document
    - Paper
      - ReviewedPaper
        - AccpetedPaper
          - CameraReadyPaper
        - RejectedPaper
  - Person

1st Trial



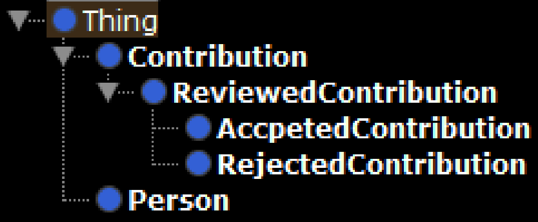
- Thing
  - Contribution
    - ReviewedContribution
      - AccpetedContribution
      - RejectedContribution
  - Person



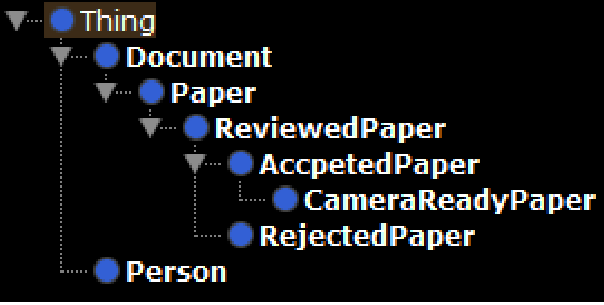
- Thing
  - Document
    - Paper
      - ReviewedPaper
        - AccpetedPaper
          - CameraReadyPaper
        - RejectedPaper
  - Person

- Hard constraints
  - 1:1 constraint on concept level
- Soft constraints
  - Add similarity for each `tmap(......)` that is in the solution
- Results
  - `tmap(1:Accpeted, 2:Accpeted)`
  - `tmap(1:Reviewed, 2:Reviewed)`
  - `tmap(1:Rejected, 2:Rejected)`
  - `tmap(1:Person, 2:Person)`

Linking ...



- Thing
  - Contribution
    - ReviewedContribution
      - AccpetedContribution
      - RejectedContribution
  - Person

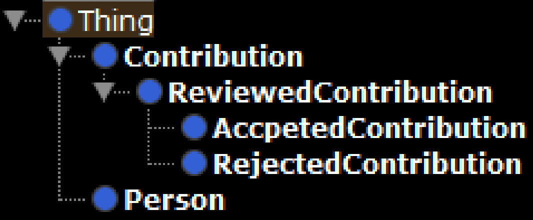


- Thing
  - Document
    - Paper
      - ReviewedPaper
        - AccpetedPaper
          - CameraReadyPaper
        - RejectedPaper
  - Person

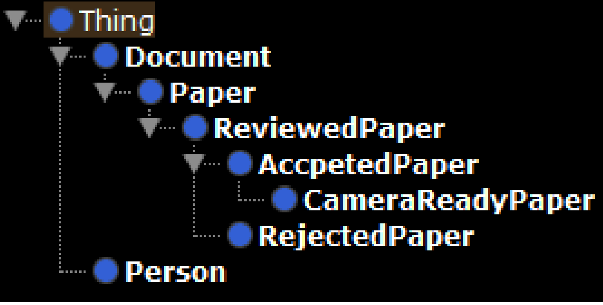
- Hard constraints
  - 1:1 constraint on concept level
  - **NEW: mapping tokens => mapping concepts**
- Soft constraints
  - Add similarity for each `tmap(......)` that is in the solution
- Results
  - `tmap("1:Accpeted", "2:Accpeted")`
  - `tmap("1:Reviewed", "2:Reviewed")`
  - `tmap("1:Rejected", "2:Rejected")`
  - `tmap("1:Person", "2:Person")`

  - `cmap("1#Person", "2#Person")`

Stability



- Thing
  - Contribution
    - ReviewedContribution
      - AccpetedContribution
      - RejectedContribution
  - Person

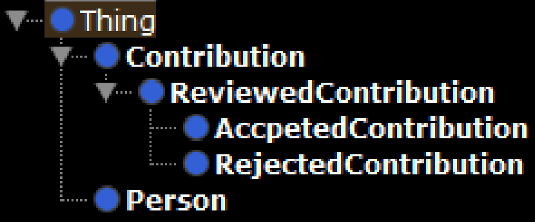


- Thing
  - Document
    - Paper
      - ReviewedPaper
        - AccpetedPaper
          - CameraReadyPaper
        - RejectedPaper
  - Person

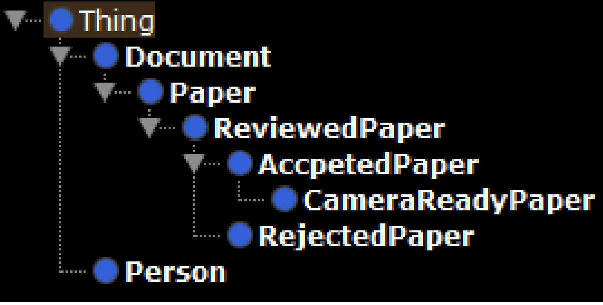
- Hard constraints
  - 1:1 constraint on concept level
  - mapping tokens => mapping concepts
- Soft constraints
  - Add similarity for each `tmap(    )` that is in the solution
  - **NEW: Stability constraint**
- Results:
  - `tmap("1:Accpeted", "2:Accpeted")`
  - `tmap("1:Reviewed", "2:Reviewed")`
  - `tmap("1:Rejected", "2:Rejected")`
  - `tmap("1:Person", "2:Person")`

  - `cmap("1#Document", "2#Contribution")`
  - `cmap("1#AccpetedPaper", "2#RejectedContribution")`
  - `cmap("1#CameraReadyPaper", "2#AccpetedContribution")`
  - `cmap("1#Paper", "2#ReviewedContribution")`
  - `cmap("1#Person", "2#Person")`

Reversed



- Thing
  - Contribution
    - ReviewedContribution
      - AccpetedContribution
      - RejectedContribution
  - Person



- Thing
  - Document
    - Paper
      - ReviewedPaper
        - AccpetedPaper
          - CameraReadyPaper
        - RejectedPaper
  - Person

- Hard constraints
  - 1:1 constraint on concept level
  - mapping tokens => mapping concepts
  - **NEW: mapping concepts => mapping tokens**
- Soft constraints
  - Add similarity for each `tmap(......)` that is in the solution
  - Stability constraint
- Results
  - `tmap("1:Paper", "2:Contribution")`
  - `tmap("1:Accpeted", "2:Accpeted")`
  - `tmap("1:Reviewed", "2:Reviewed")`
  - `tmap("1:Rejected", "2:Rejected")`
  - `tmap("1:Person", "2:Person")`

  - `cmap("1#AccpetedPaper", "2#AccpetedContribution")`
  - `cmap("1#RejectedPaper", "2#RejectedContribution")`
  - `cmap("1#Paper", "2#Contribution")`
  - `cmap("1#ReviewedPaper", "2#ReviewedContribution")`
  - `cmap("1#Person", "2#Person")`

Some Remarks

- The same result can also be generated without the token/entity distinction?
  - Adding entity mappings with low confidence
  - Giving a high weight to the stability constraint

- Why not this way?
  - Stability has to „win" against several mappings with low confidence
  - Will generate lots of incorrect mappings

- In general:
  - Token vs. Entity approach is in line with our intuitive way of reasoning
  - Can be extended towards complex matching

Towards Complex Matching I

- If a property 1#p is described by a label 1:p and a property 1#q is described by a label 1:q and 1:p is the passive voice of 1:q then
  - pmap(1#p,1#q^{-1})
  - or maybe pmap(1#p, inv(1#q))
  - or maybe pmap-inv(1#p, 1#q)
- Example
  - pmap-inv(1#writtenBy,2#writes)

Towards Complex Matching II

- What about this:
  - AcceptedPaper ≡ Contribution ⊓ ∃hasBeenAccepted.⊤

```
cmap-exists(1#AcceptedPaper,1#Contribution, 2#hasBeenAccepted)
```

- Can be generated without any optimization / Markow Logic (Ritze et al., OM-2009/2010)
- However, using the optimization approach:
  - **Interference with soft and hard constraints !**
  - Easy to add/extend relevant constraints

Thanks a lot,
any Questions?

Constraints

```
// soft constraints
-0.2  !subsumes1(c1, b1) v !subsumes2(c2, b2) v !cmap(c1, c2) v !cmap(b1, b2)

cconf: !tmapConfidence(c1, c2, cconf) v  tmap(c1, c2)

|x| cmap(x,y) <= 1
|y| cmap(x,y) <= 1

//  token => entity
!onlyHeadNoun1(c1) v !onlyHeadNoun2(c2) v !headNoun1(c1, h1) v !headNoun2(c2, h2) v !tmap(h1, h2) v cmap(c1, c2).
!modifiedNoun1(c1) v !modifiedNoun2(c2) v !headNoun1(c1, h1) v !headNoun2(c2, h2) v !modifier1(c1, m1) v
!modifier2(c2, m2) v !tmap(h1, h2) v !tmap(m1, m2) v cmap(c1, c2).

//  entity => token
!onlyHeadNoun1(c1) v !onlyHeadNoun2(c2) v !headNoun1(c1, h1) v !headNoun2(c2, h2) v !cmap(c1, c2) v tmap(h1, h2).
!modifiedNoun1(c1) v !modifiedNoun2(c2) v !headNoun1(c1, h1) v !headNoun2(c2, h2) v !cmap(c1, c2) v tmap(h1, h2).
!modifiedNoun1(c1) v !modifiedNoun2(c2) v !modifier1(c1, m1) v !modifier2(c2, m2) v !cmap(c1, c2) v tmap(m1, m2).
```

Evidence

```
onlyHeadNoun1("1#Person")
headNoun1("1#Person", "1:Person")

modifiedNoun1("1#ReviewedPaper")
modifier1("1#ReviewedPaper", "1:Reviewed")
headNoun1("1#ReviewedPaper", "1:Paper")

onlyHeadNoun1("1#Document")
headNoun1("1#Document", "1:Document")

modifiedNoun1("1#AccpetedPaper")
modifier1("1#AccpetedPaper", "1:Accpeted")
headNoun1("1#AccpetedPaper", "1:Paper")

modifiedNoun1("1#RejectedPaper")
modifier1("1#RejectedPaper", "1:Rejected")
headNoun1("1#RejectedPaper", "1:Paper")

modifiedNoun1("1#CameraReadyPaper")
modifier1("1#CameraReadyPaper", "1:Camera")
modifier1("1#CameraReadyPaper", "1:Ready")
headNoun1("1#CameraReadyPaper", "1:Paper")

onlyHeadNoun1("1#Paper")
headNoun1("1#Paper", "1:Paper")

modifiedNoun1("1#CamerareadyPaper")
modifier1("1#CamerareadyPaper", "1:Cameraready")
headNoun1("1#CamerareadyPaper", "1:Paper")

subsumes1("1#ReviewedPaper", "1#AccpetedPaper")
subsumes1("1#ReviewedPaper", "1#RejectedPaper")
...
```

```
...

modifiedNoun2("2#AccpetedContribution")
modifier2("2#AccpetedContribution", "2:Accpeted")
headNoun2("2#AccpetedContribution", "2:Contribution")

subsumes2("2#ReviewedContribution",
"2#RejectedContribution")
subsumes2("2#ReviewedContribution",
"2#AccpetedContribution")
subsumes2("2#Contribution", "2#RejectedContribution")
subsumes2("2#Contribution", "2#ReviewedContribution")
subsumes2("2#Contribution", "2#AccpetedContribution")


tmapConfidence("1:Paper", "2:Accpeted",-0.25)
tmapConfidence("1:Paper", "2:Reviewed",-0.375)
tmapConfidence("1:Paper", "2:Contribution",-0.4166)
tmapConfidence("1:Paper", "2:Rejected",-0.375)
tmapConfidence("1:Paper", "2:Person",-0.33384)
tmapConfidence("1:Accpeted", "2:Accpeted",0.5)
tmapConfidence("1:Accpeted", "2:Reviewed",-0.125)
```