

# Applying Logical Constraints on Ontology Matching (Christian Meilicke & Heiner Stuckenschmidt)

... could also be called  
“Incoherence Handling in Mappings“

Koblenz, April 15th, 2008  
(partially based on a KI-2007 presentation)

# Overview

- Preliminaries
  - Problem of matching ontologies
  - Mapping extraction as subproblem
  - Naive solution to the extraction problem
  -
- Extracting consistent mappings
  - Mapping consistency
  - Reasoning about mapping consistency
  - Algorithms for extraction
- Experimental results
  - Problems, surprises & some results
- Discussion / Future work

# Matching Problem

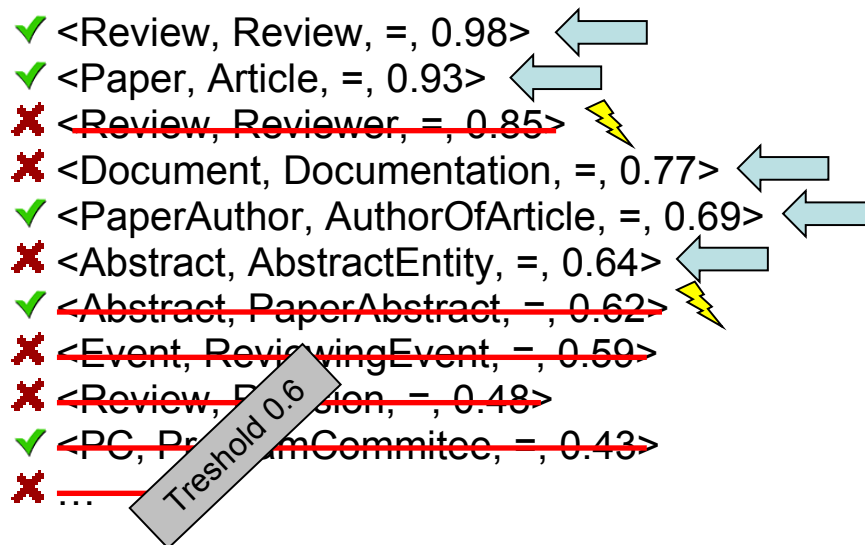
- Given ontologies  $T_1$  and  $T_2$ , find all correct correspondences (a mapping) between the entities of  $T_1$  and  $T_2$ .
- Correspondences are four-tuples  $\langle e, e', r, c \rangle$  with
  - $e$  and  $e'$  are matchable elements of  $T_1$  resp.  $T_2$
  - $r$  is the semantic relation
  - $c$  is a confidence values
- In this work  $e$  and  $e'$  are concepts / properties,  $c$  in  $[0,1]$ , and  $r$  is the equivalence or subsumption relation, e.g for ontologies  $T_1$  and  $T_2$  we might have  $\langle 1:\text{Author}, 2:\text{Person}, \subseteq, 0.88 \rangle$
- Many matching systems compute similarities to solve this problem:
  - Syntactic similarity (greatest common substring, tokenisation, word equality)
  - Similarity estimation by external resources (e.g. wordnet distance)
  - Similarity by correspondences of individuals
  - Structural similarity (taxonomy, graphs)
  - Propagation of similarities (similarity flooding)

# Subproblem of Mapping Extraction

- As result of the similarity computation we have a similarity matrix as intermediary matching results => understand as comprehensive mapping  $M$
- Problem of mapping extraction: Given the intermediary matching result find final extraction  $M' \subseteq M$  such that  $M'$  contains all of the correct correspondences in  $M$
- Standard extraction methods to solve this problem are ...
  - (1) Applying a threshold.
  - (2) Finding one-to-one mappings via a greedy approach.
  - (3) Finding optimal one-to-one mappings with respect to the sum of confidences
  - (4) ...
- In many systems (e.g. Falcon-AO, Rimom, HMatch) a combination of (1) and (2) is implemented

# Naive Solution to the Problem

- Example of the threshold + greedy approach:



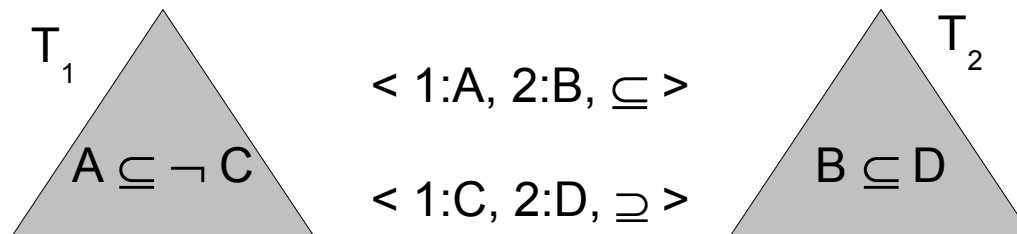
- Knowledge encoded in the ontologies is only used indirectly via knowledge encoded in the confidence values. => Extraction is ontology independent.
- Can the semantics of the ontologies be directly used in the extraction process to improve the quality of the extraction?

# Mapping Consistency

- Translate correspondences of mapping  $M$  into the corresponding axioms in a straight forward way.
  - Datatype properties are a problem, because properties might have a (nearly) equivalent meaning, but differ with respect to their range (cannot be explained in detail)
  - e.g.  $\langle 1:\text{hasName}, 2:\text{hasName}, =, 1.0 \rangle$   
(could be datatype property in  $T_1$  and object property in  $T_2$ )
  - Solution: Don't translate correspondence. Instead:
    - Add artificial concepts as subclasses of domain for both properties
    - Add axiom that expresses semantic relation between these concepts
- Construct the merged ontology  $T_1 \cup_M T_2$  that consists of
  - Axioms of  $T_1$  and  $T_2$
  - Translated mapping axioms
- $M$  is consistent (more precise: coherent) iff there exists no unsatisfiable class in  $T_1 \cup_M T_2$

# Reasoning

- Complete (and straight forward): Use **one** reasoner as black box
  - Create, load and classify  $T_1 \cup_M T_2$
  - Unsatisfiable concept in  $T_1 \cup_M T_2 \Leftrightarrow M$  is inconsistent
- Efficient but incomplete: Use **two** reasoners as black boxes and exploit simple structural pattern to check consistency of correspondence pairs
  - Load and classify  $T_1$  and  $T_2$  once!
  - Check, if the following pattern occurs:



- Pattern occurs  $\Rightarrow$  pair is inconsistent!
- Some pair in  $M$  is inconsistent  $\Rightarrow M$  is inconsistent

## Greedy extraction algorithm

```
Extractcom(T1, T2, M)
  M' = {}
  sort M descending with respect to confidences
  foreach c in M:
    M' = M' ∪ {c}
    if not IsConsistentcom(T1, T2, M'):
      M' = M' \ {c}
  return M'
```

- Checks mapping subsets of increasing size for consistency
  - Each time removing the correspondences with lowest confidence, when an inconsistency occurs
  - Requires  $|M|$  times classifying (or at least reasoning in) the merged ontology
- By replacing  $\text{IsConsistent}_{\text{com}}$  by  $\text{IsConsistent}_{\text{eff}}$  (checking each pair of correspondences in the mapping) we get an incomplete but efficient extraction method  $\text{Extract}_{\text{eff}}$ .
  - Requires classifying  $T_1$  and  $T_2$  once, pattern checks are nearly for free (even though we got lots of them)

## Combining both approaches (rather efficient and complete)

```
HybridExtract( $T_1$ ,  $T_2$ , M)
  sort M descending with respect to confidences
   $M^* = \{\}$  // final result (consistent every time)
   $M' = M$  // doubted mapping (might not be consistent)
  while not  $M' = \{\}$ :
     $M'' = \text{Extract}_{\text{eff}}(T_1, T_2, M^* \cup M')$ 
    search binary for index  $i$  such that
    (1)  $\text{IsConsistent}_{\text{com}}(T_1, T_2, M''[\dots i-1])$  and
    (2) not  $\text{IsConsistent}_{\text{com}}(T_1, T_2, M''[\dots i])$ 
    (3)  $i = |M''|$  iff  $\text{IsConsistent}_{\text{com}}(T_1, T_2, M'')$ 
    // we can be sure that efficient reasoning was complete up to  $i$ 
     $M^* = M''[\dots i-1]$ 
    // everything beyond  $i$  has to be doubted => reset  $M'$ 
     $M' = M[\text{adjust}(i+1) \dots]$ 
  return  $M^*$ 
```

# Runtime of approach

- For efficient component:
  - 1 \* classifying  $T_1$  and 1 \* classifying  $T_2$
  - Several times checking structural pattern (nearly for free)

Let  $n$  be the number of correspondences causing inconsistencies detected by complete reasoning but not detected by efficient reasoning.

Let  $k$  be the number of correspondences causing inconsistencies detected by efficient reasoning.

- For complete component:
  - $n(\log(|M|))$  \* classifying  $T_1 \cup_M T_2$  (with decreasing size of  $M$ )
- Empirical observation:
  - In most cases we have  $n \ll |M|$  and **more important**  $n \ll k$
  - For all testcases `HybridExtract` outperforms `Extractcom` significantly

## Evaluation: First results (1/2)

- Problems with evaluating the approach
  - Only a **few testcases with reference mappings** are available
  - In general **only equivalence correspondences** are specified
  - Many testcases do **not contain disjointness axioms** (or something else that causes inconsistencies)
  - **Not easy to generate many-to-many mappings** with matching systems
- First experiments: Benchmark dataset of the OAEI 2007
  - Accepted standard, nearly all testcases are synthetic datasets, except testcases #301 to #304 (real ontologies about publications)
  - Reference mappings contain correspondences between concepts and correspondences between properties
  - Disjointness axioms had to be added manually
  - Instead of using many-to-many mappings, we used 1:1 mappings generated by matching systems
- First results: Minor positive effects, often only a trade off between precision and recall! **We did not expect this, what happened?**

## Evaluation: First results (2/2)

- Results: Reference mappings (used for 3 years!) contain some incorrect correspondences that can be found by logical reasoning!
- Some examples:
  - Part of reference mapping <101:Part, 302:Publication, =, 1.0>
    - In  $T_{101}$ : **101:Part** has as disjoint sibling class **101:Book**
    - In  $T_{302}$ : **302:Publication** is a superclass of **302:Book**
    - => concepts cannot be equivalent, conflicts with matching 101: book on 302:book
  - Part of reference mapping <101:year, 301:hasYear, =, 1.0>
    - In  $T_{101}$ : Datatype property **101:year** has domain **101:Date**
    - In  $T_{301}$ : Datatype property **301:hasYear** has domain **301:Entry** (a kind of publication)
    - => properties cannot be equivalent, due to different (and disjoint) domains.
- We detected 9 of 224 correspondences that were incorrect, and have been removed by our approach and thus falsify our results
  - Fixed reference mappings and informed OAEI organizer
  - Results of approach (based on fixed reference mappings) increased f-measure up to 5% for different matchers, but still in many cases only a minor improvement

## Discussion / Future Work

- Much better results can be expected when working with many-to-many mappings (thresholded similarity matrix)
  - Applying the approach to 1:1 mappings can only increase precision.
  - ‚Alternatives‘ will be available for many-to-many mappings, positive effect on recall can be expected (compared to naive extraction)
- Approach seems to work surprisingly good as support for mapping revision
  - Integrate approach in GUI
  - Quality control for mappings
- Missing disjointness is a problem
  - Disjointness can be learned, when not specified
  - Meilicke, Völker, Stuckenschmidt: Learning Disjointness for Debugging Mappings between Lightweight Ontologies (submitted to EKAW-08)
- Greedy approach works well, but ..
  - In some cases might be problematic (one incorrect correspondence with high confidence might cause removal of several correct correspondences)
  - Use optimal approach, to maximize sum of confidences.

Thanks for your attention!  
Any questions?

## Appendix A – Counterexample in AL

- Not all pairwise inconsistencies will be detected by our reasoning approach:

$$\begin{array}{l} \mathcal{T}_1 \\ \text{Painter} \sqsubseteq \forall \text{hasCreated}. \text{Image} \sqcap \exists \text{hasCreated}. \top \\ \text{range}(\text{hasCreated}) \equiv \text{Artwork} \quad (\top \sqsubseteq \forall \text{hasCreated}. \text{Artwork}) \end{array}$$

$$\begin{array}{l} \langle 1:\text{Artwork}, 2:\text{Article}, =, 1.0 \rangle \\ \langle 1:\text{Image}, 2:\text{Image}, =, 1.0 \rangle \end{array}$$

$$\mathcal{T}_2 \quad \text{Image} \sqsubseteq \neg \text{Article}$$

$$\mathcal{T}_1 \cup_{\mathcal{M}} \mathcal{T}_2 \models 1:\text{Image} \sqsubseteq \neg 1:\text{Artwork}$$

$$\mathcal{T}_1 \cup_{\mathcal{M}} \mathcal{T}_2 \models \forall 1:\text{hasCreated}. 1:\text{Image} \sqsubseteq \neg \exists 1:\text{hasCreated}. \top$$

$$\mathcal{T}_1 \cup_{\mathcal{M}} \mathcal{T}_2 \models \perp \sqsupseteq 1:\text{Painter}$$


## Appendix B – Weighted Vertex Cover

- A vertex cover of an undirected graph  $G = (V, E)$  is a subset  $V'$  of  $V$  which contains at least one of the two endpoints of each edge.
- Translate problem into conflict graph  $G = (V, E)$ :
  - $V$  is the set of correspondence of the input mapping, the weight of each vertex is the confidence of the correspondence
  - $E$  is the set of conflict between pairs of correspondences
  - Find the vertex cover  $V'$  with minimal aggregated weight and return  $V - V'$

