Computing minimal mappings

Fausto Giunchiglia, Vincenzo Maltese, Aliaksandr Autayeu
Roadmap

- Lightweight ontologies
- Mapping and minimal mapping
  - Computing a mapping: SMatch
  - Computing the minimal mapping: MinSMatch
- Evaluation
- Conclusions
Lightweight ontologies (formal classifications)

- We translate the graphs in input into lightweight ontologies
  - Node labels are formulas in propositional Description Logic (DL)
  - Concepts are taken from WordNet senses
  - Tree structures: each node formula is subsumed by parent node formula
Computing a mapping using SMatch

- A Mapping is a set of mapping elements \(<\text{source}, \text{target}, R>\)
  - \(R \in \{\perp, \equiv, \subseteq, \supseteq\}\) partially ordered
  - For each pair of nodes a call to a SAT solver verifies if a given semantic relation holds between the two, given the available background knowledge
  - Visualization and usability problems (e.g. validation and maintenance)
Redundancy patterns

- We provide:
  - A definition of *redundant mapping element* (dashed arrows) based on the redundancy patterns below (redundancy w.r.t. another element).
  - A demonstration of soundness and completeness

- **Dependencies across-symbols**: equivalence is the combination of more and less specific
  - Pattern 4 can be seen as the combination of patterns 1 and 2
  - Patterns 1 and 2 are still valid in case of equivalence between B-E

![Redundancy Patterns Diagram](image-url)
Minimal and redundant mappings

- We compute the **Minimal Mapping**
  - The subset of mapping elements of maximum size among those without redundant elements

- A **Redundant Mapping**
  - is a set containing redundant mapping elements

- The **Mapping of maximum size**
  - is the set containing the maximum number of mapping elements
  - It can be obtained from the propagation of the elements in the minimal set.
MinSMatch: computing the minimal mapping

- The **minimal mapping** always exists and it is unique
- Advantages in visualization, validation and maintenance
MinSMatch: the algorithm

Computing the minimal mapping $M$:

```plaintext
function TreeMatch(tree T1, tree T2) {
    TreeDisjoint(root(T1), root(T2)); (3)
    direction := true;
    TreeSubsumedBy(root(T1), root(T2)); (1)
    direction := false;
    TreeSubsumedBy(root(T2), root(T1)); (2)
    TreeEquiv(); (4) from (1) and (2)
}
```

Computing the set of maximum size:

```plaintext
function Propagate(M)
```
MinSMatch: evaluation w.r.t. SMatch

- We evaluated it on 4 datasets of different dimensions:
  - 34 x 39 (University courses)
  - 542 x 999 (Art domain)
  - 2857 x 6628 (Web directories)
  - 3358 x 5239 (Business directories)

- SAT calls: 43-66% less
- Runtime: 16-59% less
- Size of the minimal mapping: 68-96% less
- Recall: up to 0.6% elements more (*)

(*) We minimize the problem of lack of background knowledge; the deeper the classifications the better.
   The result of the propagation of the minimal set computed by MinSMatch is equivalent to the result of SMatch modulo inconsistencies.
Conclusions

- **The minimal mapping:**
  - always exists and it is unique
  - offers usability advantages in visualization, validation and maintenance

- **The MinSMatch algorithm:**
  - significantly faster w.r.t. SMatch
  - efficiently computes the mapping of maximum size (by propagation)
  - increased recall (the deeper the classifications the better)

- **Next steps:**
  - Experimenting MinSmatch on large scale knowledge organization systems (>400k nodes)
  - Avoid SAT
  - User interaction issues (navigation and validation tasks)
Questions

Search on google and Wikipedia: Minimal mappings

Contact info: maltese@disi.unitn.it