Logic-based Ontology Integration using ContentMap

Ernesto Jiménez-Ruiz[1]*, Bernardo Cuenca Grau[2],
Ian Horrocks[2], and Rafael Berlanga[1]

[1] Universitat Jaume I, Spain, {ejimenez,berlanga}@uji.es
[2] University of Oxford, UK, {berg,ian.horrocks}@comlab.ox.ac.uk

Abstract We present ContentMap, a system that uses a general method and novel algorithmic techniques to facilitate the integration of independently developed ontologies using mappings. Our method and techniques aim at helping users understand and evaluate the semantic consequences of the integration, as well as to detect and fix potential errors.

1 Motivation

Ontology integration techniques are often needed both during ontology development (e.g., when an ontology being developed reuses one or more external ontologies), and when ontologies are used in conjunction with data (e.g. when integrating and querying data sources annotated using different ontologies).

When the ontologies to be integrated have been independently developed, their vocabularies will most likely diverge, either because they use different names or naming conventions to refer to their entities. *Ontology Matching Techniques* [1] are intended to automatically discover the correspondences (i.e. *mappings*) between entities of different ontologies. This task is rather hard since in most cases there is not a common/similar nomenclature for the entity names.

Once the mappings (manually or automatically) have been generated the ontologies can be integrated. At this point, errors due to semantic incompatibility, may arise. There are two main causes for these errors. On the one hand, mappings suggested by automated tools are likely to include some errors. On the other hand, even if the correct mappings have been found (e.g. gold standard mappings), the ontologies may contain conflicting descriptions of the overlapping entities. These errors manifest themselves as unintended logical consequences (e.g. unsatisfiable concepts or wrong inferences), and they can be difficult to detect, understand and repair.

In [2] we presented a new general method and novel algorithmic techniques to obtain the right logical consequences when integrating ontologies using mappings. In this paper we focus on ContentMap (A logiC-based ONtology inTEgratioN Tool using MAPpings), the system we developed to provide just such support.

*This work has been partially supported by MICINN (TIN2008-01825/TIN) and Generalitat Valenciana (BPI06/372).

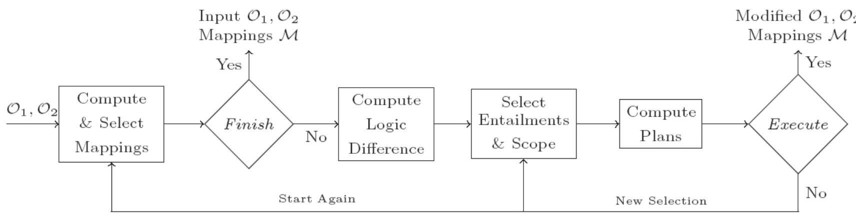

Input $\mathcal{O}_1, \mathcal{O}_2$
Mappings $\mathcal{M}$
Yes
$\mathcal{O}_1, \mathcal{O}_2$
Compute & Select Mappings
Finish
No
Compute Logic Difference
Select Entailments & Scope
Compute Plans
Execute
Modified $\mathcal{O}_1, \mathcal{O}_2$
Mappings $\mathcal{M}$
Yes
No
Start Again
New Selection

Figure 1. ContentMap Integration Method

2 Ontology Integration Method

We assume from now on that a set of mappings is represented as an OWL 2 [3] ontology $\mathcal{M}$, where mappings (often expressed as a tuple $\langle id, e_1, e_2, n, \rho \rangle$ [1]) are given as OWL 2 axioms of the form SubClassOf(e_1, e_2), EquivalentClasses(e_1, e_2), or DisjointClasses(e_1, e_2), for ρ of the form $(\sqsubseteq)$, $(\equiv)$, or $(\perp)$ respectively. e_1, e_2 are entity names in the vocabulary of $\mathcal{O}_1$ and $\mathcal{O}_2$ respectively, id is a unique identifier for the mapping, and n is a numeric confidence measure between 0 and 1 which is represented as an annotation axiom [3] of the mapping axiom.

Figure 1 shows the ontology integration method followed in ContentMap to evaluate and repair the logic consequences of merging two independent ontologies using mappings. The method can be split as follows:

1. Compute mappings $\mathcal{M}$ between $\mathcal{O}_1$ and $\mathcal{O}_2$ using a mapping algorithm, and filter them according a given criteria.
2. Compute logic difference and evaluate impact by comparing the entailments holding before and after the integration.
3. Detect unintended entailments and select them.
4. Compute repair plans and execute best one according to the user necessities.

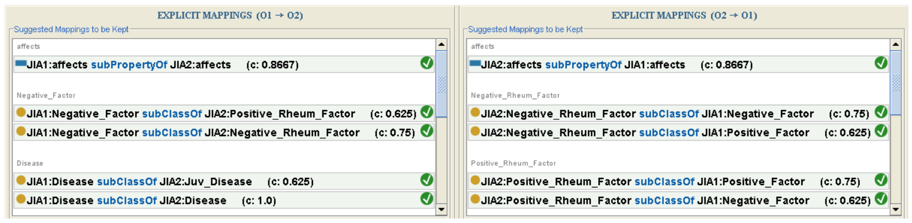
3 Underlying Techniques and **ContentMap** Support

Next we briefly comment the main underlying techniques used in ContentMap. For additional information about the methods and proofs refer to [2].

3.1 Computation of the Mappings

Ontology mappings can be computed using one or more mapping tools (in [2] we used for our experiments OLA [3], AROMA[4] and CIDER[5]). ContentMap loads pre-computed mappings in the form of an OWL 2 ontology and provides a GUI for visualising the mappings (see Figure 2). Users can either manually accept or reject mappings or automatically filter them by setting a confidence threshold.

[3]OWL Lite Alignment: http://ola.gforge.inria.fr/
[4]AROMA: http://www.inrialpes.fr/exmo/people/jdavid/
[5]CIDER: http://sid.cps.unizar.es/SEMANTICWEB/ALIGNMENT/



EXPLICIT MAPPINGS (O1 → O2)

Suggested Mappings to be Kept

affects

JIA1:affects subPropertyOf JIA2:affects (c: 0.8667) ✓

Negative_Factor

JIA1:Negative_Factor subClassOf JIA2:Positive_Rheum_Factor (c: 0.625) ✓

JIA1:Negative_Factor subClassOf JIA2:Negative_Rheum_Factor (c: 0.75) ✓

Disease

JIA1:Disease subClassOf JIA2:Juv_Disease (c: 0.625) ✓

JIA1:Disease subClassOf JIA2:Disease (c: 1.0) ✓

EXPLICIT MAPPINGS (O2 → O1)

Suggested Mappings to be Kept

affects

JIA2:affects subPropertyOf JIA1:affects (c: 0.8667) ✓

Negative_Rheum_Factor

JIA2:Negative_Rheum_Factor subClassOf JIA1:Negative_Factor (c: 0.75) ✓

JIA2:Negative_Rheum_Factor subClassOf JIA1:Positive_Factor (c: 0.625) ✓

Positive_Rheum_Factor

JIA2:Positive_Rheum_Factor subClassOf JIA1:Positive_Factor (c: 0.75) ✓

JIA2:Positive_Rheum_Factor subClassOf JIA1:Negative_Factor (c: 0.625) ✓

Figure 2. GUI for Visualising Explicit Mappings in ContentMap

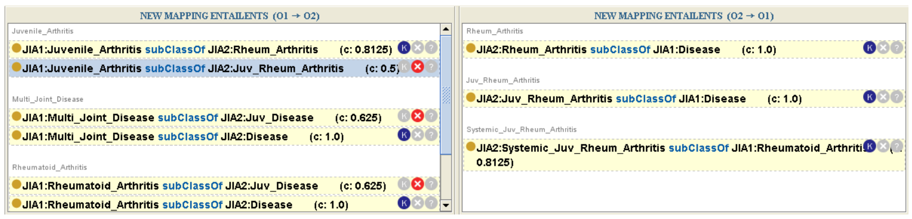
3.2 Computation of New Entailments and Error Detection

To help users understand the semantic consequences of the integration, they should be informed about new entailments that hold in the merged ontology $\mathcal{U}$, but not in $\mathcal{O}_1$, $\mathcal{O}_2$ and $\mathcal{M}$ alone. To this end, we use the notion of *logic difference* [4]. Intuitively, the logical difference between $\mathcal{O}$ and $\mathcal{O}'$ w.r.t a signature Σ is the set of entailments constructed over Σ that do not hold in $\mathcal{O}$, but do hold in $\mathcal{O}'$.

The notion of logic difference, however, has several drawbacks in practice. First, there is no algorithm for computing the logic difference in expressive DLs such as $\mathcal{SROIQ}$ (OWL 2) and $\mathcal{SHOIQ}$ (OWL DL) [4]. Second, the number of entailments in the difference can be huge (even infinite), and so likely to overwhelm users.

The GUI implemented in ContentMap allows users to customise approximations of the logic difference by selecting among the following simple kinds of entailment, where A, B are atomic concepts (including $\top, \bot$) and R, S atomic roles or inverses of atomic roles: *(i)* $A \sqsubseteq B$, *(ii)* $A \sqsubseteq \neg B$, *(iii)* $A \sqsubseteq \exists R.B$, *(iv)* $A \sqsubseteq \forall R.B$, and *v)* $R \sqsubseteq S$. The smallest implemented approximation considers only axioms of the form *(i)* (i.e. reasoner output), while the largest one considers all types *(i)*—*(v)*.

Figure 3 represents the ContentMap GUI where the entailments (for the selected logic difference approximation) are shown and can be selected to create $\Im^+$ (intended entailments to keep) and $\Im^-$ (unintended entailments to delete). ContentMap also provides a mechanism that automatically proposes candidate entailments to populate $\Im^+$ and $\Im^-$ sets.



NEW MAPPING ENTAILENTS (O1 → O2)

Juvenile_Arthritis

JIA1:Juvenile_Arthritis subClassOf JIA2:Rheum_Arthritis (c: 0.8125)

JIA1:Juvenile_Arthritis subClassOf JIA2:Juv_Rheum_Arthritis (c: 0.5)

Multi_Joint_Disease

JIA1:Multi_Joint_Disease subClassOf JIA2:Juv_Disease (c: 0.625)

JIA1:Multi_Joint_Disease subClassOf JIA2:Disease (c: 1.0)

Rheumatoid_Arthritis

JIA1:Rheumatoid_Arthritis subClassOf JIA2:Juv_Disease (c: 0.625)

JIA1:Rheumatoid_Arthritis subClassOf JIA2:Disease (c: 1.0)

NEW MAPPING ENTAILENTS (O2 → O1)

Rheum_Arthritis

JIA2:Rheum_Arthritis subClassOf JIA1:Disease (c: 1.0)

Juv_Rheum_Arthritis

JIA2:Juv_Rheum_Arthritis subClassOf JIA1:Disease (c: 1.0)

Systemic_Juv_Rheum_Arthritis

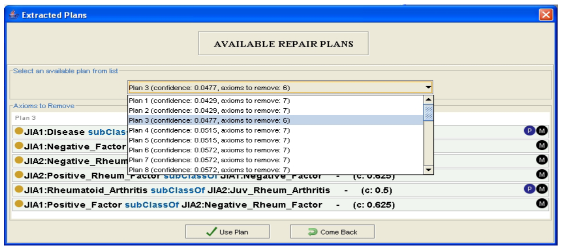
JIA2:Systemic_Juv_Rheum_Arthritis subClassOf JIA1:Rheumatoid_Arthritis (c: 0.8125)

Figure 3. GUI for Visualising New Entailments in ContentMap

3.3 Computation of Repair Plans

If the user has selected one or more unintended entailments (i.e., $\Im^- \neq \emptyset$), then $\mathcal{U}$ clearly contains errors. Errors could be due to erroneous mappings, to inherently conflicting information in the two ontologies, or to some combination of both. Repairing such errors might, therefore, require the removal of axioms from one or more of $\mathcal{M}$, $\mathcal{O}_1$ and $\mathcal{O}_2$. Additionally, any removal of axioms should also respect $\Im^+$. A repair plan for $\mathcal{U}$ given $\Im^+$ and $\Im^-$ is a set $\mathcal{P} \subseteq \mathcal{U}$ such that: *1)* $(\mathcal{O} \setminus \mathcal{P}) \models \alpha$ for each $\alpha \in \Im^+$, and *2)* $(\mathcal{O} \setminus \mathcal{P}) \not\models \beta$ for each $\beta \in \Im^-$. Notice that, our approach is related to existing approaches for revising mappings [5] and for debugging and repairing inconsistencies in OWL ontologies [6].

Figure 4 shows an example of a set of extracted repair plans, for a given selection of $\Im^-$ and $\Im^+$. When displaying plans, the GUI indicates whether the axioms in the plan come from $\mathcal{O}_1$, $\mathcal{O}_2$, or $\mathcal{M}$ (marked with '1', '2' and 'M' respectively). Additionally axioms shared by all plans are marked with a 'P'.



Plan 3 (confidence: 0.0477, axioms to remove: 6)

Plan 1 (confidence: 0.0429, axioms to remove: 7)
Plan 2 (confidence: 0.0429, axioms to remove: 7)
Plan 3 (confidence: 0.0477, axioms to remove: 6)
Plan 4 (confidence: 0.0515, axioms to remove: 7)
Plan 5 (confidence: 0.0515, axioms to remove: 7)
Plan 6 (confidence: 0.0572, axioms to remove: 7)
Plan 7 (confidence: 0.0572, axioms to remove: 7)
Plan 8 (confidence: 0.0572, axioms to remove: 7)

Plan 3
● JIA1:Disease subClas P M
● JIA1:Negative_Factor M
● JIA2:Negative_Rheum M
● JIA2:Positive_Rheum_Factor subClassOf JIA1:Negative_Factor - (c: 0.625) M
● JIA1:Rheumatoid_Arthritis subClassOf JIA2:Juv_Rheum_Arthritis - (c: 0.5) P M
● JIA1:Positive_Factor subClassOf JIA2:Negative_Rheum_Factor - (c: 0.625) M

Figure 4. Selection of available Repair Plans in ContentMap

References

1. Euzenat, J., Shvaiko, P.: Ontology matching. Springer-Verlag (2007)
2. Jimenez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga, R.: Ontology integration using mappings: Towards getting the right logical consequences. In: Proc. of European Semantic Web Conference (ESWC). Volume 5554 of LNCS., Springer-Verlag (2009) 173–187 Technical Report and Tool also available at: `http://krono.act.uji.es/people/Ernesto/contentmap`.
3. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The next step for OWL. J. Web Semantics **6**(4) (2008) 309–322
4. Konev, B., Walther, D., Wolter, F.: The logical difference problem for description logic terminologies. In: Proc. of IJCAR. (2008) 259–274
5. Meilicke, C., Stuckenschmidt, H., Tamilin, A.: Reasoning Support for Mapping Revision. Journal of Logic and Computation (2008)
6. Kalyanpur, A., Parsia, B., Sirin, E., Cuenca Grau, B.: Repairing unsatisfiable concepts in OWL ontologies. In: Proc. of ESWC. (2006) 170–184