# LogMap results for OAEI 2011

Ernesto Jiménez-Ruiz, Antón Morant, and Bernardo Cuenca Grau

Department of Computer Science, University of Oxford
{ernesto,anton.morant,berg}@cs.ox.ac.uk

**Abstract.** We present the preliminary results obtained by the ontology matching system LogMap within the OAEI 2011 campaign. This is the first participation of LogMap in the campaign, and the results have so far been quite promising.

## 1 Presentation of the System

The LogMap project started in January 2011 with the objective of developing a scalable and logic-based ontology matching system.

Such system should be able to deal efficiently with large-scale ontologies; furthermore, it should exploit logic-based reasoning and diagnosis techniques to compute output mappings that do not lead to logical inconsistencies when integrated with the input ontologies [7]. Although the development of LogMap is relatively recent, the authors' experience in the field of ontology integration dates back to 2008 [9, 11].

### 1.1 Motivation and Problem Statement

Despite the impressive state of the art, large-scale biomedical ontologies still pose serious challenges to existing ontology matching tools [15, 6].

**Insufficient scalability.** Although existing matching tools can efficiently deal with moderately sized ontologies (e.g. those in the OAEI Anatomy track), large-scale ontologies such as FMA, SNOMED CT and NCI are still beyond their reach.

**Logical inconsistencies.** OWL ontologies have well-defined semantics based on first-order logic, and mappings are commonly represented as OWL class axioms. Many existing tools, however, disregard the semantics of the input ontologies; thus, they are unable to detect and repair inconsistencies that logically follow from the union of the input ontologies and the computed mappings. Although there is a growing interesting in applying reasoning techniques to ontology matching, reasoning is known to severely aggravate the scalability problem.

### 1.2 Technical Approach

LogMap is a highly scalable ontology matching system with 'built-in' reasoning and diagnosis capabilities, which aims at addressing the aforementioned challenges.

We next present a brief overview of LogMap and refer the reader to [7] for a comprehensive description. The main steps performed by LogMap are schematically represented in Figure 1.

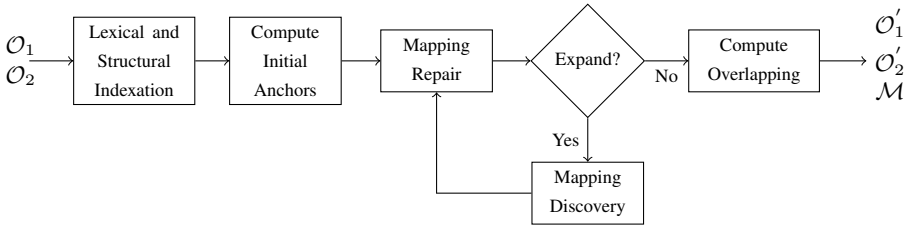


**Fig. 1.** LogMap in a nutshell.

| **Inverted index for NCI Anatomy labels** | | **Index for NCI Anatomy class URIs** | |
|---|---|---|---|
| ***Entry*** | ***Cls ids*** | ***Cls id*** | ***URI*** |
| external,ear | 1 | 1 | NCI_C12292 (external_ear) |
| atrial,auricle | 1,392 | 392 | NCI_C32165 (auricle) |
| auricle | 1,392; 529 | 529 | NCI_C12394 (ear) |
| ear | 529 | | |
| **Inverted index for Mouse Anatomy labels** | | **Index for Mouse Anatomy class URIs** | |
| ***Entry*** | ***Cls ids*** | ***Cls id*** | ***URI*** |
| auricle | 214 | 214 | MA_0000259 (auricle) |
| atrial,auricle | 214 | 216 | MA_0000258 (outer_ear) |
| ear,external | 216 | | |
| outer,ear | 216 | | |

**Table 1.** Fragment of the lexical indices for NCI and Mouse anatomy ontologies

**Lexical indexation.** The first step after parsing the input ontologies is their lexical indexation. LogMap indexes the labels of the classes in each ontology as well as their lexical variations, and allows for the possibility of enriching these indices by using external sources (e.g., WordNet or UMLS-lexicon) or a stemming algorithm (e.g., [14]). LogMap constructs an 'inverted' lexical index (see Table 1) for each input ontology. In general, an entry in the index can be mapped to several classes (e.g., see 'auricle' in Table 1). This type of index, which is commonly used in information retrieval applications, will be exploited by LogMap to efficiently compute an initial set of candidate mappings, called *anchors*.

**Structural indexation.** LogMap exploits the information in the (extended) class hierarchy of the input ontologies in different steps of the matching process. Efficient access to the information in these hierarchies is critical to LogMap's scalability.

LogMap classifies the input ontologies using either incomplete structural heuristics, or an off-the-shelf complete DL reasoner. Then, the classified hierarchies are indexed using an interval labelling schema—an optimised data structure for storing DAGs and trees [1], which has been shown to significantly reduce the cost of computing typical queries over large class hierarchies [3, 13].

| Entry | NCI ids | Mouse ids | Mappings |
|---|---|---|---|
| external,ear | 1 | 216 | NCI_C12292 ≡ MA_0000258 |
| atrial,auricle | 1,392 | 214 | NCI_C12292 ≡ MA_0000259<br>NCI_C32165 ≡ MA_0000259 |
| auricle | 1,392; 529 | 214 | NCI_C12292 ≡ MA_0000259<br>NCI_C32165 ≡ MA_0000259<br>NCI_C12394 ≡ MA_0000259 |

**Table 2.** Fragment of the intersection between the inverted indices for NCI and Mouse ontologies

The class hierarchies computed by LogMap are *extended* since, apart from the typical classification output of DL reasoners, they also include those explicit axioms in the input ontologies that can be directly encoded in Horn propositional logic (e.g., *class disjointness* axioms, subsumption axioms between an intersection of named classes and a named class).

**Computation of 'anchor mappings'.** LogMap computes an initial set of *anchor mappings* by intersecting the inverted indices of the input ontologies (i.e., by checking whether two lexical entries in those indices contain exactly the same strings). Anchor computation can hence be implemented very efficiently. Table 2 shows the intersection of the inverted indices of Table 1, which yields four anchors.

Given an anchor $m = (C_1 \equiv C_2)$, LogMap uses the string matching tool ISUB [16] to match the neighbours of $C_1$ in the hierarchy of $\mathcal{O}_1$ to the neighbours of $C_2$ in the hierarchy of $\mathcal{O}_2$. LogMap then assigns a confidence value to $m$ by computing the proportion of matching neighbours weighted by the ISUB similarity values. This technique is based on a *principle of locality*: if classes $C_1$ and $C_2$ are correctly mapped, then the classes semantically related to $C_1$ in $\mathcal{O}_1$ are likely to be mapped to those semantically related to $C_2$ in $\mathcal{O}_2$. Thus, if the hierarchy neighbours of the classes in an anchor mapping match with low confidence, then the anchor may be incorrect.

**Mapping repair and discovery.** The core of LogMap is an iterative process that alternates mapping repair and mapping discovery steps (see Figure 1).

Unsatisfiability checking and repair. LogMap uses a Horn propositional logic representation of the extended hierarchy of each ontology together with all existing mappings. Although such propositional Horn encoding is possibly incomplete, it is key to LogMap's scalability. Probably complete DL reasoners do not scale well when integrating large ontologies via mappings; the scalability problem is exacerbated by the number of unsatisfiable classes (more than 10,000 found by LogMap when integrating SNOMED and NCI using only anchors) and the large number of additional reasoner calls required for repairing each unsatisfiability.

For unsatisfiability checking, LogMap implements the highly scalable Dowling-Gallier algorithm [5] for propositional Horn satisfiability, and calls the Dowling-Gallier module once (in each repair step) for each class. Our implementation takes as input a

class $C$ (represented as a propositional variable) and determines the satisfiability of the propositional theory $\mathcal{P}_C$ consisting of

- the rule $(\mathsf{true} \rightarrow C)$;
- the propositional representations $\mathcal{P}_1$ and $\mathcal{P}_2$ of the extended hierarchies of the input ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$; and
- the propositional representation $\mathcal{P}_M$ of the mappings computed thus far.

LogMap computes a *repair* for each unsatisfiable class in the input ontologies. Given an unsatisfiable class $C$ and the propositional theory $\mathcal{P}_C$, a *repair* $\mathcal{R}$ of $\mathcal{P}_C$ is a minimal subset of the mappings in $\mathcal{P}_M$ such that $\mathcal{P}_C \setminus \mathcal{R}$ is satisfiable.

LogMap extends Dowling-Gallier's algorithm to record all *active mappings* ($\mathcal{P}_{act}$) that may be involved in each unsatisfiability. To improve scalability, repair computation is based on a 'greedy' algorithm. Given each unsatisfiable class $C$ and the relevant active mappings $\mathcal{P}_{act}$ computed using Dowling-Gallier, the algorithm identifies subsets of $\mathcal{P}_{act}$ of increasing size until a repair is found. Thus, our algorithm is guaranteed to compute all repairs of smallest size. If more than one repair is found, LogMap selects the one with involving mappings with the lowest confidence values.

Mapping discovery. In order to *discover new mappings*, LogMap maintains two *contexts* (sets of 'semantically related' classes) for each anchor. Contexts for the same anchor are expanded in parallel using the class hierarchies of the input ontologies. New mappings can then be found by matching classes in the relevant contexts using ISUB. Matches with a similarity value exceeding a given *confidence* threshold are considered as candidate mappings.

LogMap continues the iteration of repair and discovery steps until no context is expanded. The output of this process is a set of mappings that is likely to be 'clean', in the sense that it will not lead to unsatisfiable classes when merged with the input ontologies.

**Ontology overlapping estimation.** In addition to the mappings, LogMap can also returns two (hopefully small) fragments $\mathcal{O}'_1$ and $\mathcal{O}'_2$ of $\mathcal{O}_1$ and $\mathcal{O}_2$, respectively. Intuitively, $\mathcal{O}'_1$ and $\mathcal{O}'_2$ represent the 'overlapping' between $\mathcal{O}_1$ and $\mathcal{O}_2$, in the sense that each 'correct' mapping not found by LogMap is likely to involve only classes in these fragments. The computation of $\mathcal{O}'_1$ and $\mathcal{O}'_2$ is performed in two steps.

1. *Computation of 'weak' anchors*. LogMap computed the initial anchor mappings by checking whether two entries in the inverted index of $\mathcal{O}_1$ and $\mathcal{O}_2$ contained *exactly the same* set of strings. For the purpose of overlapping estimation, LogMap also computes new anchor mappings that are 'weak' in the sense that relevant index entries are only required to contain *some* common string. Thus, weak anchors represent correspondences between classes with a common lexical component.
2. *Module extraction*. The sets $S_i$ of classes in $\mathcal{O}_i$ involved in either a weak anchor or a mapping computed by LogMap are then used as 'seed' signatures for module extraction. In particular, $\mathcal{O}'_1$ (resp. $\mathcal{O}'_2$) are computed by extracting a locality-based module [4] for $S_1$ in $\mathcal{O}_1$ (resp. for $S_2$ in $\mathcal{O}_2$).

### 1.3 Adaptations made for the evaluation

To participate in the OAEI 2011, LogMap has been extended with a property matching facility as well as with the ability to consider 'weak' anchors as candidate mappings.

**Computation of property anchors.** Similarly to the case of anchor mappings between classes, the computation of anchor mappings between (object or data) properties also relies on the intersection of inverted lexical indexes. These mappings, however, are currently not taken into account by LogMap's repair module.

In the current version of LogMap, a mapping between properties $p_1$ and $p_2$ is returned as output only if both their respective domains $D_1, D_2$ and ranges $R_1, R_2$ are 'compatible'— that is, if LogMap's repair module does not find inconsistencies when extending the final output class mappings with the mappings $D_1 \equiv D_2$ and $R_1 \equiv R_2$.

For example, in the OAEI conference track, LogMap identified an equivalence mapping between the properties cmt:writtenBy and confOf:writtenBy. This mapping, however, was discarded since the extension of LogMap's output class mappings with the mappings cmt:Reviewer $\equiv$ confOf:Author and cmt:Review $\equiv$ confOf:Contribution between the respective domains and ranges of these properties led to an inconsistency.

**Inclusion of 'weak' anchors.** Weak anchor mappings are well-suited for overlapping estimation purposes (see Section 1.2); however, it is dangerous to treat them as candidate output mappings since they are likely to introduce unmanageable levels of 'noise' during mapping repair.

The upper part of Table 3 shows an excerpt of the inverted indices for NCI and Mouse Anatomy ontologies extended with partial lexical entries. The intersection of these inverted indices includes the entry *'smooth,muscle'*, which appears in 19 concepts in Mouse Anatomy and in 9 concepts in NCI Anatomy; as a result, 171 weak anchor mappings can be obtained (see lower part of Table 3). Most of these mapping are obviously incorrect (e.g. NCI_C49483 $\equiv$ MA_0001741 or NCI_C49306 $\equiv$ MA_0001741), however valid mappings can still be discovered (e.g. NCI_C49483 $\equiv$ MA_0001635).

The current version of LogMap considers a weak anchor as a candidate output mapping (hence taking it into account for mapping repair) only if exceeds a given ISUB confidence threshold.

## 2 Results

In this section, we present the preliminary results obtained by LogMap for the OAEI tracks provided by the SEALS client: Anatomy 2010, Conference 2010, and Benchmark 2011.[1] Tests were performed using a laptop computer with 4 Gb of RAM.

[1] Final results will be provided in the workshop proceedings.

| **Extended inverted index for NCI Anatomy** | | **Index for NCI Anatomy class URIs** | |
|---|---|---|---|
| ***Lexical entry*** | ***Cls ids*** | ***Cls id*** | ***Cls name*** |
| gallbladder,smooth,tissue,muscle | 2061 | 2061 | NCI_C49483 (gallbladder_smooth_muscle_tissue) |
| smooth,muscle | 2061; 3214,... | 3214 | NCI_C49306 (trachea_smooth_muscle_tissue) |
| **Extended inverted index for Mouse Anatomy** | | **Index for Mouse Anatomy class URIs** | |
| ***Lexical entry*** | ***Cls ids*** | ***Cls id*** | ***Cls name*** |
| gall,bladder,smooth,muscle | 600 | 600 | MA_0001635 (gall_bladder_smooth_muscle) |
| smooth,muscle | 600; 2629; ... | 2629 | MA_0001741 (prostate_gland_smooth_muscle) |

| **Entry** | **NCI ids** | **Mouse ids** | **Mappings** |
|---|---|---|---|
| smooth,muscle | 2061; 3214; ... | 600; 2629; ... | NCI_C49483 $\equiv$ MA_0001635<br>NCI_C49483 $\equiv$ MA_0001741<br>NCI_C49306 $\equiv$ MA_0001635<br>NCI_C49306 $\equiv$ MA_0001741<br>... |

**Table 3.** Extend inverted indices an their intersection for NCI and Mouse Anatomy ontologies

| **Systems** | **Precision** | **Recall** | **F-score** |
|---|---|---|---|
| ***LogMap*** | ***0.938*** | ***0.836*** | ***0.884*** |
| AgrMaker | 0.903 | 0.853 | 0.877 |
| Ef2Match | 0.955 | 0.781 | 0.859 |
| NBJLM | 0.920 | 0.803 | 0.858 |
| SOBOM | 0.949 | 0.778 | 0.855 |
| BLOOMS | 0.954 | 0.731 | 0.828 |

**Table 4.** Comparing LogMap with the top 5 tools in the Anatomy Track of the OAEI 2010

### 2.1 Anatomy 2010 Track

This track involves two biomedical ontologies: the Adult Mouse Anatomy ontology (2,744 classes) and a fragment of the NCI ontology describing human anatomy (3,304 classes). The reference alignment [2] has been manually curated, and it contains a significant number of non-trivial mappings.

Table 4 compares LogMap's results with the top 5 tools in the Anatomy 2010 track. LogMap was faster than the other tools: while LogMap matched these ontologies in $44.5$ seconds, tools like AgrMaker and SOBOM required $23$ and $19$ minutes respectively.[2] We verified, using an off-the-shelf DL reasoner, that the integration of these ontologies with LogMap's output mappings did not lead to unsatisfiable classes.

### 2.2 Conference 2010 Track

The Conference 2010 Track contains 16 ontologies describing the domain of conference organisation.

Table 5 compares LogMap's results with the official results obtained by the top 5 tools in 2010. LogMap obtained the best results in terms of F-score; furthermore, reasoning with the union of the input ontologies and LogMap's output mappings did not lead to unsatisfiable classes; thus, the degree of incoherence [12] of LogMap's output mappings was 0%.

[2] These times correspond to the 2009 OAEI results since no official times were given in 2010.

| Systems | Precision | Recall | F-score | Incoherence |
|---|---|---|---|---|
| ***LogMap*** | ***0.85*** | ***0.53*** | ***0.66*** | ***0%*** |
| CODI | 0.86 | 0.48 | 0.62 | 0.1% |
| ASMOV | 0.57 | 0.63 | 0.60 | 5.6% |
| Ef2Match | 0.61 | 0.58 | 0.60 | 7.2% |
| Falcon | 0.74 | 0.49 | 0.59 | >4.8% |
| AgrMaker | 0.53 | 0.62 | 0.58 | >14.8% |

**Table 5.** Comparing LogMap with the top 5 tools in the Conference Track of the OAEI 2010.

### 2.3 Benchmark 2011 Track

The goal of this track is to evaluate the tools' behaviour when the input ontologies are lacking important information. The test ontologies for this track have been obtained by performing certain synthetic transformations on realistic ontologies (e.g., suppressing entity labels, flattening the class hierarchy).

The computation of candidate mappings in LogMap heavily relies on the similarities between the lexicons of the input ontologies; hence, replacing entity names by random strings has a direct negative impact in the number of discovered mappings.

When taking into account only those tests for which LogMap was able to compute at least one mapping, we obtained an average precision of 0.992 and an average recall of 0.605. In 17 (out of 112) test cases, however, LogMap found no mappings. When taking into account also these cases, we obtained average precision and recall values of 0.827 and 0.504, respectively.

## 3 General Comments and Conclusions

**Comments on the results.** We find LogMap's results quite promising.

- In all cases, LogMap was able to compute a clean set of output mappings (i.e., not leading to unsatisfiable classes when merged with the input ontologies).
- LogMap was the fastest of all tools in the the Anatomy 2010 Track (computationally, the most challenging of all tracks).
- LogMap obtained the best results in terms of F-score for both the Anatomy 2010 and Conference 2010 tracks.

LogMap's main weakness is that the computation of candidate mappings relies on the similarities between the lexicons of the input ontologies. As already mentioned, LogMap could not find any mappings for 17 of the test cases in the Benchmark 2011 track, since class names were substituted by random strings.

**Comments on the OAEI 2011 test cases.** Ontology matching tools have significantly improved in the last few years, and there is a need for more challenging and realistic matching problems [15, 6]. To address this need, in [10, 8] we proposed the use of (clean subsets of) UMLS mappings as reference alignments between the large-scale biomedical ontologies FMA, SNOMED CT and NCI. The use in an OAEI track of

these ontologies represents a significant leap in complexity w.r.t. the existing anatomy track; however, we take our positive experiences with LogMap as an indication that a new track based on these ontologies and their UMLS alignments would be feasible.

**Acknowledgements.**

We would like to acknowledge the funding support of the Royal Society and the EPSRC project *LogMap*, and also thank the organizers of the OAEI campaign for providing test data and infrastructure.

## References

1. Agrawal, R., Borgida, A., Jagadish, H.V.: Efficient management of transitive relationships in large data and knowledge bases. SIGMOD Rec. 18, 253–262 (1989)
2. Bodenreider, O., Hayamizu, T.F., et al.: Of mice and men: Aligning mouse and human anatomies. In: AMIA Annu Symp Proc. pp. 61–65 (2005)
3. Christophides, V., Plexousakis, D., Scholl, M., Tourtounis, S.: On labeling schemes for the Semantic Web. In: Proc. of WWW. pp. 544–555. ACM (2003)
4. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Just the right amount: extracting modules from ontologies. In: Proc. of WWW. pp. 717–726 (2007)
5. Dowling, W.F., Gallier, J.H.: Linear-time algorithms for testing the satisfiability of propositional Horn formulae. J. Log. Program. pp. 267–284 (1984)
6. Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., Trojahn, C.: Ontology Alignment Evaluation Initiative: six years of experience. J Data Semantics (2011)
7. Jimenez-Ruiz, E., Cuenca Grau, B.: LogMap: Logic-based and Scalable Ontology Matching. In: et al., L.A. (ed.) The 10th International Semantic Web Conference (ISWC). LNCS, vol. 7031, pp. 273–288. Springer (2011)
8. Jimenez-Ruiz, E., Cuenca Grau, B.: Towards more challenging problems for ontology matching tools. In: Proc. of the 3th International Workshop on Ontology Matching (OM) (2011)
9. 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). pp. 173–187 (2009)
10. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga, R.: Towards a UMLS-based silver standard for matching biomedical ontologies. In: Proc. of the 5th International Workshop on Ontology Matching (2010)
11. Jiménez-Ruiz, E., Cuenca Grau, B., Horrocks, I., Berlanga, R.: Logic-based assessment of the compatibility of UMLS ontology sources. J Biomed. Sem. 2 (2011)
12. Meilicke, C., Stuckenschmidt, H.: Incoherence as a basis for measuring the quality of ontology mappings. In: Proc. of the 3rd International Workshop on Ontology Matching (2008)
13. Nebot, V., Berlanga, R.: Efficient retrieval of ontology fragments using an interval labeling scheme. Inf. Sci. 179(24), 4151–4173 (2009)
14. Paice, C.: Another stemmer. SIGIR Forum 24(3), 56–61 (1990)
15. Shvaiko, P., Euzenat, J.: Ten challenges for ontology matching. In: On the Move to Meaningful Internet Systems (OTM Conferences) (2008)
16. Stoilos, G., Stamou, G.B., Kollias, S.D.: A string metric for ontology alignment. In: Proc. of the International Semantic Web Conference (ISWC). pp. 624–637 (2005)