# Tools for ontology matching—practical considerations

Maria Ganzha[1,4], Marcin Paprzycki[1], Wiesław Pawłowski[2], Paweł Szmeja[1], Katarzyna Wasielewska[1], and Giancarlo Fortino[3]

[1] Systems Research Institute, Polish Academy of Sciences
*firstname.lastname*@ibspan.waw.pl
[2] Faculty of Mathematics, Physics, and Informatics, University of Gdańsk
wieslaw.pawlowski@inf.ug.edu.pl
[3] Dept of Informatics, Modeling, Electronics and Systems, University of Calabria
giancarlo.fortino@unical.it
[4] Warsaw University of Technology

**Abstract.** There exists a large body of scientific literature devoted to ontology matching, aligning, mapping translating and merging. With it, comes a long list (90+) of tools that support various aspects of these operations. We have approached such tools from the perspective of the INTER-IoT project, in which one of the goals is to facilitate semantic interoperability of Internet of Things platforms. Thus, we had to answer a question: what is *actually* available when one needs to align/merge ontologies. Here, we summarize our findings.

**Keywords:** ontology, ontology aligning, ontology merging, Internet of Things

## 1 Introduction

Today, research concerning ontology alignment, matching, merging, mapping, translating remains popular among researchers. In [21], 694 recent papers from that area have been meticulously selected and evaluated, clearly showing that not only the activity level remains high, but that the interest is growing. Furthermore, in Section 4.3 and in Table 5 of [21], a list of approximately 60 tools, deadling with various aspects of ontology mapping, by using different approaches, has been compiled. This suggests that the area is maturing. As a matter of fact, one of the key conclusions, from this paper, was a "positive outlook" for ontology engineering, seen both from the academic and practical perspectives. Moreover, when we have extended our research to tools/platforms found in other sources, such as [10], websites [1,2], as well as resulting from general Web searches, a total of 97 references were gathered.

Our interest in ontology engineering comes from the INTER-IoT project [4]. Its goal is to provide novel solutions to the lack of interoperability among Internet of Things (IoT) platforms. While the project, as a whole, is to facilitate interoperability across the hardware-software stack, in the context of this contribution we

are interested in the highest layer, semantic, interoperability. The background of our work is provided by two use case scenarios. First, involves joining two eHealth (IoT) platforms. Second, deals with joining an IoT platform from a sea port terminal, supervising container management, with an IoT platform of a logistics company that deals with truck fleet management. In both cases, semantic interoperability involves instantiating a "common ontology". Here, it should be stressed that the actual way, in which the common ontology will be instantiated, is likely to be context and system architecture dependent. Nevertheless, to be able to achieve this goal, we would like to use (if only possible) already existing tools. Hence, we have decided to establish, which of the existing 97 tools can be immediately (or after small modifications) applied to the problem we are facing. The aim of this paper is to summarize results of our investigations.

The main contribution of this work is to go beyond the optimism expressed in [21] and provide a realistic assessment of *tools that are actually available today* (in 2016), their capabilities, and limitations. In this way, unless one of "defunct-tools" is resurrected, only tools listed here (and new ones, created afterwards) need to be considered by practitioners who align/merge ontologies.

To this effect we proceed as follows. In Section 2 we provide definitions of key terms used in the paper (and relations between them). We follow, in Section 3, with brief summary of the two use case scenarios. These two sections provide the foundation for the analysis of the state-of-the-world of tools for ontology alignment/matching/merging/mapping/translating, A.D. 2016. Here, we first (in Section 4) reflected on how someone who needs a specific job done would look at these tools. As a result, by approaching tools from a very pragmatic perspective, 7 tools were selected as worthy further evaluation. In Section 5, we present more detailed description of each of them. Next, we present a high-level reflection on the problem of "computing" the common semantics, that we have to address, and for which finding ontology alignments is just the first step (Section 6). Section 7, summarizes our key findings.

## 2 Definitions

Let us start from defining key terms. In the context of semantic interoperability, topics such as ontology alignment, matching, merging and mapping need to be disambiguated. These terms are closely related and sometimes used interchangeably. For each, there are many, sometimes overlapping, definitions. For the purpose of clarity, in the scope of this article, we use the following definitions (see, also, [10]).

***Ontology alignment***, refers to *finding correspondences between two or more ontologies*. The result of this process is an *alignment*—a set of *correspondences* between entities (atomic alignment) or groups of entities and sub-structures (complex alignment) from different ontologies. A correspondence can be either a predicate about similarity, called a *matching*, or a logical axiom—a *mapping*. Typically used mapping axioms are equivalence and subsumption. In practice, ontology alignment tools often state a degree of confidence for every correspon-

dence in the mapping. An equivalence axiom with a degree of confidence is very close in meaning to a predicate about similarity (a matching). The terms "mapping" and "matching" are often not distinguished in the terminology used by the alignment tools. A set of correspondences can be called "alignment", "matching", or "mapping" practically interchangeably.

***Ontology merging*** is a *process of combining two, or more, ontologies into one*. Consequently, the resulting ontology stores knowledge from all merged ones. Merging often utilizes a set of alignments to create deep interconnections between ontologies and, in the end, merge them into one.

Finally, ***ontology translation***, or, more precisely, *semantics translation*, is a *process of changing the underlying semantics of a piece of knowledge*. Given some information described semantically, in terms of a source ontology, it is transformed into information described in terms of a target ontology. Resulting information contains no references to source semantics, instead it has only target semantics. In a good translation, the meaning should be preserved. Semantics translation is an application of ontology alignment. The goal is to enable one-way or two-way "understanding" between software artifacts that implement differing semantics. This is directly applicable to multiple domains such as IoT, bioinformatics and others, because there are competing ontologies that describe the same or very similar area of knowledge. For instance, the IoT ontologies: OpenIoT [5], SAREF [6], and oneM2M [7], have very similar scope. Therefore, it is reasonable to expect that good alignments may be found between them. It is unreasonable, however, to expect each "hybrid IoT" system to implement semantics originating from all of those (and possibly other) ontologies. Therefore, semantics translation is a practical endeavor.

## 3 INTER-IoT use cases

The context of our work is provided by the two large-scale pilots that are the core of the INTER-IoT project:

- ***(e/m)Health*** The goal is to facilitate interoperability between two heterogeneous IoT platforms—one for remote use of *non-wearable devices*, and another for devices organized in a *body sensor network*. Both platforms use cloud infrastructure and Bluetooth technology to interact with measuring devices. However, their technologies are different (and thus are not interoperable). On the data and semantics level, one platform exposes JSON web services for third party systems, whereas the other utilizes Google Datastore API. While both platforms gather data with intuitively similar semantic meaning (e.g. temperature, blood pressure) they store them in different formats, and use somewhat different semantics. For example, in one platform temperature is stored in an attribute *Temp* while in the other in an attribute

[5] https://github.com/OpenIotOrg/openiot
[6] https://sites.google.com/site/smartappliancesproject/ontologies/reference-ontology
[7] http://www.onem2m.org/technical/onem2m-ontologies

*BodyTemperature*. Integration of the two platforms should result in a comprehensive mHealth system, where querying for patient's "parameters" should provide the physician with measurements gathered by both platforms. It should be noted that the solution should be extendable to include additional platforms with different data formats and semantics (e.g. an ambulance fleet IoT platform, with its own set of on-board measuring devices).

- ***Transportation and logistics*** The goal is to provide interoperability between at least two IoT platforms in the Port of Valencia (Spain). Platforms that gather sensors data used in container management during loading, unloading and shipment, differ in architecture and technology. However, on the data and semantics level they share some common concepts, e.g. *virtual containers* and *virtual trucks*. At different stages of container management lifecycle, container is controlled and monitored by different systems, e.g. haulier company, container terminal, carrier, consignee. They all store information about containers, but use different data formats and semantics. To be able to exchange information and jointly monitor the situation, common understanding (e.g. of virtual container) is needed. Similarly, when the port haulier company subcontracts shipment services from another company, they want to temporarily consider them as part of their fleet. To exchange data and jointly monitor the flow of transport (e.g. truck position), a common understanding (e.g. of virtual truck) is needed.

In [16], we have discussed semantic representation, currently available in the two areas. The plenitude of models, standards and ontologies (that exist, are in use or under development), as well as semantic and syntactic heterogeneity of data, pose a serious challenge for interoperability. Note, that these (available) models describe only selected aspects of their domains. There is no single comprehensive ontology for (e/m)Health, or for transportation and logistics. Therefore, to adequately semantically represent a domain one needs to deal with multiple ontologies, and consider different data formats and ontology languages used to model parts of the domain. Furthermore, additional ontologies, defining concepts related to various aspects of IoT platforms, also need to be incorporated.

## 4 Available research results and tools—summary

Let us now look into tools that can be used in our work. Before proceeding, let us stress that, we are interested *only* in "operations performed on ontologies". In other words, we assume that either (which is unlikely) IoT platforms that are to interoperate use ontologies represented in RDF/OWL, or "extraction of semantics" (e.g. from XML, JSON, etc.) has been performed, and RDF/OWL ontologies created as a result. Now, ontology aligning has to take place.

### 4.1 Criteria for tool selection

First, let us stress that we do not approach/classify tools on the basis of the underlying algorithms/methods. This has been already done (see, for instance, [21]).

Instead, we propose very pragmatic criteria that are essential when selecting methods for application in real-life use cases:

- availability of the website and the date of the last update—presence on the web site and "recent" date of the last update show vitality of the tool—as a matter of fact, tools that have not been updated for more than 2 years are very suspicious from the point of view of lock-down to a dead-end software,
- number of related publications and date of last publication—larger number of publications indicates that the method is better established (has been reviewed more often), while date of last publication (again) indicates vitality,
- availability of the source code and documentation—crucial for actual use,
- used technology, and I/O data format—indicate what levels of expressiveness can be handled by the method, and what input/output data can be processed; here we also consider the interfaces (GUI and/or command line),
- known academic and commercial utilization—it is very valuable when a method was applied outside of a purely "academic environment",
- scalability—use in the IoT requires tools that are scalable and efficient.

Overall, we are interested in tools that are mature (went though a number of development cycles and resulted in multiple publications), actively maintained and systematically developed, preferably have been applied in real-life scenarios, and bring some promise of scalability.

### 4.2 Filtering tools found in the literature

We have investigated all methods/tools mentioned in [21,1,2], as well as tools found as a result of internet search, a total of *97*, taking into account criteria from Section 4.1. As a result we have reached the following conclusions.

- While numerous tools, implementing ontology matching, appear in the literature, most of them are defunct. We have identified only *nine* that are still alive and more or less correspond to our needs (see, Section 5).
- For $\sim$60% of the tools, we have not found an active website. In many cases, if the website was available, it was very basic and not recently updated.
- We have observed a tendency to present mostly OAEI contest results. While the OAEI initiative is very useful when comparing and evaluating methods, lack of other/follow-up publications shows that the method/tool was developed primarily to participate in the contest.
- Besides few cases, we have not found information about tool's use in projects or commercial applications. Almost all documented use cases came from the OAEI contests.
- Scalability can be deduced only from the results of the OAEI contests. We have not found other results explicitly benchmarking scalability of the methods.
- For $\sim$85% of the tools, we could not find either source code or executables. For the remaining $\sim$15%, significant part had no technical documentation, or user manual. Instead, only tools/matching methods were described in publications.

- In ∼85% of cases, explicitly stated description of what are the input/output ontologies formats and languages was missing.
- Almost no tool seriously considered situation when the input semantics is not explicitly represented in one of the core ontology languages (RFD/OWL). However, lack of explicit formal RDF/OWL ontology is a typical situation for the ICT systems of today (e.g. use cases of the INTER-IoT project).

## 5 Working tools

Let us now look into more details of tools that met our criteria. Note that, in addition to the seven listed below, there are two more "active tools" that could have been listed in this section: YAM++ [9][8] and LODE [9]. However, YAM++ was omitted because there is no source code available (only executables, that cannot be modified, if needed); while LODE is accessible only as a web application (it lacks of a command line interface, or an API).

### 5.1 LogMap

LogMap [18,22][10] is an open-source tool, developed at the University of Oxford. It can match very large ontologies, such as FMA and SNOMED. Since 2011, LogMap takes part in the OAEI contests, constantly achieving very good results. In 2015, it was the only tool taking part in all OAEI tracks.

LogMap was written in Java, and can be used both from the command-line and via a web-based Ajax interface[11]. The command-line version is available as a stand-alone distribution, as well as in the form of the OAEI packages. As input, the tool accepts any of the OWL API formats, and produces alignments between *classes*, *properties*, and *instances*. As one of very few ontology matching tools, LogMap provides an on-the-fly inconsistency repair capabilities. For consistency checking, it utilizes a method based on propositional Horn-clause satisfiability (Dowling-Gallier algorithm [8]). The source code (last updated in May 2016) is freely available from the *GitHub*. Pre-build packages can be downloaded from the *SourceForge*.

Certain weakness of LogMap lies in the way it computes the candidate mappings/matches. The algorithm finds similarities between concepts, utilizing vocabularies of the input ontologies. Therefore, the result may not be satisfactory if the ontologies are (seriously) lexically disparate, or do not provide enough lexical information.

The website of LodMap lists 11 publications, devoted to various aspects of the tool, with the most recent from 2016.

[8] http://www.lirmm.fr/yam-plus-plus/
[9] http://lode.informatik.uni-mannheim.de/
[10] https://www.cs.ox.ac.uk/isg/tools/LogMap/
[11] http://csu6325.cs.ox.ac.uk/

### 5.2 COMA

COMA 3.0 [5][12] (previously called GOMA or COMA++) is a framework that supports various matching algorithms and is highly customizable. It is an open source project (code last updated in January 2013) that evolved from the work done at the University of Leipzig. The tool performs matching and merging on XSD (XML Schema), OWL (OWL-Lite), XDR (XML Data Reduced) and relational database schemas. Internally, any supported data format is transformed into a generic model of a directed acyclic graph, which enables processing of schemas and ontologies distributed among multiple namespaces and files. COMA has full GUI support for all its operations.

COMA implements an iterative algorithm based on a collection of matching algorithms (matchers). Selection of matchers, as well as decisions, which matching axioms are correct, is made by the user. Specifically, the user assigns a confidence value to each matching axiom, and can manually create and delete them. Any number of iterations (computing and refining matching axioms) can be performed, each building on the result of previous one. The end result can be saved to a file in a COMA specific format. COMA can use the resulting matching to create, among others, merged ontology, intersection of ontologies, etc. Merging of ontologies and schemas is limited to the, paid, Business Edition of COMA.

Because of its architecture, COMA is a good candidate for a framework for implementation and testing of new matchers. Lack of support for RDF, or more expressive profiles of OWL, are a limiting factor.

### 5.3 AgreementMaker

AgreementMakerLight [13][13] (AML, a continuation of AgreementMaker and part of the SOMER project) is an automated matching system that acts as an extensible framework that implements many matchers. It is open source and actively updated. Initially, the AgreementMaker was specialized to work with biomedical ontologies but, currently, it can be applied to any ontology in OWL, OBO or SKOS format. It has performed very well in 2014 [11] and 2015 [12] editions of the OAEI competition. It is claimed that the AML can efficiently (i.e. within several minutes) compute alignments on very large ontologies (e.g. WordNet), although it (understandably) requires large amounts of RAM (e.g. 8GB for ontologies with less than 100 000 classes).

Currently, AML implements 6 matchers that range from simple (label similarity) to complex (so-called, structural matcher), as well as 3 filters (e.g. cardinality filter). Each matcher is configurable, e.g. the string matcher has a choice of four similarity measures. Background knowledge matcher can calculate similarity scores by using an *external* knowledge source, like WordNet. However, it supports matchings between classes and properties, but not individuals. Alignment

[12] http://dbs.uni-leipzig.de/Research/coma.html
[13] http://somer.fc.ul.pt/aml.php

can be reviewed and each axiom is explained on a graph. Alignment axioms may also be added or removed manually. The results are in the Alignment API [7] format.

The AML can work both as a GUI and as a command line application. The possibility to extend the framework with new matchers is very valuable. Unfortunately, the AML does not natively perform ontology merging.

### 5.4 Alignment API

Alignment API [7][14] is a definition of format (and schema) for storing alignments in RDF, and a set of tools that operate on them. It is designed to be tool-agnostic and to enable storing, exchanging, and sharing alignments. The API itself, outside of simple reference implementations, does not define any matchers, nor does it provide matching or merging services for ontologies or schemas. Instead, it defines a set of standard operations and interfaces for working with alignments. Alignment API specification and tools are actively updated and open source.

An Alignment Server (part of the Alignment API) can store, compare and manage alignments. It can be accessed via pluggable interfaces that currently include: HTTP, SOAP and REST web services and FIPA ACL[15]. The server allows information about the alignment computation process (e.g. program/matcher name, processing time) to be stored in the alignment file. The format is extensible, so any kind of additional information can be added and the schema itself can be extended.

The API defines interfaces for matching algorithms, query translation, finding existing alignments, manipulating alignments, rendering them in a different language, etc. Alignment Server provides a reference implementation of those operations, but for specific problems, own implementations are encouraged.

The official webpage lists close to a hundred tools that are compatible with the Alignment API (including some we list in this article).

### 5.5 Silk Framework

Silk Framework [23][16] is an open source tool for discovery of links between datasets in the context of the Open Linked Data. It generates links between sources, based on user-provided link specifications. The supported formats include RDF, CSV and XML, with strong focus on RDF. Querying of data is done through a user-specified SPARQL endpoint. Link specifications can be written manually in Silk-LSL (Link Specification Language), or constructed in the Silk Workbench—a Java web application. They can be exported and incorporated into original data sets. Results produced by the Silk can be stored in an Alignment API compatible format.

[14] http://alignapi.gforge.inria.fr/
[15] Jade (http://jade.tilab.com/) Agent Communication Language
[16] http://silkframework.org/

As opposed to the schema matching systems, Silk discovers and verifies, links between data values and nodes. Support for any SPARQL endpoint means that large amounts of data, spread among SPARQL datasets, can be queried, with full interlinking between different graphs and namespaces. Furthermore, Silk allows defining complex data transformations that go well beyond simple `owl:sameAs` links. This can be useful in translation of data between semantics.

Manual input of link specifications means that links cannot be discovered entirely automatically—one needs to specify what kind of linkage pattern (s)he is looking for. In this way, Silk is more of a tool to interlink data, rather than to discover alignments. Note that, Silk does not perform automatic schema matching or ontology merging.

### 5.6 S-Match

S-Match [17][17] is an open source semantic matching framework that transforms tree-like structures such as catalogs, conceptual models, etc., into lightweight ontologies to then determine the semantic correspondences between them. The project has an up-to-date website with information, including documentation, and tutorials. There are over 20 papers (last from 2011) devoted to various aspects of the project e.g. algorithms implementation.

S-Match is a Java application that can be run from GUI or from command line. The input to the method are text files, in which tree like structures are defined. Using a native input format is one of disadvantages of the tool. Here, input ontologies have to be transformed before running the tool.

The source code is available from *GitHub*(last updated in January 2015). Ready to use pre-build packages (most recent from 2013), can be downloaded from *SourceForge*. S-Match was utilized in 10 documented projects that are referenced on the website.

### 5.7 OntoBuilder

OntoBuilder [18] project provides an open source set of tools to extract (generate) ontologies from web pages and map ontologies from similar domains, generating an ever-improved single ontology, with which a domain can be queried. OntoBuilder services for schema matching provide several algorithms e.g. similarity flooding, combined algorithm, precedence algorithm, term and value combined algorithm, graph algorithm, value algorithm, term algorithm. The Top K Framework graphical tool allows to view and save best mappings (based on a user-defined threshold). OntoBuilder is written in Java and can be used as a graphical tool, as a jar package, or as a command line tool. The source code and documentation are freely available from the *Bitbucket*repository.

Even though the last publication is from 2010, and last update to the website with downloadable OntoBuilder was done in June 2011, the tool is well documented with 15 publications linked from the website. We suspect that the project

[17] http://semanticmatching.org/
[18] http://iew3.technion.ac.il/OntoBuilder/

is not actively developed (making it the "weakest" of the seven). However, the deliverables produced in the past can provide useful input for our work.

## 6 From alignments to common semantics

In the project, we need to consider alignments between ontologies from possibly different domains, so that the "content", expressed in semantics of one platform, can be translated to/understood by other IoT platform(s) (e.g. monitoring virtual container/truck). Moreover, the "common ontology" should provide homogeneous access to heterogeneous data (e.g. patient vital signs originating from different platforms).

At the abstract level, the above situation can be expressed in terms of a *network of ontologies* (cf. [10]), which consists of a set of ontologies, together with a family of pairwise alignments between them. Alignments, which are the essential part of the network of ontologies, can be obtained/developed using tools, which we have discussed so far. Note that the choice of a particular tool, is highly use-case dependent. For some ontologies, alignment might be best computed at the language-level, whereas for others a different approach might be preferable. Therefore, within the INTER-IoT project, we are inclined to use the matching methods in a parametric fashion. It is important to note that tools like the Alignment API (see Section 5.4), or the more recent, Distributed Ontology Language (DOL) [19,20], allow for treating alignments as *first-class* citizens. By utilizing them, we can store and perform computations on alignments.

Having obtained the necessary alignments, what we need in the next step, is a systematic way of using them to produce the final *merged ontology*, i.e., the *common semantics* for all involved platforms. An interesting approach, based on concepts from the *category theory* has been proposed in [24], further refined in [6] and implemented as a part of the Heterogeneous Toolset (HETS) [3]. The network of aligned ontologies has to be converted to a *diagram* of ontologies, for which, eventually, HETS computes the merged result. The computation of the final result corresponds to taking the categorical *limit* of the diagram. In case of an ontology format mismatch (e.g. RDF and database schema), tools such as D2RQ (e.g. Openlink Virtuoso) can be used to obtain a common format (discussion of such tools is outside the scope of this paper). Using HETS, one can even maintain the heterogeneous nature of the final ontology, although from the practical point of view it might not be the best approach.

The choice of the ontology merging tool(s), which we shall utilize within the INTER-IoT project is still open, though. HETS is definitely an interesting option, but we still need to investigate the landscape of available solutions.

## 7 Concluding remarks

The aim of this paper was to investigate what tools are actually available for someone who needs to align/merge ontologies. The results of investigating 97 possibilities, are somewhat discouraging. Majority of existing research results

are purely theoretical, i.e. they end with publication(s), but no tools become available. Large number of tools disappear/stop being maintained within 1-2 years after publication of the last paper. This concerns also tools that have participated in the OEAI competition.

Furthermore, almost no attention is paid to the realistic scenario where ontologies, to be aligned/merged are represented in different formalisms. Recall, that none of the IoT platforms that we are going to deal with, have semantics represented in RDF/OWL. Therefore, translation between JSON/XML demarcated semantics and OWL/RDF will be required. Overall, this indicates a serious chasm between the focus of semantic research (let us deal with the best case scenario of RDF/OWL defined ontologies) and where the needs of the real world are (use of semantic technologies requires, first, extracting and formally representing knowledge existing in real-world ICT systems and, second, building two-way translators). This may also seriously, negatively, affect acceptance and use of semantic technologies, and bringing about the vision of the Semantic Web.

Overall, we have identified seven tools that are alive and can be used for practical applications. They will be considered from the point of view of use case scenarios and potential for generalization outside of the current domain of interest (to support establishing interoperability between *any* IoT platforms). Finally, we aim at implementing interoperable ontologies in agent-based IoT middlewares, similar to these described in [14,15].

**Acknowledgments** Research presented in this paper has been partially supported by EU-H2020-ICT grant INTER-IoT 687283.

## References

1. 50 Ontology Mapping and Alignment Tools. http://www.mkbergman.com/1769/50-ontology-mapping-and-alignment-tools/
2. 50 Ontology Matching. http://ontologymatching.org/projects.html
3. Heterogenous Toolset. http://theo.cs.uni-magdeburg.de/Research/Hets.html
4. INTER-IoT Project. http://www.inter-iot-project.eu
5. Aumueller, D., Do, H.H., Massmann, S., Rahm, E.: Schema and ontology matching with COMA++. In: Proc. of the 2005 ACM SIGMOD Int. Conf. on Management of Data. pp. 906–908. ACM (2005)
6. Codescu, M., Mossakowski, T., Kutz, O.: A categorical approach to ontology alignment. In: Shvaiko, P., Euzenat, J., Mao, M., Jiménez-Ruiz, E., Li, J., Ngonga, A. (eds.) Proc. 9th International Workshop on Ontology Matching collocated with the 13th International Semantic Web Conference (ISWC 2014), Riva del Garda, Trentino, Italy. CEUR Workshop Proceedings, vol. 1317, pp. 1–12. CEUR-WS.org (2014), http://ceur-ws.org/Vol-1317/om2014_proceedings.pdf
7. David, J., Euzenat, J., Scharffe, F., Trojahn dos Santos, C.: The alignment API 4.0. Semantic Web 2(1), 3–10 (2011)
8. Dowling, W.F., Gallier, J.H.: Linear-time algorithms for testing the satisfiability of propositional Horn formulae. J. of Logic Programming 1(3), 267–284 (1984)

9. Duyhoa, N., Bellahsene, Z.: Overview of YAM++ — (not) Yet Another Matcher for ontology alignment task. Research report, LIRMM (sep 2014), http://hal-lirmm.ccsd.cnrs.fr/lirmm-01079124
10. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, 2 edn. (2013)
11. Faria, D., Martins, C., Nanavaty, A.: AgreementMakerLight results for OAEI 2014. In: ISWC Int. Workshop on Ontology Matching (OM), CEUR Workshop Proceedings (2014)
12. Faria, D., Martins, C., Nanavaty, A., Oliveira, D., Balasubramani, B.S., Taheri, A., Pesquita, C., Couto, F.M., Cruz, I.F.: AML results for OAEI 2015. In: ISWC Int. Workshop on Ontology Matching (OM), CEUR Workshop Proceedings (2015)
13. Faria, D., Pesquita, C., Santos, E., Palmonari, M., Cruz, I.F., Couto, F.M.: The AgreementMakerLight ontology matching system. In: On the Move to Meaningful Internet Systems: OTM 2013 Conferences. pp. 527–541. Springer (2013)
14. Fortino, G., Guerrieri, A., Lacopo, M., Lucia, M., Russo, W.: An agent-based middleware for cooperating smart objects. In: Highlights on Practical Applications of Agents and Multi-Agent Systems - International Workshops of PAAMS 2013, Salamanca, Spain, May 22-24, 2013. Proceedings. pp. 387–398 (2013), http://dx.doi.org/10.1007/978-3-642-38061-7_36
15. Fortino, G., Guerrieri, A., Russo, W., Savaglio, C.: Middlewares for smart objects and smart environments: Overview and comparison. In: Internet of Things Based on Smart Objects, Technology, Middleware and Applications, pp. 1–27 (2014), http://dx.doi.org/10.1007/978-3-319-00491-4_1
16. Ganzha, M., Paprzycki, M., Pawłowski, W., Szmeja, P., Wasielewska, K.: Semantic technologies for the IoT – an Inter-IoT perspective. In: 2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI). pp. 271–276. IEEE, Berlin, Germany (April 2016)
17. Giunchiglia, F., Autayeu, A., Pane, J.: S-Match: An open source framework for matching lightweight ontologies. Semantic Web 3(3), 307–317 (2012)
18. Jiménez-Ruiz, E., Grau, B.C.: LogMap: Logic-based and scalable ontology matching. In: International Semantic Web Conference (ISWC). LNCS, vol. 7031, pp. 273–288. Springer (2011)
19. Lange, C., Kutz, O., Mossakowski, T., Grüninger, M.: The distributed ontology language (DOL): Ontology integration and interoperability applied to mathematical formalization. In: Jeuring, J., Campbell, J.A., Carette, J., Dos Reis, G., Sojka, P., Wenzel, M., Sorge, V. (eds.) Intelligent Computer Mathematics, Proc. of CICM 2012. LNAI, vol. 7362, pp. 463–467. Springer (2012)
20. Mossakowski, T., Codescu, M., Neuhaus, F., Kutz, O.: The distributed ontology, modeling and specification language—DOL. In: Koslow, A., Buchsbaum, A. (eds.) The Road to Universal Logic: Festschrift for the 50th Birthday of Jean-Yves Béziau Volume II, pp. 489–520. Springer (2015)
21. Otero-Cerdeira, L., Rodríguez-Martínez, F.J., Gómez-Rodríguez, A.: Ontology matching: A literature review. Expert Systems with Applications 42(2), 949–971 (2015)
22. Ruiz, E.J., Grau, B.C., Zhou, Y., Horrocks, I.: Large-scale interactive ontology matching: Algorithms and implementation. In: Proc. of the 20th European Conference on Artificial Intelligence (ECAI). pp. 444–449. IOS Press (2012)
23. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Silk—A link discovery framework for the web of data. LDOW 538 (2009)
24. Zimmermann, A., Krötzsch, M., Euzenat, J., Hitzler, P.: Formalizing ontology alignment and its operations with category theory. In: Bennett, B., Fellbaum, C.

(eds.) Formal Ontology in Information Systems, Proc. Fourth International Conference, FOIS 2006, Baltimore, MD, USA. Frontiers in Artificial Intelligence and Applications, vol. 150, pp. 277–288. IOS Press (2006), http://www.booksonline.iospress.nl/Content/View.aspx?piid=2214