A Benchmark for Ontologies Merging Assessment

Mariem Mahfoudh[1], Germain Forestier[2], and Michel Hassenforder[2]

[1] CNRS, LORIA, UMR 7503
615 rue du Jardin Botanique, 54506 Vandœuvre-lès-Nancy, France
mariem.mahfoudh@loria.fr
[2] MIPS EA 2332, Université de Haute Alsace
12 rue des Frères Lumière 68093 Mulhouse, France
germain.forestier@uha.fr, michel.hassenforder@uha.fr

Abstract. In the last years, ontology modeling became popular and thousands of ontologies covering multiple fields of application are now available. However, as multiple ontologies might be available on the same or related domain, there is an urgent need for tools to compare, match, merge and assess ontologies. Ontology matching, which consists in aligning ontology, has been widely studied and benchmarks exist to evaluate the different matching methods. However, somewhat surprisingly, there are no significant benchmarks for merging ontologies, proving input ontologies and the resulting merged ontology. To fill this gap, we propose a benchmark for ontologies merging, which contains different ontologies types, for instance: taxonomies, lightweight ontologies, heavyweight ontologies and multilingual ontologies. We also show how the GROM tool (Graph Rewriting for Ontology Merging) can address the merging process and we evaluate it based on coverage, redundancy and coherence metrics. We performed experiments and show that the tool obtained good results in terms of redundancy and coherence.

Keywords: ontologies merging, benchmark, graph rewriting, GROM tool

1 Introduction

In the two last decades, ontologies have become widely used in several domains such as semantic web, medicine, e-commerce and natural language processing. With this multitude of ontologies that represent and cover sometimes the same domain, there is a growing need to merge these ontologies [1].

Merging ontologies have the goal of *"creating a new ontology from two or more existing ontologies with overlapping parts, which can be either virtual or physical"* [2]. It starts with the identification of the overlapping part (the similarities) between the ontologies entities and based on this result, it merges them and creates a new one [3]. Merging ontologies is a challenging issue that depends on several factors. Among them, the most important are: 1) the quality of initials ontologies (consistent or inconsistent), 2) the quality of ontology alignment

This is the author's version of an article published at KSEM 2010. The final authenticated version is available online at: http://dx.doi.org/10.1007/978-3-319-47650-6_44

(the identification of the similarities could recognize syntactic and/or semantic and/or structural correspondences between the ontologies ?), and 3) the merging strategy and the quality of its results. To evaluate the merging process and compare the tools proposed by the community, it is important to: 1) check the consistency of the initials ontologies (it could be done with a reasoner such as Pellet, Hermit, etc.) ; 2) evaluate the ontologies alignment ; 3) evaluate the merging result.

To evaluate ontologies alignment, researchers can use existing proposed benchmarks. This research field has now reached a significant maturity and there are specialized conferences that propose important benchmarks. As an example, the Ontology Matching[3] conference presents more than 20 datasets and publishes annually the results of their alignment. However, somewhat surprisingly, there are no significant benchmarks for merging ontologies [1,3]. To the best of our knowledge, the only work is the one of Raunich and Rahm [1]. It presents some taxonomies and the result of their merging using the ATOM (Automatic Target-driven Ontology Merging) tool [3]. This work has two main limitations. The first one is that it studies only taxonomies. Therefore, it can not test and evaluate the heavyweight ontologies (no property, no axioms contradiction, etc.). The second limit is that the benchmark is not published and therefore cannot be used.

We propose in this paper a benchmark for ontologies merging, which contains both lightweight and heavyweight ontologies. We show how our tool GROM (Graph Rewriting for Ontology Merging) [4] can address the merging process and we evaluate it based on coverage, redundancy and coherence metrics. All the ontologies, the result of their alignment and of their merging are available on the web for download.

The paper is structured as follows: Section 2 introduces the background of the work. It presents the merging process and the typed graphs grammars formalism. Section 3 defines our formalism and discusses how to use it to merge ontologies and resolve their inconsistencies. Section 4 presents a benchmark for ontologies merging and evaluates the GROM tool. Finally, a conclusion summarizes the presented work.

2 Background

2.1 Ontologies merging

Ontologies are living objects that represent knowledge with an explicit and formal way [5]. They conceptualize a given domain by their concepts (classes, properties, individuals, axioms, etc.) in order to offer mechanisms of reasoning and inference. Given two or more ontologies, ontologies merging aims at producing a new ontology with their overlapping parts [2]. This process can be symmetric or asymmetric. Symmetric solutions aim at completely integrating all input ontologies with the same priority. Asymmetric approaches, by contrast, take one of

[3] http://www.ontologymatching.org

the ontologies as the source and merge the other as a target. In this type of approach, the concepts of the source ontology are preserved and only the concepts of the target ontology, that not alter the consistency, are added [3].

Several approaches have been proposed in the literature, we briefly present in this section the approaches that are implemented with a proposed tool. The Table 1 summarizes these approaches according to: 1) the merging strategy (symmetric or asymmetric), 2) the ontology specification (OWL (Web Ontology Language), RDFS (Resource Description Framework Schema), Frame, etc.), 3) the tool, and 4) their specificities and the inconsistencies resolution (conflicts management). Note that because of the no-existence of the benchmark for merging ontologies, the researchers do not present details evaluation for their tools.

Approach	**Merge strategy**	**Specification**	**Tool**	**Specificity and conflicts management**
Stumme et al. [6]	Symmetric	Frame	FCA-Merge	- Semi-automatic approach - Approach based on the formal concept analysis (FCA) - No conflicts management
Noy et al. [7]	Symmetric & Asymmetric	Frame	Prompt	- Semi-automatic approach - Conflicts detection - User intervention for the conflicts resolution
Kotis et al. [8]	Symmetric	–	HCONE	- Semi-automatic approach - Approach based on the Latent semantic analysis (LSA) - No conflicts management
Li et al. [9]	Symmetric	OWL	MOMIS	- Automatic approach - Management of some conflicts (structural and semantic)
Raunich et al. [3]	Asymmetric	OWL Taxonomy	ATOM	- Automatic approach - Delete the redundancy - Management of some structural conflicts

Table 1: Summary of some ontologies merging approaches.

2.2 Quality measures for ontology merging

Measuring the quality of ontology merging finds its origins in the field of conceptual schemas. Indeed, to evaluate the quality of an integrated schema, Duchateau et al. [10] proposed two measures: minimality and completeness. The minimality checks that no redundant concept appears in the integrated schemas. The completeness represents the percentage of concepts presented in the data sources that are covered by the integrated schemas. It is calculated as follows:

$$comp(Si_{tool}, Si_{exp}) = \frac{|Si_{tool} \cap Si_{exp}|}{Si_{exp}} \quad (1)$$

where Si_{exp} is the integrated schema proposed by an expert and the Si_{tool} is the integrated schema generated by a tool.

Rahm et al. [1] proposed the same metrics to evaluate the quality of the merging ontologies result and they called them: *coverage* (for the completeness) and *redundancy* (for the minimality). The coverage is then related to the degree of information preservation. It measures the share of input concepts preserved in the result and it depends on the type of approach symmetric or asymmetric one. With symmetric approaches (full merge) the coverage is equal to 1: all the concepts are preserved. For the asymmetric approaches, the concepts of the target ontology are preserved but only the concepts non-redundant of the source ontology are preserved (*i.e.* all the redundant concepts are removed).

Besides the coverage and the redundancy metrics, it is also important to check the *coherence* of the ontologies merging result. This metric checks if the ontology result is coherent or contains some conflicts. To ensure theses three metrics, we have used the typed graph grammar formalism for our approach of merging ontologies.

2.3 Typed Graph Grammars

Typed Graph Grammars (TGG) are a mathematical formalism that permits to represent and manage graphs. They are used in several fields of computer science such as software systems modelling and formal language theory [11]. Recently, they started to be used in the ontology field, in particular for the modular ontologies formalization [12] and consistent ontologies evolution [13,14]. A typed graph grammar is defined by $TGG = (G, TG, P)$ where:

- G is a start graph also called host graph.
- TG is a type graph that represents the elements type of the graph G.
- P is a set of production rules also called graph rewriting rules which are defined by a pair of graphs patterns (LHS, RHS) where: 1) LHS (Left Hand Side) represents the preconditions of the rewriting rule and describes the structure that has to be found in G ; 2) RHS (Right Hand Side) represents the postconditions of the rule and must replaces LHS in G (see Figure 1).

A rewriting rule can be extended with a set of negative application conditions ($NACs$). A NAC is another graph pattern such as: "if there exist a morphism from NAC to the host graph G, then, the rule cannot be applied". In this way, a graph transformation defines how a graph G can be transformed to a new graph G'. More precisely, there must exist a morphism that replaces LHS by RHS to obtain G'. To apply this replacement, different graph transformations approaches are proposed [15]. In this work, we use the algebraic approach [16] based on the *pushout* concept [17]. Given three objects (in our case graphs) G_1, G_2 and G_3 and two morphisms $f : G_1 \to G_2$ and $g : G_1 \to G_3$, the pushout of G_2 and G_3 consists of: 1) an object G_4 and two morphisms $f' : G_2 \to G_4$ and $g' : G_3 \to G_4$ where $f' \circ f = g' \circ g$; 2) for any morphisms $f'' : G_2 \to X$ and $g'' : G_3 \to X$ such that $f \circ f'' = g \circ g''$, there is a unique morphism $k : G_4 \to X$ such that $f' \circ k = f''$ and $g' \circ k = g''$. Algebraic approaches are divided into two categories: the *Single PushOut, SPO* [18] and the *Double*

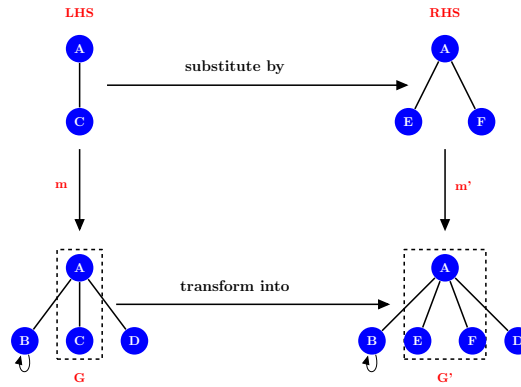


Fig. 1: The principle of graph transformation.

PushOut, DPO [19]. The *DPO* approach consists of two pushouts and requires an additional condition called the "dangling condition". This condition states that the transformation is applicable only if it does not lead to "dangling edges", i.e. an edge without a source or a target node. Indeed, in the *SPO* approach, one pushout is required and the dangling edges are removed which permits to write a wide variety of transformations not allowed by the *DPO* approach. Thus, in this work, we only consider the *SPO* approach. Applying a rewriting rule to an initial graph (G) with the *SPO* method consists in the following steps:

1. find a matching of LHS in G, i.e. find a morphism $m : LHS \rightarrow G$.
2. delete the sub-graph $m(LHS) - m(LHS \cap RHS)$ from G.
3. add the sub-graph $m(RHS) - m(LHS \cap RHS)$ to G to get G'.

3 Merging ontologies with typed graph grammars

3.1 Ontologies as typed graphs grammars

In order to represent the ontologies and the ontology changes with the typed graph grammars (see Figure 2), we use the *TGGOnto* (Typed Graph Grammars for Ontologies) model [13,14]:

$$TGGOnto = \{TG_O, G_O, R_O\}, \text{ where:}$$

TG_O is a type graph that represents the meta-model of an ontology, G_O is a host graph that represents an ontology and R_O is a set of rewriting rules that formalize the ontology changes. In our work, we consider the OWL ontologies. Therefore, the type graph (TG_O) represents the OWL meta-model. Thus, the vertices types of TG_O are:

$$V_T = \{Class(C), Property(P), ObjectProperty(OP),\text{-}$$
$$DataProperty(DP), Individual(I), DataType(D), Restriction(R)\}.$$

The edge types correspond to properties used to relate different entities. For example, $subClassOf$ is used to link nodes of the type $Class$.

$$E_T = \{subClassOf, equivalentTo, range, domain, ...\}.$$

An ontology change (CH) is formalized by rewriting rules and executed as graphs transformation using SPO algebraic method [14].

$$CH = r_i \in R_O = (NACs, LHS, RHS, CHDs), \text{ where:}$$

- $NACs$ are graph patterns that define the conditions should not be satisfied to apply the change ;
- LHS is a graph pattern that defines the preconditions that should be satisfied to apply an ontology change ;
- RHS is a graph pattern that defines the change to apply in the ontology ;
- $CHDs$ are derived changes added to the principal change (CH) for keeping the consistency of the modified ontology.

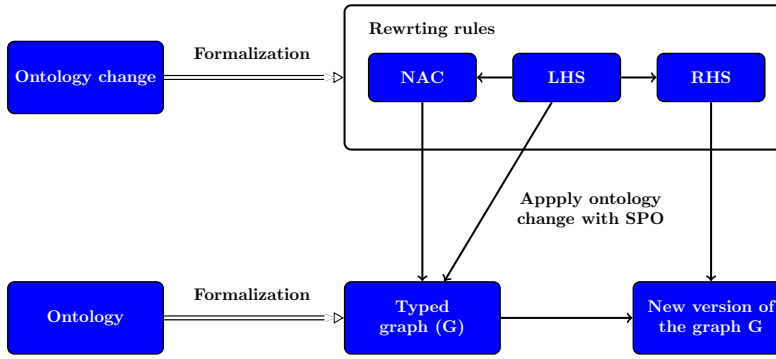


Fig. 2: The coupling between ontologies and typed graph grammars

3.2 GROM approach

To merge ontologies, we use the automatic and asymmetric approach GROM (Graph Rewriting for Ontologies Merging) which consists in three main steps that are briefly described below (for more details, we invite readers to refer to our previous work [4]).

1. Similarity search that identifies the correspondences between the ontologies entities based on syntactic, structural and semantic similarities. Given two ontologies (O_1 and O_2), we distinguish: 1) CN, the set of commons nodes between O_1 and O_2 ; 2) EN, the set of the syntactically equivalent nodes ; 3) SN, the set of the synonyms nodes and 4) $IsaN$, the set of the nodes that share a subsumption relation.

2. Ontologies merging that represents ontologies with $TGGOnto = \{TG_O, G_O, R_O\}$ model and merges them based on the result of the similarity search step. The merging process consists in applying a set of consecutive rewriting rules with the SPO algebraic method in order to create a consistent global ontology (see Algorithm 1).

Inputs: two ontologies O_1, O_2
a set of correspondences: CN, EN, SN, $IsaN$
Outputs: a global ontology GO

for $N \in EN$ **do**
  $O'_1 \leftarrow SPO\_RenameEntity\ (O_1, EN\{O_1\}, EN\{O_2\}\ )$;
end

$CN \leftarrow CN \cup EN\{O_1\}$;

$CO \leftarrow$ Create the common ontology;

$GO \leftarrow SPO\_MergeGraph\ (O'_1, CO, O_2)$;

`/* Adapt the global ontology */`
for $N \in SN$ **do**
  $GO \leftarrow SPO\_AddEquivalentEntity\ (GO, SN\{O_1\}, SN\{O_2\}\ )$;
end
for $N \in IsaN$ **do**
  $GO \leftarrow SPO\_AddSubClass\ (GO, IsaN\{O_1\}, IsaN\{O_2\}\ )$;
end

Algorithm 1: Merging ontologies algorithm.

3. Global ontology adaptation step enriches the global ontology with the synonym (SN) and the subsumption ($IsaN$) relations identified in the step 1 (similarity search). As example, the rewriting rule of $AddEquivalentClasses(C_1, C_2)$ ontology change is presented in the Figure 3.

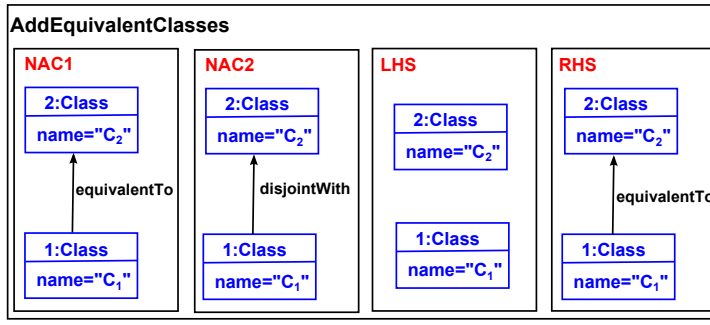


Fig. 3: Rewriting rule for the `AddEquivalentClasses` change.

The Rewriting rule (Figure 3) preserves the ontology consistency thanks to the $NACs$. $NAC1$ ensures the no redundancy. It prohibits the adding of existing knowledge. $NAC2$ ensure that no contradictory axiom is added to the ontology by the application of the *AddEquivalentClasses* ontology change.

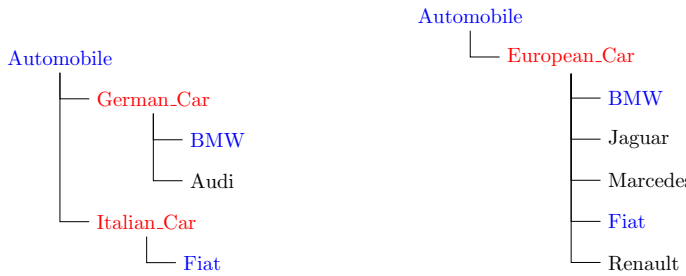
4 Experimental results

4.1 Test scenarios

We presented here two examples of ontologies (lightweight and heavyweight) to show and discuss the conflicts that can be found in the merging process.

Taxonomies case (Cars ontologies example [1]): The Figure 4 shows an example of two taxonomies that represent the vehicle domain. They share commons concepts ("Automobile", "BMW", "Fiat") and subsumption relations (isA ("German_Car", "European_Car"), isA ("Italian_Car", "European_Car") and isA ("Mercedes", "German_Car")). Merging these taxonomies can cause the following conflicts and situations:

- *Data redundancy.* Considering that the ontologies share common concepts, their merge result can contain redundant elements, for example ("Automobile", "Automobile"), ("Audi", "Audi"), etc.
- *Sharing subsumption relations.* The ontologies could share subsomption relations, for example: isA("German_Car", "European_Car") and isA("Italian_Car", "European_Car").
- *Existence of cycles.* Adding the subsumption relations could provide cycles. For example, if we merge the ontologies we can obtain the cycle: isA("German_Car", "European_Car"), isA("German_Car", "Automobile"), isA("European-Car", "Automobile").

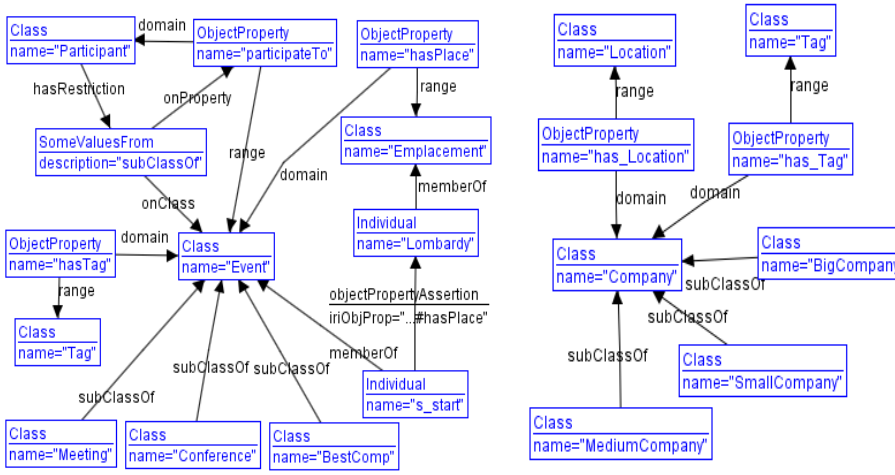


(a) Ontology O_1 (b) Ontology O_2

Fig. 4: European Cars ontologies [1].

OWL ontologies case (CCAlps ontologies example [14]): Figure 5 presents extracts from the EventCCAlps and CompanyCCAlps ontologies developed in the frame of the European project CCAlps.
Merge OWL ontologies can cause the following situations and conflicts:

- *Data redundancy.* The concepts that have names syntactical close could be considered as redundant knowledge, for example: "hasPlace" and "has_place".
- *Synonyms concepts.* The ontologies could share synonyms relations, for example: "Individual" and "Person". It is important in this case, to link theses concepts by synonyms relations on the global ontology.
- *Axioms contradiction.* Merging OWL ontologies could cause several axioms contradictions relating to the disjunction, the equivalence, the restrictions, the subclass axiom, etc.



(a) Extract from EOCCAlps ontology (O_1)

(b) Extract from COCCAlps ontology (O_2)

Fig. 5: CCAlps ontologies [14].

4.2 Benchmark and results

To evaluate the ontologies merging process and test the GROM tool, we are based on the benchmark below, which is available for download in the supplementary material attached with this paper[4]:

- Cars ontologies [1] are taxonomies which are composed of classes and subsumption relations (already described in Section 4.1, Figure 4). They are

[4] http://mariem-mahfoudh.info/ksem2016

mainly included in the benchmark, for testing the hierarchical properties and for checking if a tool of ontologies merging can remove the cycles.
- CCAlps ontologies [13] are heavyweight OWL ontologies (described in Section 4.1, Figure 5). These ontologies represent restrictions, properties, axioms, etc. and offer a good case study to check if an approach can manage or not the contradictory axioms and preserve or not the consistency of the global ontology.
- Lebensmittel (Google, web) [20] are multilingual ontologies that cover the food domain. `Google.Lebensmittel.owl` is represented in the English language and it is composed of 59 classes, 1306 individuals and 58 subClass axioms. `Web.Lebensmittel.owl` is represented in the German language and it is composed of 53 classes, 1566 individuals and 52 subClass axioms. The two ontologies have 15 synonyms classes that describe the same concepts but in two different languages.
- Freizeit (dmoz, Google) are multilingual ontologies that represent leisure. `Google.Freizeit.owl` is represented in the English language and it is composed of 71 classes. `Web.Freizeit.owl` is represented in the German language and it is composed of 76 classes. The two ontologies have 67 synonyms classes.
- Conference (ekaw, iasted, cmt, confof) ontologies [21] describe the conferences organization. They are datasets published by the OEAI (Ontology alignment evaluation initiative) to provide benchmark for the ontology alignment. We use the same ontologies to provide benchmark for the ontology merging.

The Table 2 presents the set of ontologies that form the proposed benchmark and their result merge by GROM approach. It specifies: 1) the number of concepts of each ontology (nbC), 2) their similarities (the set of commons nodes (CN), the set of synonyms nodes (SN), the set of nodes that share isa relations ($IsaN$) and the set of nodes that are syntactically close (EN)), 3) the number of concepts of the merge result, 4) the number of redundant concepts in the merge result, 5) the number of inconsistencies and 6) the value of coverage (Cov).

The Table 2 shows that all the output ontologies (the results of the process merging) don't contain any redundancy. Furthermore, GROM tool manages the inconsistencies and produces consistent ontologies.

Ontologies	**nbC**	**Similarities**	**Merge**	**Redundancy**	**Inconsistencies**	**Cov**	**Used in**
Cars	7 .. 6	6 #CN=3, #IsaN =3	13	0	0	1.00	[1]
CCAlps (Event, Company)	12 .. 8	4 #CN=1, #EN=1, #SN=1, #IsaN=1	18	0	0	0.9	[4]

Lebensmittel (Google, web)	53 .. 59	15 #SN=15	112	0	0	1.00	[20]
Freizeit (dmoz, Google)	71.. 67	67 #EN=67	71	0	0	0.51	[20]
Conference (ekaw, iasted)	107 .. 182	10 #CN=5, #EN=3, #SN=2	280	0	1[5]	0.97	[21]
Conference (cmt, confOf)	29 .. 38	9 #CN=9	58	0	0	0.86	[21]

Table 2: Merging result of some ontologies with GROM tool.

5 Conclusion

In this paper, we presented a benchmark for ontologies merging that covers different ontologies types: taxonomies, lightweight ontologies, heavyweight ontologies and multilingual ontologies. We have ensured that the benchmark presents different pathological cases for merging process (data redundancy, existence of cycles, axioms contradiction, etc.) in order to assess the performance of ontologies merging tools. We also presented our tool, GROM (Graph Rewriting for Ontology Merging) and we described how it can automatically merge ontologies. Thanks to the graph grammar formalism, in particular the NAC (Negative Application conditions), GROM preserves the ontology consistency and removes the redundancies. To evaluate the merging process, we have used the coverage, the redundancy and the coherence metrics. The evaluation has shown that GROM obtained good results in terms of redundancy and coherence. All the ontologies are provided for download. In the near future, we plan to expand this benchmark with the results of other researchers in order to compare the different results of the related work and test the performance of the ontologies merging tools.

References

1. Raunich, S., Rahm, E.: Towards a benchmark for ontology merging. In: Move to Meaningful Internet Systems: OTM 2012 Workshops, Springer (2012) 124–133
2. Klein, M.: Combining and relating ontologies: an analysis of problems and solutions. In: IJCAI-2001, Workshop on ontologies and information sharing. (2001) 53–62
3. Raunich, S., Rahm, E.: Target-driven merging of taxonomies with atom. Information Systems **42** (2014) 1–14

[5] The detected inconsistency is from iasted ontology.

4. Mahfoudh, M., Thiry, L., Forestier, G., Hassenforder, M.: Algebraic graph transformations for merging ontologies. In: Model and Data Engineering. Springer (2014) 154–168
5. Gandon, F.: Ontologies in Computer Science. In: Management and Design: Advanced Tools and Models. (2010)
6. Stumme, G., Maedche, A.: Fca-merge: Bottom-up merging of ontologies. In: Seventeenth International Joint Conference on Artificial Intelligence. (2001) 225–230
7. Noy, N.F., Musen, M.A.: Algorithm and tool for automated ontology merging and alignment. In: 17th National Conference on Artificial Intelligence, AAAI Press/The MIT Press (2000) 450–455
8. Kotis, K., Vouros, G.A.: The hcone approach to ontology merging. In: The Semantic Web: Research and Applications. Springer (2004) 137–151
9. Li, G., Luo, Z., Shao, J.: Multi-mapping based ontology merging system design. In: 2nd International Conference on Advanced Computer Control, IEEE (2010) 5–11
10. Duchateau, F., Bellahsene, Z.: Measuring the quality of an integrated schema. In: Conceptual Modeling–ER. Springer (2010) 261–273
11. Ehrig, H., Montanari, U., Rozenberg, G., Schneider, H.J.: Graph Transformations in Computer Science. Geschäftsstelle Schloss Dagstuhl (1996)
12. d'Aquin, M., Doran, P., Motta, E., Tamma, V.A.: Towards a parametric ontology modularization framework based on graph transformation. In: WoMO. (2007)
13. Mahfoudh, M., Forestier, G., Thiry, L., Hassenforder, M.: Consistent ontologies evolution using graph grammars. In: Knowledge Science, Engineering and Management (KSEM). Springer (2013) 64–75
14. Mahfoudh, M., Forestier, G., Thiry, L., Hassenforder, M.: Algebraic graph transformations for formalizing ontology changes and evolving ontologies. Knowledge-Based Systems **73** (2015) 212–226
15. Rozenberg, G.: Handbook of graph grammars and computing by graph transformation. Volume 1. World Scientific (1999)
16. Ehrig, H., Pfender, M., Schneider, H.J.: Graph-grammars: An algebraic approach. In: Switching and Automata Theory, pages=167–180, year=1973, organization=IEEE
17. Barr, M., Wells, C.: Category theory for computing science. Volume 10. Prentice Hall New York (1990)
18. Löwe, M.: Algebraic approach to single-pushout graph transformation. Theoretical Computer Science **109**(1) (1993) 181–224
19. Ehrig, H.: Introduction to the algebraic theory of graph grammars (a survey). In: Graph-Grammars and Their Application to Computer Science and Biology, Springer (1979) 1–69
20. Peukert, E., Massmann, S., Koenig, K.: Comparing similarity combination methods for schema matching. In: GI Jahrestagung (1), Citeseer (2010) 692–701
21. OAEI: Ontology alignment evaluation initiative. http://oaei.ontologymatching.org/2016/conference (2016)