A unified approach for aligning taxonomies and debugging taxonomies and their alignments - Extended version

Valentina Ivanova and Patrick Lambrix

Department of Computer and Information Science
and the Swedish e-Science Research Centre
Linköping University, 581 83 Linköping, Sweden

Abstract. With the increased use of ontologies in semantically-enabled applications, the issues of debugging and aligning ontologies have become increasingly important. The quality of the results of such applications is directly dependent on the quality of the ontologies and mappings between the ontologies they employ. A key step towards achieving high quality ontologies and mappings is discovering and resolving modeling defects, e.g., wrong or missing relations and mappings. In this paper we present a unified framework for aligning taxonomies, the most used kind of ontologies, and debugging taxonomies and their alignments, where ontology alignment is treated as a special kind of debugging. Our framework supports the detection and repairing of missing and wrong is-a structure in taxonomies, as well as the detection and repairing of missing (alignment) and wrong mappings between ontologies. Further, we implemented a system based on this framework and demonstrate its benefits through experiments with ontologies from the Ontology Alignment Evaluation Initiative.

This is an extended version combining the papers [11, 10] while adding additional parts. The paper [11] was a research paper and [10] was a demonstration paper, both at the 10th Extended Semantic Web Conference - ESWC 2013. The final publications of [11, 10] are available at link.springer.com. For reference purposes, please use [11].

1 Motivation

To obtain high-quality results in semantically-enabled applications such as the ontology-based text mining and search applications, high-quality ontologies and alignments are both necessary. However, neither developing nor aligning ontologies are easy tasks, and as the ontologies grow in size, it is difficult to ensure the correctness and completeness of the structure of the ontologies. For instance, some structural relations may be missing or some existing or derivable relations may be unintended. This is not an uncommon case. It is well known that people who are not expert in knowledge representation often misuse and confuse equivalence, is-a and part-of (e.g., [3]), which leads to problems in

the structure of the ontologies. Further, ontology alignment systems are used for generating alignments and, as shown in the Ontology Alignment Evaluation Initiative (OAEI, http://oaei.ontologymatching.org/), alignments usually contain mistakes and are incomplete. Such ontologies and alignments, although often useful, lead to problems when used in semantically-enabled applications. Wrong conclusions may be derived or valid conclusions may be missed.

A key step towards high-quality ontologies and alignments is debugging the ontologies and alignments. During the recent years several approaches have been proposed for debugging semantic defects in ontologies, such as unsatisfiable concepts or inconsistent ontologies (e.g., [26, 16, 17, 8]) and related to mappings (e.g., [24, 13, 25, 30]) or integrated ontologies [15]. Further, there has been some work on detecting modeling defects (e.g., [9, 4]) such as missing relations, and repairing modeling defects [21, 20, 18]. The increased interest in this field has also led to the creation of an international workshop on this topic [22]. In a separate sub-field of ontology engineering, ontology alignment, the correctness and completeness of the alignments has traditionally received much attention (e.g., [27]). Systems have been developed that generate alignments and in some cases validation of alignments is supported.

In this paper we propose a unified approach for ontology debugging and ontology alignment, where ontology alignment can be seen as a special kind of debugging. We propose an integrated framework that, although it can be used as an ontology debugging framework or an ontology alignment framework, presents additional benefits for both and leads to an overall improvement of the quality of the ontologies and the alignments. The ontology alignment provides new information that can be used for debugging and the debugging provides new information that can be used by the ontology alignment. Further, the framework allows for the interleaving of different debugging and alignment phases, thereby in an iterative way continuously generating new information and improving the quality of the information used by the framework.

In Section 3 we propose our unified approach for ontology alignment and debugging. To our knowledge this is the first approach that integrates ontology debugging and ontology alignment in a uniform way and that allows for a strong interleaving of these tasks. We present a framework as well as algorithms for the components. Section 4 provides an overview of the use and user interface of an implemented system. Further, we show the advantages of our approach in Section 5 through experiments with the ontologies and alignment of the OAEI 2011 Anatomy track. Related work is given in Section 6 and the paper concludes in Section 7. However, we start with some preliminaries.

2 Preliminaries

In this section we introduce notions that are needed for our approach. This paper focuses on taxonomies, which are the most widely used type of ontologies. (We will use 'taxonomy' and 'ontology' interchangeably in this paper.) Taxonomies consist of named concepts and subsumption (is-a) relations between the concepts. For this paper we use the following definition.

Definition 1. *A* **taxonomy** $\mathcal{O}$ *is represented by a tuple* $(\mathcal{C}, \mathcal{I})$ *where* $\mathcal{C}$ *is its set of named concepts and* $\mathcal{I} \subseteq \mathcal{C} \times \mathcal{C}$ *is a set of asserted is-a relations, representing the is-a structure of the ontology.*

The ontologies are connected into a network through alignments which are sets of mappings between concepts from two different ontologies. We currently consider *equivalence* mappings ($\equiv$) and is-a mappings (subsumed-by ($\rightarrow$) and subsumes ($\leftarrow$)).

Definition 2. *An* **alignment** *between ontologies* $\mathcal{O}_i$ *and* $\mathcal{O}_j$ *is represented by a set* $\mathcal{M}_{ij}$ *of pairs representing the* **mappings***, such that for concepts* $c_i \in \mathcal{O}_i$ *and* $c_j \in \mathcal{O}_j$*:* $c_i \rightarrow c_j$ *is represented by* (c_i, c_j)*;* $c_i \leftarrow c_j$ *is represented by* (c_j, c_i)*; and* $c_i \equiv c_j$ *is represented by both* (c_i, c_j) *and* (c_j, c_i)*.*[1]

The concepts that participate in mappings we call **mapped concepts**. Each mapped concept can participate in multiple mappings and alignments.

The output of ontology alignment systems are **mapping suggestions**. These should be validated by a domain expert and if accepted, they become part of an alignment.

Definition 3. *A* **taxonomy network** $\mathcal{N}$ *is a tuple* $(\mathbb{O}, \mathbb{M})$ *with* $\mathbb{O} = \{\mathcal{O}_k\}_{k=1}^{n}$ *the set of the ontologies in the network and* $\mathbb{M} = \{\mathcal{M}_{ij}\}_{i,j=1;i<j}^{n}$ *the set of representations for the alignments between these ontologies.*

Without loss of generality, in this paper we assume that the sets of named concepts for the different ontologies in the network are disjoint.

Figure 1 shows a small ontology network with two ontologies (concepts are represented by nodes and the is-a structures are represented by directed edges) and an alignment (represented by dashed edges).[2] The alignment consists of 7 equivalence mappings. One of these mappings represents the fact that the concept *bone* in the first ontology is equivalent to the concept *bone* in the second ontology. As these two concepts appear in a mapping, they are mapped concepts.

The domain knowledge of an ontology network is represented by its induced ontology.

Definition 4. *Let* $\mathcal{N} = (\mathbb{O}, \mathbb{M})$ *be an ontology network, with* $\mathbb{O} = \{\mathcal{O}_k\}_{k=1}^{n}$*,* $\mathbb{M} = \{\mathcal{M}_{ij}\}_{i,j=1;i<j}^{n}$*. Let* $\mathcal{O}_k = (\mathcal{C}_k, \mathcal{I}_k)$*. Then the* **induced ontology** *for network* $\mathcal{N}$ *is the ontology* $\mathcal{O}_N = (\mathcal{C}_N, \mathcal{I}_N)$ *with*
$\mathcal{C}_N = \cup_{k=1}^{n} \mathcal{C}_k$ *and* $\mathcal{I}_N = \cup_{k=1}^{n} \mathcal{I}_k \cup_{i,j=1;i<j}^{n} \mathcal{M}_{ij}$

In the algorithms we use the notion of knowledge base (KB). The notion that we define here is a restricted[3] variant of the notion as defined in description logics [1].

---

[1] Observe that for every $\mathcal{M}_{ij}$ there is a corresponding $\mathcal{M}_{ji}$ such that $\mathcal{M}_{ij} = \mathcal{M}_{ji}$. Therefore, in the remainder of this paper we will only consider the $\mathcal{M}_{ij}$ where i < j.

[2] The first ontology is a part of AMA, the second ontology is a part of NCI-A, and the alignment is a part of the alignment between AMA and NCI-A as defined in OAEI 2011.

[3] We use only concept names and no roles. The axioms in the TBox are of the form $A \dot{\sqsubseteq} B$ or $A \doteq C$, and the ABox is empty.

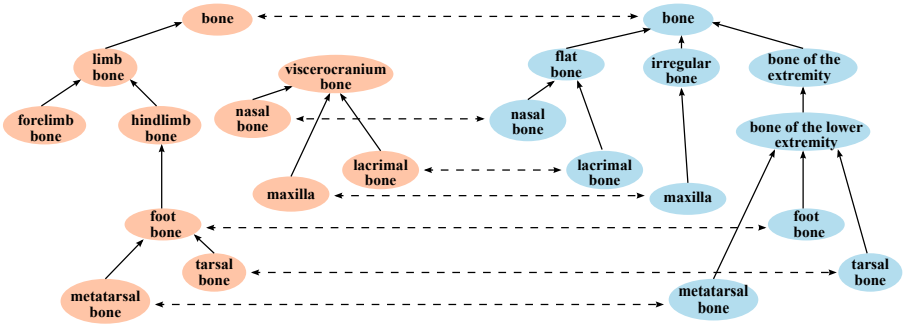

Fig. 1. (Part of an) Ontology network.

Definition 5. *Let $\mathcal{C}$ be a set of named concepts. A* **knowledge base** *is then a set of axioms of the form $A \rightarrow B$ with $A \in \mathcal{C}$ and $B \in \mathcal{C}$. A model of the knowledge base satisfies all axioms of the knowledge base.*

In the algoritms we initialize KBs with an ontology. This means that for ontology $\mathcal{O}$=$(\mathcal{C}, \mathcal{I})$ we create a KB such that (A,B) $\in \mathcal{I}$ iff $A \rightarrow B$ is an axiom in the KB.

For the KBs, we assume that they are able to do deductive logical inference. Further, we need the following reasoning services. For a given statement the KB should be able to answer whether the statement is entailed by the KB.[4] If a statement is entailed by the KB, it should be able to return the derivation paths (explanations) for that statement. For a given named concept, the KB should return the super-concepts and the sub-concepts.

The KBs can be implemented in several ways. For instance, any description logic system could be used. In our settting, where we deal with taxonomies, we have used an efficient graph-based implementation. We have represented the ontologies using graphs where the nodes are concepts and the directed edges represent the is-a relations. The entailment of statements of the form $a \rightarrow b$ can be checked by transitively following edges starting at a. If b is reached, then the statement is entailed, otherwise not. If $a \rightarrow b$ is entailed, then the derivation paths are all the different paths obtained by following directed edges that start at a and end at b. The super-concepts of a are all the concepts that can be reached by following directed edges starting at a. The sub-concepts of a are all the concepts for which there is a path of directed edges starting at the concept and ending in a.

3 Approach and Algorithms

Our framework consists of two major components - a debugging component and an alignment component. They can be used independently or in close interaction. The alignment component detects and repairs missing and wrong mappings between ontologies, while the debugging component additionally detects and repairs missing and

[4] In our setting, entailment by ontology can be reformulated as entailment by KB.

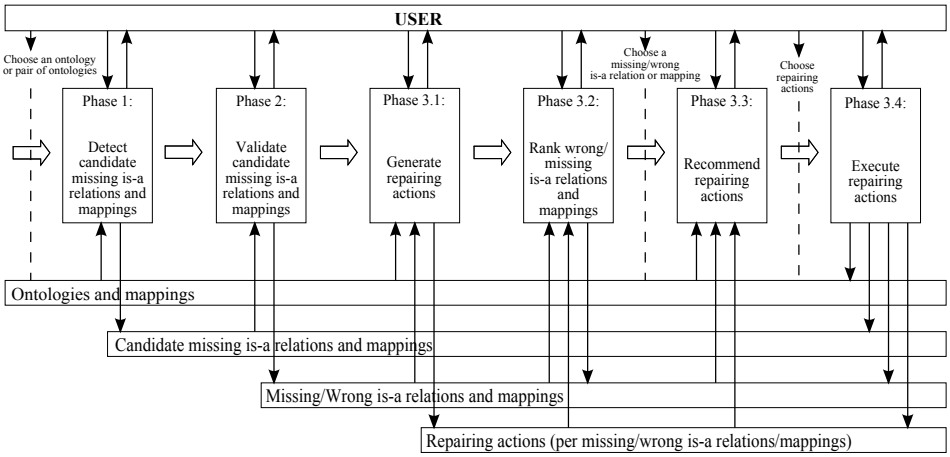

Fig. 2. Workflow.

wrong is-a structure in ontologies. Although we describe the two components separately, in our framework ontology alignment can be seen as a special kind of debugging.

The workflow in both components consists of three phases during which wrong and missing is-a relations/mappings are detected, validated and repaired in a semi-automatic manner by a domain expert (Figure 2). We note that at any time during the debugging/alignment workflow, the user can switch between different ontologies, start earlier phases, or switch between the repairing of wrong is-a relations, the repairing of missing is-a relations, the repairing of wrong mappings and the repairing of missing mappings. We also note that the repairing of defects often leads to the discovery of new defects. Thus several iterations are usually needed for completing the debugging/alignment process. The process ends when no more missing or wrong is-a relations and mappings are detected or need to be repaired.

In the following subsections we describe the components and their interactions, and present algorithms we have developed for the different components and phases.

3.1 Debugging component

The input for the debugging component is a taxonomy network, i.e., a set of taxonomies and their alignments. The output is the set of repaired taxonomies and alignments.

Phase 1: Detect candidate missing is-a relations and mappings.

In this component we focus on detecting wrong and missing is-a relations and mappings in the ontology network, based on knowledge that is inherent in the network. Therefore, given an ontology network, we use the domain knowledge represented by the ontology network to detect the deduced is-a relations and mappings in the network.

For each ontology in the network, the set of candidate missing is-a relations (CMIs) derivable from the ontology network consists of is-a relations between two concepts of the ontology, which can be inferred using logical derivation from the domain knowledge inherent in the network (i.e. from the induced ontology of the network), but not from the ontology alone. Similarly, for each pair of ontologies in the network, the set of candidate missing mappings (CMMs) derivable from the ontology network consists of mappings between concepts in the two ontologies, which can be inferred using logical derivation from the domain knowledge inherent in the network, but not from the two ontologies and their alignment alone.

Definition 6. *Let $\mathcal{N} = (\mathbb{O}, \mathbb{M})$ be an ontology network, with $\mathbb{O} = \{\mathcal{O}_k\}_{k=1}^{n}$, $\mathbb{M} = \{\mathcal{M}_{ij}\}_{i,j=1;i<j}^{n}$ and induced ontology $\mathcal{O}_N = (\mathcal{C}_N, \mathcal{I}_N)$. Let $\mathcal{O}_k = (\mathcal{C}_k, \mathcal{I}_k)$. Then, we define the following.*
(1) $\forall k \in 1..n$: *CMI*$_k$ = $\{(a,b) \in \mathcal{C}_k \times \mathcal{C}_k \mid \mathcal{O}_N \models a \rightarrow b \wedge \mathcal{O}_k \not\models a \rightarrow b\}$
is the set of candidate missing is-a relations for $\mathcal{O}_k$ derivable from the network.
(2) $\forall i, j \in 1..n, i < j$:
CMM$_{ij}$ = $\{(a,b) \in (\mathcal{C}_i \times \mathcal{C}_j) \cup (\mathcal{C}_j \times \mathcal{C}_i) \mid \mathcal{O}_N \models a \rightarrow b \wedge (\mathcal{C}_i \cup \mathcal{C}_j, \mathcal{I}_i \cup \mathcal{I}_j \cup \mathcal{M}_{ij}) \not\models a \rightarrow b\}$
is the set of candidate missing mappings for $(\mathcal{O}_i, \mathcal{O}_j, \mathcal{M}_{ij})$ derivable from the network.
(3) CMI = $\cup_{k=1}^{n}$ *CMI*$_k$ *is the set of* **candidate missing is-a relations derivable from the network**.
(4) CMM = $\cup_{i,j=1;i<j}^{n}$ *CMM*$_{ij}$ *is the set of* **candidate missing mappings derivable from the network**.

The CMIs and CMMs derivable from the network could be found using a brute-force method by checking each pair of concepts in the network. However, for large ontologies or ontology networks, this is infeasible. Further, some of these CMIs and CMMs are redundant in the sense that they can be repaired by the repairing of other CMIs and CMMs. Therefore, instead of checking all pairs of concepts in the network we only check all pairs of mapped concepts.[5] This choice is motivated by the fact that, in the restricted setting where we assume that all existing is-a relations in the ontologies and all existing mappings in the alignments are correct, it can be shown that all CMIs and CMMs[6] will be repaired when we repair the CMIs and CMMs between mapped concepts. This guarantees that for the part of the network for which the is-a structure and mappings are correct, we find all CMIs and CMMs when using the set of all pairs of mapped concepts. In addition, we may generate CMIs and CMMs that were derived using wrong information. These may later be validated as correct or wrong. As our debugging approach is iterative, after repairing, larger and larger parts of the network will contain only correct is-a structure and mappings. When finally all the network contains

[5] In the worst case scenario the number of mapped concept pairs is equal to the total number of concept pairs. In practice, the use of mapped concepts may significantly reduce the search space, e.g., when some ontologies are smaller than other ontologies in the network or when not all concepts participate in mappings. For instance, in the experiments in Section 5 the search space is reduced by almost 90%.

[6] In this setting all candidate missing is-a relations are also missing is-a relations, and all candidate missing mappings are also missing mappings.

1. Initialize KB_N with ontology network $\mathcal{N}$;
2. For k := 1 .. n: initialize KB_k with ontology $\mathcal{O}_k$;
3. For i := 1 .. n-1: for j := i+1 .. n:
    initialize KB_{ij} with ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$;
    for every mapping $(m, n) \in \mathcal{M}_{ij}$: add the axiom $m \rightarrow n$ to KB_{ij};

Fig. 3. Initialization for detection.

only correct is-a structure and mappings, it is guaranteed that all defects that can be found using the knowledge inherent in the network, are found using our approach.

Theorem 1. *Let $\mathcal{N} = (\mathbb{O}, \mathbb{M})$ be an ontology network with $\mathbb{O} = \{\mathcal{O}_k\}_{k=1}^{n}$ the set of the ontologies in the network and $\mathbb{M} = \{\mathcal{M}_{ij}\}_{i,j=1;i<j}^{n}$ the set of representations for the alignments between these ontologies. Further, assume that all is-a relations in the ontologies and all mappings in the alignments are correct. Then the following holds:*

(i) For each candidate missing is-a relation (a, b) in ontology $\mathcal{O}_i$, there exists a candidate missing is-a relation (x, y) in ontology $\mathcal{O}_i$ where x and y are mapped concepts in alignments between $\mathcal{O}_i$ and other ontologies in the network, such that the repairing of (x, y) also repairs (a, b).

(ii) For each candidate missing mapping (c, d) such that $c \in \mathcal{O}_i$ and $d \in \mathcal{O}_j$ with $i \neq j$, there exists a candidate missing mapping (x, y) such that $x \in \mathcal{O}_i$ and $y \in \mathcal{O}_j$, x is a mapped concept in an alignment between $\mathcal{O}_i$ and another ontology in the network and y is a mapped concept in an alignment between $\mathcal{O}_j$ and another ontology in the network, such that the repairing of (x, y) also repairs (c, d).

Proof. Assume (a, b) is a candidate missing is-a relation in $\mathcal{O}_i$. According to the definition of candidate missing is-a relation, the relation $a \rightarrow b$ is not derivable from $\mathcal{O}_i$ but derivable from the ontology network. So, there must exist at least one concept from another ontology in the network, for instance z, such that $\mathcal{O}_N \models a \rightarrow z \rightarrow b$. Because concepts a and z reside in different ontologies, the relation $a \rightarrow z$ must be supported by a mapping between a concept x in $\mathcal{O}_i$ and a concept x' in another ontology, e.g. $\mathcal{O}_r$, in the network, such that (x,x') $\in M_{ir}$ (if i $<$ r) or (x,x') $\in M_{ri}$ (if r $<$ i), and $\mathcal{O}_N \models a \rightarrow x \rightarrow x' \rightarrow z$. Likewise, for concepts z and b, the relation $z \rightarrow b$ must also be supported by a mapping between a concept y in $\mathcal{O}_i$ and a concept y' in another ontology, e.g. $\mathcal{O}_s$, in the network, such that (y',y) $\in M_{sj}$ (if s $<$ j) or (y',y) $\in M_{js}$ (if j $<$ s), such that $\mathcal{O}_N \models z \rightarrow y' \rightarrow y \rightarrow b$. We can then deduce that $x \rightarrow y$ is derivable from the ontology network because $\mathcal{O}_N \models a \rightarrow x \rightarrow x' \rightarrow z \rightarrow y' \rightarrow y \rightarrow b$. Since $a \rightarrow b$ is not inferrable from $\mathcal{O}_i$, the relation $x \rightarrow y$ can not be inferred from $\mathcal{O}_i$ either. This means that (x, y) is also a candidate missing is-a relation in $\mathcal{O}_i$, and the repairing of (x, y) also repairs (a, b). This proves statement (i). A similar proof can be given for statement (ii).

In our algorithm we initialize (Figure 3) a KB for the ontology network (KB_N), KBs for each ontology (KB_k) and for each pair of ontologies and their alignment (KB_{ij}). For each pair of mapped concepts within the same ontology, we check whether

an is-a relation between the pair can be derived from the KB of the network, but not from the KB of the ontology, and if so, it is a CMI. Similarly, for each pair of mapped concepts belonging to two different ontologies, we check whether an is-a relation between the pair can be derived from the KB of the network, but not from the KB of the two ontologies and their alignment, and if so, it is a CMM.

Phase 2: Validate candidate missing is-a relations and mappings.

Since the structure of the ontologies may contain wrong is-a relations and the alignments may contain wrong mappings, some of the CMIs and CMMs may be derived due to some wrong is-a relations and mappings. Therefore, they have to be validated by a domain expert. During Phase 2 the domain expert validates the CMIs and partitions them into two sets; $\mathcal{MI}_N$ containing the **missing is-a relations** and $\mathcal{WI}_N$ containing the **wrong is-a relations**. In this case we have that $\mathcal{MI}_N = \cup_{k=1}^{n} \mathcal{MI}_k$ with $\mathcal{MI}_k$ the set of missing is-a relations in $\mathcal{O}_k$, and $\mathcal{WI}_N = \cup_{k=1}^{n} \mathcal{WI}_k$ with $\mathcal{WI}_k$ the set of wrong is-a relations in $\mathcal{O}_k$. Similarly, the CMMs are validated and partitioned into two sets; $\mathcal{MM}_N$ containing the **missing mappings** and $\mathcal{WM}_N$ containing the **wrong mappings**. In this case we have that $\mathcal{MM}_N = \cup_{i,j;i<j=1}^{n} \mathcal{MM}_{ij}$ with $\mathcal{MM}_{ij}$ the set of missing mappings between $\mathcal{O}_i$ and $\mathcal{O}_j$, and $\mathcal{WM}_N = \cup_{i,j;i<j=1}^{n} \mathcal{WM}_{ij}$ with $\mathcal{WM}_{ij}$ the set of wrong mappings between $\mathcal{O}_i$ and $\mathcal{O}_j$. As an aid to the domain expert, we have developed recommendation algorithms based on the existence of is-a and part-of relations in the ontologies and external domain knowledge (WordNet [31] and UMLS [29]). In addition, the domain expert is provided with the derivation path (*justification*) for the CMI/CMM under validation. The derivation path is the sequence of is-a relations and/or mappings in the KB that leads to the logical derivation of the CMI/CMM. We note that each validation leads to a debugging opportunity. If a CMI or CMM is validated to be correct, then information is missing and is-a relations or mappings need to be added; otherwise, some existing information is wrong and is-a relations and/or mappings need to be removed.

Phase 3: Repair wrong and missing is-a relations and mappings.

Once missing and wrong is-a relations and mappings have been obtained[7], we need to repair them. For each ontology in the network, we want to repair the is-a structure in such a way that (i) the missing is-a relations can be derived from their repaired host ontologies and (ii) the wrong is-a relations can no longer be derived from the repaired ontology network. In addition, for each pair of ontologies, we want to repair the mappings in such a way that (iii) the missing mappings can be derived from the repaired host ontologies of their mapped concepts and the repaired alignment between the host ontologies of the mapped concepts and (iv) the wrong mappings can no longer be derived from the repaired ontology network. To satisfy requirement (i), we need to add a set of is-a relations to the host ontology. To satisfy requirement (iii), we need to add a set of is-a relations to the host ontologies of the mapped concepts and/or mappings to

[7] Using the technique for detection described above or the techniques used by the alignment component or any other technique.

the alignment between the host ontologies of the mapped concepts. To satisfy requirements (ii) and (iv), a set of asserted is-a relations and/or mappings should be removed from the ontology network. The notion of structural repair formalizes this. It contains is-a relations and mappings that should be added to or removed from the ontologies and alignments to satisfy these requirements. These is-a relations and mappings are called **repairing actions**.

Definition 7. *Let $\mathcal{N} = (\mathbb{O}, \mathbb{M})$ be an ontology network, with $\mathbb{O} = \{\mathcal{O}_k\}_{k=1}^{n}$, $\mathbb{M} = \{\mathcal{M}_{ij}\}_{i,j=1;i<j}^{n}$ and induced ontology $\mathcal{O}_N = (\mathcal{C}_N, \mathcal{I}_N)$. Let $\mathcal{O}_k = (\mathcal{C}_k, \mathcal{I}_k)$. Let $\mathcal{MI}_k$ and $\mathcal{WI}_k$ be the missing, respectively wrong, is-a relations for ontology $\mathcal{O}_k$ and let $\mathcal{MI}_N = \cup_{k=1}^{n}\mathcal{MI}_k$ and $\mathcal{WI}_N = \cup_{k=1}^{n}\mathcal{WI}_k$. Let $\mathcal{MM}_{ij}$ and $\mathcal{WM}_{ij}$ be the missing, respectively wrong, mappings between ontologies $\mathcal{O}_i$ and $\mathcal{O}_j$ and let $\mathcal{MM}_N = \cup_{i,j=1;i<j}^{n}\mathcal{MM}_{ij}$ and $\mathcal{WM}_N = \cup_{i,j=1;i<j}^{n}\mathcal{WM}_{ij}$. A* **structural repair for $\mathcal{N}$ with respect to** $(\mathcal{MI}_N, \mathcal{WI}_N, \mathcal{MM}_N, \mathcal{WM}_N)$*, denoted by $(\mathcal{R}^+, \mathcal{R}^-)$, is a pair of sets of is-a relations and mappings, such that*
(1) $\mathcal{R}^- \cap \mathcal{R}^+ = \emptyset$
(2) $\mathcal{R}^- = \mathcal{R}_M^- \cup \mathcal{R}_I^-$; $\mathcal{R}_M^- \subseteq \cup_{i,j=1,i<j}^{n}\mathcal{M}_{ij}$; $\mathcal{R}_I^- \subseteq \cup_{k=1}^{n}\mathcal{I}_k$
(3) $\mathcal{R}^+ = \mathcal{R}_M^+ \cup \mathcal{R}_I^+$; $\mathcal{R}_M^+ \subseteq \cup_{i,j=1,i<j}^{n}((\mathcal{C}_i \times \mathcal{C}_j) \setminus \mathcal{M}_{ij})$; $\mathcal{R}_I^+ \subseteq \cup_{k=1}^{n}((\mathcal{C}_k \times \mathcal{C}_k) \setminus \mathcal{I}_k)$
(4) $\forall k \in 1..n : \forall (a,b) \in \mathcal{MI}_k$: $(\mathcal{C}_k, (\mathcal{I}_k \cup (\mathcal{R}_I^+ \cap (\mathcal{C}_k \times \mathcal{C}_k))) \setminus \mathcal{R}_I^-) \models a \rightarrow b$
(5) $\forall i,j \in 1..n, i < j : \forall (a,b) \in \mathcal{MM}_{ij}$: $((\mathcal{C}_i \cup \mathcal{C}_j), (\mathcal{I}_i \cup ((\mathcal{C}_i \times \mathcal{C}_i) \cap \mathcal{R}_I^+) \cup \mathcal{I}_j \cup ((\mathcal{C}_j \times \mathcal{C}_j) \cap \mathcal{R}_I^+) \cup \mathcal{M}_{ij} \cup ((\mathcal{C}_i \times \mathcal{C}_j) \cap \mathcal{R}_M^+)) \setminus \mathcal{R}^-) \models a \rightarrow b$
(6) $\forall (a,b) \in \mathcal{WI}_N \cup \mathcal{WM}_N \cup \mathcal{R}^-$: $(\mathcal{C}_N, (\mathcal{I}_N \cup \mathcal{R}^+) \setminus \mathcal{R}^-) \not\models a \rightarrow b$

The definition states that (1) is-a relations and mappings cannot be added and removed at the same time, (2) the removed mappings come from the original alignments and the removed is-a relations come from the original asserted is-a relations in the ontologies, (3) the added mappings were not in the original alignments and the added is-a relations were not original is-a relations in the ontologies, (4) every missing is-a relation is derivable from its repaired host ontology, (5) every missing mapping is derivable from the repaired host ontologies of the mapped concepts and their repaired alignment, and (6) no wrong mapping, wrong is-a relation or removed mapping or is-a relation is derivable from the repaired network.

In our algorithm, at the start of the repairing phase (Figure 4) we add all missing is-a relations and mappings to the relevant KBs. As these are validated to be correct, this is extra knowledge that should be used in the repairing process. Adding the missing is-a relations and mappings essentially means that we have repaired these using the least informative repairing actions (related to the $\ll_I$ preference in [21]). Then during the repairing process we try to improve this and find more informative repairing actions. We say that a repairing action is more informative than another repairing action if adding the former to the ontology also allows to derive the latter. In general, more informative repairing actions that are correct according to the domain are preferred.

Definition 8. *Let (x_1, y_1) and (x_2, y_2) be two different is-a relations in the same ontology $\mathcal{O}$ (i.e., $x_1 \not\equiv x_2$ or $y_1 \not\equiv y_2$), then we say that (x_1, y_1) is* **more informative than** *(x_2, y_2) iff $\mathcal{O} \models x_2 \rightarrow x_1 \wedge y_1 \rightarrow y_2$.*

1. For k:= 1 .. n:
    for every missing is-a relation $(a, b) \in \mathcal{MI}_k$:
        add the axiom $a \rightarrow b$ to KB_N;
        add the axiom $a \rightarrow b$ to KB_k;
        for i := 1 .. k-1: add the axiom $a \rightarrow b$ to KB_{ik};
        for i := k+1 .. n: add the axiom $a \rightarrow b$ to KB_{ki};
2. For i := 1 .. n-1: for j := i+1 .. n:
    for every missing mapping $(m, n) \in \mathcal{MM}_{ij}$:
        add the axiom $m \rightarrow n$ to KB_N;
        add the axiom $m \rightarrow n$ to KB_{ij};
3. $\mathcal{MI} := \mathcal{MI}_N$; $\mathcal{WI} := \mathcal{WI}_N$; $\mathcal{MM} := \mathcal{MM}_N$; $\mathcal{WM} := \mathcal{WM}_N$;
4. $\mathcal{R}_I^+ := \emptyset$; $\mathcal{R}_I^- := \emptyset$; $\mathcal{R}_M^+ := \emptyset$; $\mathcal{R}_M^- := \emptyset$;
5. $\mathcal{CMI} := \emptyset$; $\mathcal{CMM} := \emptyset$;

Fig. 4. Initialization for repairing.

It follows that if (x_1, y_1) is more informative than (x_2, y_2) and $\mathcal{O} \models x_1 \rightarrow y_1$ then $\mathcal{O} \models x_2 \rightarrow y_2$. Therefore, adding or removing more informative repairing actions, adds or removes more knowledge than less informative repairing actions.

As an example, consider the missing is-a relation *(nasal bone, bone)* in the first ontology in Figure 1. Knowing that *nasal bone* $\rightarrow$ *viscerocranium bone*, according to the definition of more informative, we know that *(viscerocranium bone, bone)* is more informative than *(nasal bone, bone)*. As *viscerocranium bone* actually is a sub-concept of *bone* according to the domain, a domain expert would prefer to use the more informative repairing action for the given missing is-a relation.[8]

Further, we initialize global variables for the current sets of missing ($\mathcal{MI}$) and wrong ($\mathcal{WI}$) is-a relations, and the current sets of missing ($\mathcal{MM}$) and wrong ($\mathcal{WM}$) mappings based on the validation results. Further, the sets of repairing actions ($\mathcal{R}_I^+$ and $\mathcal{R}_I^-$ for is-a relations, and $\mathcal{R}_M^+$ and $\mathcal{R}_M^-$ for mappings), and the current sets of CMIs ($\mathcal{CMI}$) and CMMs ($\mathcal{CMM}$) are initialized to $\emptyset$.

A naive way of repairing would be to compute all possible structural repairs for the network with respect to the validated missing is-a relations and mappings for all the ontologies in the network. This is in practice infeasible as it involves all the ontologies and alignments and all the missing and wrong is-a relations and mappings in the network. It is also hard for domain experts to choose between structural repairs containing large sets of repairing actions for all the ontologies and alignments. Therefore, in our approach, we repair ontologies and alignments one at a time. For the selected ontology (for repairing is-a relations) or for the selected alignment and its pair of ontologies (for repairing mappings), a user can choose to repair the missing or the wrong is-a relations/mappings **(Phase 3.1-3.4)**. Although the algorithms for repairing are different for missing and wrong is-a relations/mappings, the repairing goes through the phases of generation of repairing actions (**Phase 3.1**), the ranking of is-a relations/mappings

[8] We also note that using *(viscerocranium bone, bone)* as repairing action would also immediately repair the missing is-a relations *(maxilla, bone)* and *(lacrimal bone, bone)*.

1. Compute $AllJust(w, r, \mathcal{O}_e)$
   where $\mathcal{O}_e = (\mathcal{C}_e, \mathcal{I}_e)$ such that $\mathcal{C}_e = \cup_{k=1}^{n} \mathcal{C}_k$ and
   $\mathcal{I}_e = ((\cup_{k=1}^{n} \mathcal{I}_k) \cup (\cup_{i,j=1;i<j}^{n} \mathcal{M}_{ij}) \cup \mathcal{MI}_N \cup \mathcal{MM}_N \cup \mathcal{R}_I^+ \cup \mathcal{R}_M^+) \setminus (\mathcal{R}_I^- \cup \mathcal{R}_M^-)$;
2. For every $\mathcal{I}' \in AllJust(w, r, \mathcal{O}_e)$:
   choose one element from $\mathcal{I}' \setminus (\mathcal{MI}_N \cup \mathcal{MM}_N \cup \mathcal{R}_I^+ \cup \mathcal{R}_M^+)$ to remove;

Fig. 5. Algorithm for generating repairing actions for wrong is-a relations and mappings.

according to the number of repairing actions (**Phase 3.2**), the recommendation of repairing actions (**Phase 3.3**) and finally, the execution of repairing actions (**Phase 3.4**).

Wrong is-a relations and mappings. Figure 5 shows the algorithm for generating repairing actions for a wrong is-a relation or mapping (Phase 3.1). This algorithm is run for all wrong is-a relations (elements in $\mathcal{WI}$) and mappings (elements in $\mathcal{WM}$). It computes all justifications (AllJust) for the wrong is-a relation or mapping (w,r) in the current ontology network ($\mathcal{O}_e$). The current network is the original network where the repairs up to now have been taken into account (i.e., all missing is-a relations are repaired by adding them while some are repaired using more informative repairing actions in $\mathcal{R}_I^+$, missing mappings have been repaired by adding them or by repairing actions in $\mathcal{R}^+M$ or $\mathcal{R}_I^+$, and some is-a relations and mappings are already repaired by removing is-a relations and mappings in $\mathcal{R}_I^-$ and $\mathcal{R}_M^-$, respectively). A justification for a wrong is-a relation or mapping can be seen as an explanation for why this is-a relation or mapping is derivable from the network.

Definition 9. *(similar definition as in [15]) Given an ontology $\mathcal{O} = (\mathcal{C}, \mathcal{I})$, and $(a, b) \in \mathcal{C} \times \mathcal{C}$ an is-a relation derivable from $\mathcal{O}$, then, $\mathcal{I}' \subseteq \mathcal{I}$ is a* **justification** *for (a, b) in $\mathcal{O}$, denoted by $\mathsf{Just}(\mathcal{I}', a, b, \mathcal{O})$ iff (i) $(\mathcal{C}, \mathcal{I}') \models a \rightarrow b$; and (ii) there is no $\mathcal{I}'' \subsetneq \mathcal{I}'$ such that $(\mathcal{C}, \mathcal{I}'') \models a \rightarrow b$. We use $\mathsf{All\_Just}(a, b, \mathcal{O})$ to denote the set of all justifications for (a, b) in $\mathcal{O}$.*

The algorithm to compute justifications initializes a KB taking into account the repairing actions up to now. To compute the justifications for $a \rightarrow b$ in our graph-based implementation, all the different paths obtained by following directed edges that start at a and end at b are collected. Among these the minimal ones (w.r.t $\subseteq$) are retained.

The wrong is-a relation or mapping can then be repaired by removing at least one element in every justification. However, missing is-a relations, missing mappings, and added repairing actions (is-a relations in ontologies and mappings) cannot be removed. Using this algorithm structural repairs are generated that include only contributing repairing actions (preference $\ll_A$ in [21]).

In general, there will be many is-a relations/mappings that need to be repaired and some of them may be easier to start with such as the ones with fewer repairing actions. We therefore rank them with respect to the number of possible repairing actions (Phase 3.2). Further, we implemented a recommendation algorithm (Phase 3.3) that assigns a priority to each possible repairing action based on how often it occurs in the justifications and its importance in already repaired is-a relations and mappings.

Repair missing is-a relation (a,b) with a $\in \mathcal{O}_k$ and b $\in \mathcal{O}_k$:
Choose an element from GenerateRepairingActions(a, b, KB_k);

Repair missing mapping (a,b) with a $\in \mathcal{O}_i$ and b $\in \mathcal{O}_j$:
Choose an element from GenerateRepairingActions(a, b, KB_{ij});

GenerateRepairingActions(a, b, KB):
1. $Source(a, b)$:= super-concepts(a) $-$ super-concepts(b) in KB;
2. $Target(a, b)$:= sub-concepts(b) $-$ sub-concepts(a) in KB;
3. $Repair(a, b)$:= $Source(a, b) \times Target(a, b)$;
4. For each $(s, t) \in Source(a, b) \times Target(a, b)$:
   if $(s, t) \in \mathcal{WI} \cup \mathcal{WM} \cup \mathcal{R}_I^- \cup \mathcal{R}_M^-$ then remove (s, t) from $Repair(a, b)$;
   else if $\exists (u, v) \in \mathcal{WI} \cup \mathcal{WM} \cup \mathcal{R}_I^- \cup \mathcal{R}_M^-$: (s, t) is more informative than (u, v) in KB
      and $u \rightarrow s$ and $t \rightarrow v$ are derivable from validated to be correct only is-a relations and/or mappings
   then remove (s, t) from $Repair(a, b)$;
5. return $Repair(a, b)$;

Fig. 6. Algorithm for generating repairing actions for missing is-a relations and mappings.

Once the user decides on repairing actions, the chosen repairing actions are then removed from the relevant ontologies and alignments (Phase 3.4) and a number of updates need to be done. First, the wrong is-a relation (or mapping) is removed from $\mathcal{WI}$ (or $\mathcal{WM}$). The chosen repairing actions that are is-a relations in an ontology are added to $\mathcal{R}_I^-$ and repairing actions that are mappings are added to $\mathcal{R}_M^-$. Some other wrong is-a relations or mappings may also have been repaired by repairing the current wrong is-a relation or mapping (update $\mathcal{WI}$ and $\mathcal{WM}$). Also, some repaired missing is-a relations and mappings may also become missing again (update $\mathcal{MI}$ and $\mathcal{MM}$). Further, new CMIs and CMMs may appear (update $\mathcal{CMI}$ and $\mathcal{CMM}$ - and after validation update $\mathcal{CMI}$, $\mathcal{MI}$, $\mathcal{WI}$, $\mathcal{CMM}$, $\mathcal{MM}$ and $\mathcal{WM}$). In other cases the possible repairing actions for wrong and missing is-a relations and mappings may change (update justifications and sets of possible repairing actions for missing is-a relations and mappings). We also need to update the KBs.

Missing is-a relations and mappings. It was shown in [18] that repairing missing is-a relations (and mappings) can be seen as a generalized TBox abduction problem. Figure 6 shows our solution for the computation of repairing actions for a missing is-a relation or mapping (Phase 3.1). The algorithm, an extension of the algorithm in [21], takes into consideration that all missing is-a relations and missing mappings will be repaired (least informative repairing action), but it does not take into account the consequences of the actual (possibly more informative) repairing actions that will be performed for other missing is-a relations and other missing mappings. The main component of the algorithm (GenerateRepairingActions) takes a missing is-a relation or mapping as input together with a KB. For a missing is-a relation this is the KB corresponding to the host ontology of the missing is-a relation; for a missing mapping this is the KB corresponding to the host ontologies of the mapped concepts in the missing mapping and their

alignment. In this component for a missing is-a relation or mapping we compute the more general concepts of the first concept (Source) and the more specific concepts of the second concept (Target) in the KB. To not introduce non-validated equivalence relations where in the original ontologies and alignments there are only is-a relations, we remove the super-concepts of the second concept from Source, and the sub-concepts of the first concept from Target. Adding an element from Source $\times$ Target to the KB makes the missing is-a relation or mapping derivable. However, some elements in Source $\times$ Target may conflict with already known wrong is-a relations or mappings. Therefore, in Repair, we take the wrong is-a relations and mappings and the former repairing actions for wrong is-a relations and mappings into account. The missing is-a relation or mapping can then be repaired using an element in Repair. We note that for missing is-a relations, the elements in Repair are is-a relations in the host ontology for the missing is-a relation. For missing mappings, the elements in Repair can be mappings as well as is-a relations in each of the host ontologies of the mapped concepts of the missing mapping. Using this algorithm structural repairs are generated that include only contributing repairing actions, and repairing actions of the form (a, t) or (s, b) for missing is-a relation or mapping (a, b) do not introduce non-validated equivalence relations (preferences $\ll_A$ and $\ll_{SH}$ in [21]).

In Phase 3.2 we rank the is-a relations/mappings that need to be repaired with respect to the number of possible repairing actions. In Phase 3.3 a recommendation algorithm (as defined in [21]) computes for missing is-a relation (a, b) the most informative repairing actions from $Source(a, b) \times Target(a, b)$ that are supported by domain knowledge (WordNet and UMLS).

When the selected repairing action is in $Repair(a, b)$, the repairing action is added to the relevant ontologies and alignments, and a number of updates need to be done. First, the missing is-a relation (or mapping) is removed from $\mathcal{MI}$ (or $\mathcal{MM}$) and the chosen repairing action is added to $\mathcal{R}_I^+$ or $\mathcal{R}_M^+$ depending on whether it is an is-a relation within an ontology or a mapping. Further, new CMIs and CMMs may appear. Some other missing is-a relations or mappings may also have been repaired by repairing the current missing is-a relation or mapping. Some repaired wrong is-a relations and mappings may also become derivable again. In other cases the possible repairing actions for wrong and missing is-a relations and mappings may change. We also need to update the KBs.

3.2 Alignment component

The input for the alignment component consists of two taxonomies. The output is an alignment.

Phase 1: Detect candidate missing mappings.

In ontology alignment mapping suggestions are generated which essentially are CMMs. While the generation of CMMs in the debugging component is a specific kind of ontology alignment using the knowledge inherent in the network, in the alignment component we use other types of alignment algorithms. Matchers are used to compute similarity values between concepts in different ontologies. The results of the matchers can

be combined and filtered in different ways to obtain mapping suggestions. In our approach we have currently used the linguistic, WordNet-based and UMLS-based algorithms from the SAMBO system [23]. The matcher *n-gram* computes a similarity based on 3-grams. The matcher *TermBasic* uses a combination of n-gram, edit distance and an algorithm that compares the lists of words of which the terms are composed. The matcher *TermWN* extends TermBasic by using WordNet for looking up is-a relations. The matcher *UMLSM* uses the domain knowledge in UMLS to obtain similarity values. The results of the matchers can be combined using a weighted-sum approach in which each matcher is given a weight and the final similarity value between a pair of concepts is the weighted sum of the similarity values divided by the sum of the weights of the used matchers. Further, we use a threshold for filtering. A pair of concepts is a mapping suggestion if the similarity value is equal to or higher than a given threshold value.

We note that in the alignment component the search space is not restricted to the *mapped concepts* only - similarity values are calculated for all pairs of concepts. KBs are initialized, in the same way as in the debugging component, for the taxonomy network and the pairs of taxonomies and their alignments. We also note that no initial alignment is needed for this component. Therefore, if alignments do not exist in the network (at all or between specific ontologies) this component may be used before starting debugging.

Phase 2: Validate candidate missing mappings.

The CMMs (mapping suggestions) are presented to a domain expert for validation, which is performed in the same way as in the debugging component. The domain expert can use the recommendation algorithms during the validation as well. As before, the CMMs are partitioned into two sets - wrong mappings and missing mappings. The wrong mappings are not repaired since they are not in the alignments. However, we store this information in order to avoid recomputations and for conflict checking/prevention. The concepts in the missing mappings are added to the set of *mapped concepts* (if they are not already there), and they will be used the next time CMMs/CMIs are derived in the debugging component.

Phase 3: Repairing missing mappings.

As mentioned, we only need to repair the missing mappings. Initially, the missing mappings are added to the KBs in the same way as in the debugging component and then we try to repair them using more informative repairing actions. For repairing a missing mapping the same algorithms as in the debugging component are used to generate the Source and Target sets and the repairing process continues with the same actions described for the debugging workflow. In Phase 3.4 the repairing actions are executed analogically to those in the debugging component and their consequences are computed. Further, the concepts in the repairing actions are added to the set of *mapped concepts* (if not there yet).

3.3 Interaction between the components

The alignment component generates CMMs that are validated in the same way as in the debugging component. The CMMs validated to be correct often are missing mappings that are not found by the debugging component. Further, they may lead to new mapped concepts that are used in the debugging component. The CMMs validated to be wrong are used to avoid unnecessary recomputations and validations.

The debugging component repairs the is-a structure and the mappings. This can be used by the alignment component. For instance, the performance of structure-based matchers (e.g., [23]) and partial-alignment-based preprocessing and filtering methods [19] heavily depends on the correctness and completeness of the is-a structure.

We also note that the different phases in the components can be interleaved. This allows for an iterative and modular approach, where, for instance, some parts of the ontologies can be fully debugged and aligned before proceeding to other parts.

4 Use of Implemented System

Detecting and validating candidate missing is-a relations and mappings.

In RepOSE, the user loads the ontologies and alignments (when available). Then the user can use the tab 'Step1: Generate and Validate Candidate Missing is-a Relations' (Figure 7) and choose an ontology for which the CMIs are computed. The user can validate all or some of the CMIs as well as switch to another ontology or another tab. CMIs are vizualized in groups in order to show them in their context, while avoiding cluterring of the display. Initially, CMIs are shown using arrows labeled by '?' (as in Figure 7 for *(acetabulum, joint)*) which the user can toggle to 'W' for wrong relations and 'M' for missing relations. For each CMI the justification in the ontology network is shown as an extra aid for the user. For instance, in Figure 7 *(palatine bone, bone)* is selected and its justifications shown in the justifications panel. Concepts in different ontologies are presented with different background color. The domain expert can also ask for recommendations. When a user decides to finalize the validation of a group of CMIs, RepOSE checks for contradictions in the current validation as well as with previous decisions and if contradictions are found, the current validation will not be allowed and a message window is shown to the user.

A similar tab 'Step 2: Generate and Validate Candidate Missing Mappings' can be used to choose a pair of ontologies and their alignment for which the CMMs are generated and, to validate them.

Detecting and validating candidate missing mappings - alignment component.

Tab 'Step 2: Generate and Validate Candidate Missing Mappings' is also used for the computation and validation of CMMs from the alignment component. Clicking on the `Configure and Run Alignment Algorithms` button opens a configuration window (Figure 8) where the user can select the matchers, their weights and the threshold for the computation of the mapping suggestions. Clicking on the `Run` button starts the alignment process. The similarity values for all pairs of concepts belonging to the

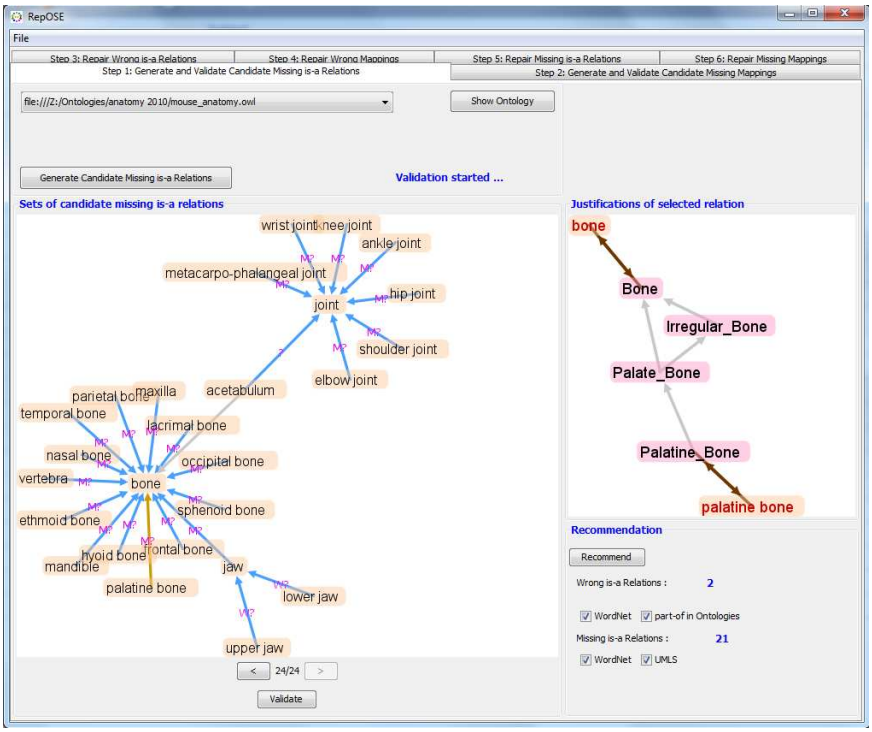

Fig. 7. Generating and validating CMIs.

selected ontologies are computed, combined and filtered, and the resulting mapping suggestions are shown to the user for validation. The validation process continues in a similar way as for the CMIs. During the validation a label on the edge shows the origin of the CMMs - derived from the network, computed by the alignment component or both. The CMMs computed only by the alignment algorithms do not have justifications since they were not logicaly derived. The rest of the process is as described above.

Repairing wrong is-a relations and mappings.

Figure 9 shows the RepOSE tab 'Step 3: Repair Wrong is-a Relations' for repairing wrong is-a relations. Clicking on the `Generate Repairing Actions` button, results in the computation of repairing actions for each wrong is-a relation of the ontology under repair. The wrong is-a relations are then ranked in ascending order according to the number of possible repairing actions and shown in a drop-down list. Then, the user can select a wrong is-a relation and repair it using an interactive display. The user can choose to repair all wrong is-a relations in groups or one by one. The display shows a directed graph representing the justifications. The nodes represent concepts. As mentioned before, concepts in different ontologies are presented with different background

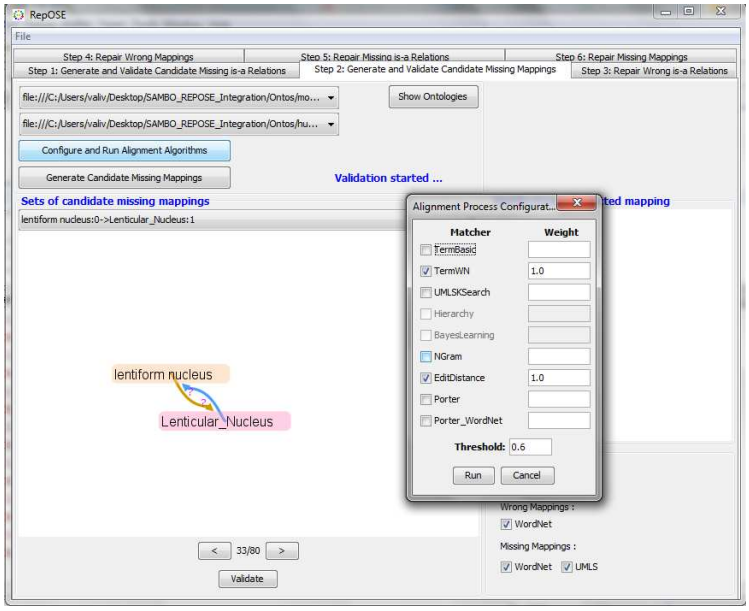

Fig. 8. Aligning.

color. The concepts in the is-a relation under repair are shown in red. The edges represent is-a relations in the justifications. These is-a relations may be existing asserted is-a relations (shown in grey), mappings (shown in brown), unrepaired missing is-a relations (shown in blue) and the added repairing actions for the repaired missing is-a relations (shown in black).

In Figure 9 the user has chosen to repair several wrong is-a relations at the same time, i.e., *(brain grey matter, white matter)*, *(cerebellum white matter, brain grey matter)*, and *(cerebral white matter, brain grey matter)*. In this example[9] we can repair these wrong is-a relations by removing the mappings between *brain grey matter* and *Brain_White_Matter*. We note that, when removing these mappings, all these wrong is-relations will be repaired at the same time.

For the wrong is-a relations under repair, the user can choose, by clicking, multiple existing asserted is-a relations and mappings on the display as repairing actions and click the `Repair` button. RepOSE ensures that only existing asserted is-a relations and mappings are selectable, and when the user finalizes the repair decision, RepOSE ensures that the wrong is-a relations under repair and every selected is-a relation and mapping will not be derivable from the ontology network after the repairing. Further,

[9] From OAEI 2010 Anatomy.

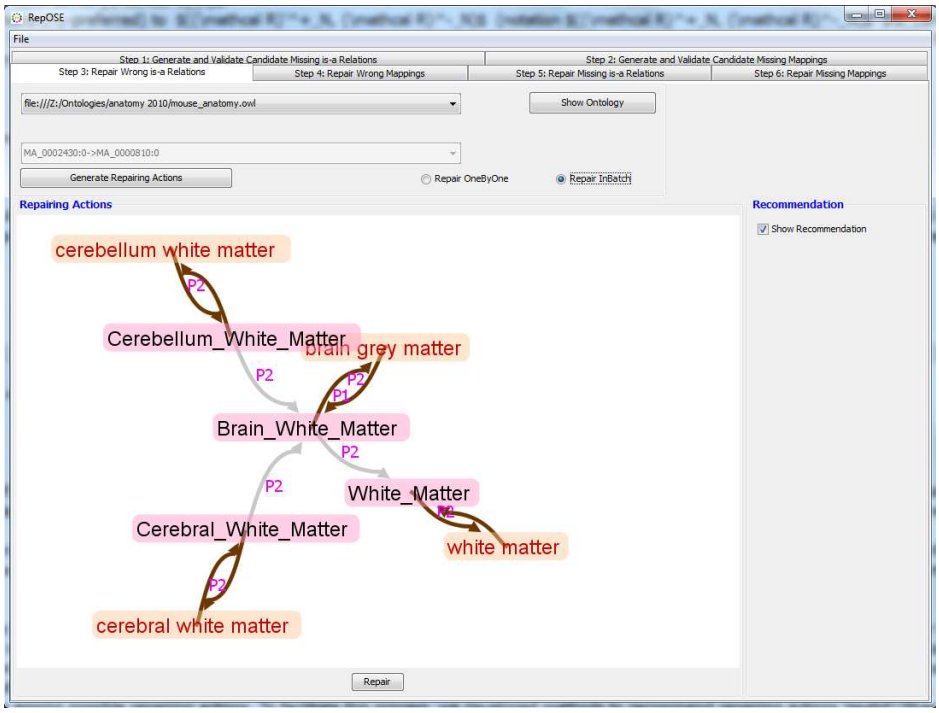

Fig. 9. Repairing wrong is-a relations.

all consequences of the repair are computed (such as changes in the repairing actions of other is-a relations and mappings and changes in the lists of wrong and missing is-a relations and mappings).

A similar tab ('Step 4: Repair Wrong Mappings') is used for repairing wrong mappings.

Repairing missing is-a relations and mappings.

Figure 10 shows the RepOSE tab 'Step 5: Repair Missing is-a Relations' for repairing missing is-a relations. Clicking on the `Generate Repairing Actions` button, results in the computation of repairing actions for the missing is-a relations of the ontology under repair. For easy visualization, these are shown to the user as Source and Target sets (instead of Repair). Once the Source and Target sets are computed, the missing is-a relations are ranked with respect to the number of possible repairing actions. The first missing is-a relation in the list has the fewest possible repairing actions, and may therefore be a good starting point. When the user chooses a missing is-a relation, its Source and Target sets are displayed on the left and right, respectively, within the `Repairing Actions` panel (Figure 10). Both have zoom control and can be opened in a separate window. Similarly to the displays for wrong is-a relations, con-

cepts in the missing is-a relations are highlighted in red, existing asserted is-a relations are shown in grey, unrepaired missing is-a relations in blue and added repairing actions for the missing is-a relations in black. For instance, Figure 10 shows the Source and Target sets for the missing is-a relation *(lower respiratory tract cartilage, cartilage)*, which contain 2 and 21 concepts, respectively. The Target panel shows also the unrepaired missing is-a relation *(nasal septum, nasal cartilage)*. The `Justifications of current relation` panel is a read-only panel that displays the justifications of the current missing is-a relation as an extra aid.

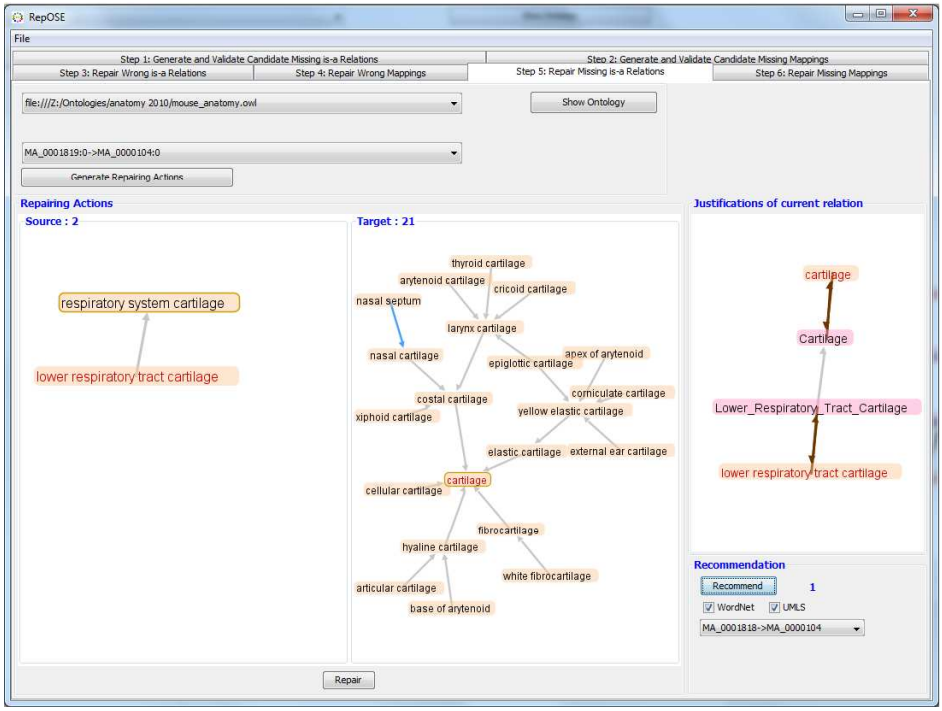

Fig. 10. Repairing missing is-a relations.

The user can repair the missing is-a relation by selecting a concept in the Source panel and a concept in the Target panel and clicking on the `Repair` button. When the selected repairing action is not in $Repair(a, b)$, the repairing will not be allowed and a message window is shown to the user. Further, all consequences of a chosen repair are computed (such as changes in the repairing actions of other is-a relations and mappings and changes in the lists of wrong and missing is-a relations and mappings).

The tab 'Step 6: Repair Missing Mappings' is used for repairing missing mappings. The main difference with the tab for repairing missing is-a relations is that we deal with

	candidate missing mappings	missing ≡/← or →	wrong ≡/← or →	repair missing ≡/←/→/derivable /more informative	repair missing is-relations
Alignment	1384	1286/39	59/39	1286/21/8/5/5	-
AMA	-	-	-	-	3
NCI-A	-	-	-	-	2

Fig. 11. Experiment 1 results - debugging of the alignment.

two ontologies and their alignment and that the repairing actions can be is-a relations within an ontology as well as mappings.

5 Experiments

We performed three experiments to demonstrate the benefits of the integrated ontology alignment and debugging framework. As input for Experiment 1 and 2 we used the two ontologies from the Anatomy track of OAEI 2011 - AMA contains 2,737 concepts and 1,807 asserted is-a relations, and NCI-A contains 3,298 concepts and 3,761 asserted is-a relations. The input for the last experiment contained the reference alignment (1516 equivalence mappings between AMA and NCI-A) together with the two ontologies. The reference alignment was used indirectly as external knowledge during the validation phase in the first two experiments. The experiments were performed on an Intel Core i7-2620M Processor 2.7GHz with 4 GB memory underWindows 7 Professional operating system and Java 1.7 compiler.

Experiment 1 - aligning and debugging OAEI Anatomy. The first experiment demonstrates a complete debugging and aligning session where the input is a set with the two ontologies. After loading the ontologies mapping suggestions were computed using matchers TermWN and UMLSM, weight 1 for both and threshold 0.5. This resulted in 1384 mapping suggestions. The 1233 mapping suggestions that are also in the reference alignment were validated as missing equivalence mappings (although, as we will see, there are defects in the reference alignment) and repaired by adding them to the alignment. The others were validated manually and resulted in missing mappings (53 equivalence and 39 is-a) and wrong mappings (59 equivalence and 39 is-a). These missing mappings were repaired by adding 53 equivalence and 29 is-a mappings, 5 more informative is-a mappings and 5 is-a relations (3 to AMA and 2 to NCI-A). 5 of these missing mappings were repaired by repairing others. Among the wrong mappings there were 3 which were derivable in the network. These were repaired by removing 2 is-a relations from NCI-A. Figure 11 summarizes the results.

The generated alignment was then used in the debugging of the network created by the ontologies and the alignment. Two iterations of the debugging workflow were performed, since the repairing of wrong and missing is-a relations in the first iteration led to the detection of new CMIs which had to be validated and repaired. Over 90% of the CMIs for both ontologies were detected during the first iteration, the detection of CMIs took less than 30 seconds per ontology. Figure 12 summarizes the results.

	candidate missing all/non-redundant	missing	wrong	repair wrong removed	repair missing self/more informative/other
AMA	410/263	224	39	30	144/57/23
NCI-A	355/183	166	17	17	127/13/26
Alignment	-	-	-	8 $\equiv$ and 1 $\rightarrow$	-

Fig. 12. Experiment 1 results - debugging of the ontologies.

In total the system detected 263 non-redundant (410 in total) CMIs for AMA and 183 non-redundant (355 in total) CMIs for NCI-A. The non-redundant CMIs were displayed in groups, 45 groups for AMA and 31 for NCI-A. Among the 263 non-redundant CMIs in AMA 224 were validated as correct and 39 as wrong. In NCI-A 166 were validated as correct and 17 as wrong. The 39 wrong is-a relations in AMA were repaired by removing 30 is-a relations from NCI-A, and 8 equivalence and 1 is-a mapping from the alignment. The 17 wrong is-a relations in NCI-A were repaired by removing 17 is-a relations in AMA. The missing is-a relations in AMA were repaired by adding 201 is-a relations - in 144 cases the missing is-a relation itself and in 57 cases a more informative is-a relation. 23 of the 224 missing is-a relations became derivable after repairing some of the others. To repair the missing is-a relations in NCI-A 140 is-a relations were added - in 127 cases the missing is-a relation itself and in 13 cases a more informative is-a relation. 26 out of the 166 missing is-a relations were repaired while other is-a relations were repaired.

We observe that for 57 missing is-a relations in AMA and 13 in NCI-A the repairing actions are more informative than the missing is-a relation itself. This means that for each of these, knowledge, which was not derivable from the network before, was added to the network. Thus the knowledge represented by the ontologies and the network has increased.

Experiment 2. For this experiment the alignment process was run twice and at the end the alignments were compared. The same matchers, weights and threshold as in Experiment 1 were used. During both runs the CMMs (mapping suggestions) were computed and validated in the same manner. This step is as in Experiment 1 and the results are the ones in Figure 11. The difference between both runs is in the repairing phase. In the first run the missing mappings were repaired by directly adding them to the final alignment without benefiting from the repairing algorithms - in the same way most of the alignment systems do. The final alignment contained 1286 equivalence and 44 is-a[10] mappings.

During the repairing phase in the second run the debugging component was used to provide alternative repairing actions than those available in the initial set of mapping suggestions. The final alignment then contained 1286 equivalence mappings from the mapping suggestions, 21 is-a mappings from the mapping suggestions, and 5 more informative is-a mappings, thus adding knowledge to the network. Further, 5 mapping suggestions were repaired adding is-a relations (3 in AMA and 2 in NCI-A) and thus

[10] 5 of these are repaired in the second run by adding is-a relations in the ontologies.

adding more knowledge to each of the ontologies. 5 more mapping suggestions became derivable from the network as a result from the repairing actions for other CMMs.

Experiment 3. In this experiment the debugging process was run twice, CMIs were detected for both ontologies and compared between the runs. The input for the first run was the set of the two ontologies and their alignment from the Anatomy track in OAEI 2011. The network was loaded in the system and the CMIs were detected. 496 CMIs were detected for AMA, of which 280 were non-redundant. For NCI-A 365 CMIs were detected of which 193 were non-redundant. The same input was used in the second run. However, the alignment algorithms were used to extend the set with mappings prior to generating the CMIs. The set-up for the aligning was the same as in Experiment 1 and the mapping suggestions were computed, validated and repaired in the same way as well. Then CMIs were generated - 638 CMIs were detected for AMA of which 357 were non-redundant, and 460 CMIs for NCI-A, of which 234 were non-redundant. In total 145 new CMIs were detected for AMA - 120 were validated as missing and 25 validated as wrong[11]. For NCI-A 103 new CMIs were detected - 53 were validated as missing and 50 as wrong.

Discussion. Experiment 1 shows the usefulness of the system through a complete session where an alignment was generated and many defects in the ontologies were repaired. Some of the repairs added new knowledge. As a side effect, we have shown that the ontologies that are used by the OAEI contain over 200 and 150 missing is-a relations, respectively and 39 and 17 wrong is-a relations, respectively. We have also shown that the alignment is not complete and contains wrong information. We also note that our system allows validation and allows a domain expert to distinguish between equivalence and is-a mappings. Most ontology alignment systems do not support this.

Experiment 2 shows the advantages for ontology alignment when also a debugging component is added. The debugging component allowed to add more informative mappings, reduce redundancy in the alignment as well as debug the ontologies leading to further reduced redundancy in the alignment. For the ontologies and alignment new knowledge not found when only aligning, was added. In general, the quality of the final alignment (and the ontologies) becomes higher.

Experiment 3 shows that the debugging process can take advantage of the alignment component even when an alignment is available. The alignment algorithms can provide additional mapping suggestions and thus extending the alignment. More mappings between two ontologies means higher coverage and possibly more detected and repaired defects. In the experiment more than 100 CMIs (of which many correct) were detected for each ontology using the extended set of mappings. We also note that the initial alignment contained many mappings (1516). In the case that the alignment contains fewer mappings the benefit to the debugging process will be even more significant.

[11] The sum of the newly generated CMIs and those in the first run is not equal to the number of the CMIs in the second run because some of the CMIs generated in the first run are derivable in the second run.

6 Related Work

To our knowledge there is no other system that integrates ontology debugging and ontology alignment in a uniform way and that allows for a strong interleaving of these tasks. There are some ontology alignment systems that do semantic verification and disallow mappings that lead to unsatisfiable concepts (e.g., [12, 14]). Further, adding missing is-a relations to ontologies was a step in the alignment process in [19].

Regarding the debugging component, this work extends the work in [21, 20] that dealt with debugging is-a structure in taxonomy networks. These were one of the few approaches dealing with repairing missing is-a structure and in the case of [20] debugging both missing and wrong is-a structure. The current work extends this by also including debugging of mappings in a uniform way as well as ontology alignment. The ontology alignment component also removed the restriction of [20] that required the existence of an initial alignment.

There are different ways to *detect* missing is-a relations. One way is by inspection of the ontologies by domain experts. Another way is to use external knowledge sources. For instance, there is much work on finding relationships between terms in the ontology learning area [2]. Regarding the detection of is-a relations, one paradigm is based on linguistics using lexico-syntactic patterns. The pioneering research conducted in this line is in [9], which defines a set of patterns indicating is-a relationships between words in the text. Another paradigm is based on machine learning and statistical methods. Further, guidelines based on logical patterns can be used [4]. These approaches are complementary to the approach used in this paper. There is, however, not much work on the *repairing* of missing is-a relations that goes beyond adding them to the ontologies except for [21] for taxonomies and [18] for $\mathcal{ALC}$ acyclic terminologies.

There is more work on the debugging of semantic defects. Most of it aims at identifying and removing logical contradictions from an ontology. Standard reasoners are used to identify the existence of a contradiction, and provide support for resolving and eliminating it [7]. In [26] minimal sets of axioms are identified which need to be removed to render an ontology coherent. In [17, 16] strategies are described for repairing unsatisfiable concepts detected by reasoners, explanation of errors, ranking erroneous axioms, and generating repair plans. In [8] the focus is on maintaining the consistency as the ontology evolves through a formalization of the semantics of change for ontologies. [28] introduces a method for interactive ontology debugging. In [24] and [13] the setting is extended to repairing ontologies connected by mappings. In this case, semantic defects may be introduced by integrating ontologies. Both works assume that ontologies are more reliable than the mappings and try to remove some of the mappings to restore consistency. The solutions are often based on the computation of minimal unsatisfiability-preserving sets or minimal conflict sets. The work in [25] further characterizes the problem as mapping revision. Using belief revision theory, the authors give an analysis for the logical properties of the revision algorithms. Another approach for debugging mappings is proposed in [30] where the authors focus on the detection of certain kinds of defects and redundancy. The approach in [15] deals with the inconsistencies introduced by the integration of ontologies, and unintended entailments validated by the user.

Regarding the alignment component there are some systems that allow validation of mappings such as SAMBO [23], COGZ [6] for PROMPT, and COMA++ [5]. Many matchers have been proposed (e.g. many papers on http://ontologymatching.org/), and most systems use similar combination and filtering strategies as in this paper. For an overview we refer to [27].

7 Conclusion

In this paper we presented a unified approach for aligning taxonomies and debugging taxonomies and their alignments. This is the first approach which integrates ontology alignment and ontology debugging and allows debugging of both the structure of the ontologies as well as their alignments. Further, we have shown the benefits of our approach through experiments. The interactions between ontology alignment and debugging significantly raise the quality of both taxonomies and their alignments. The ontology alignment provides or extends alignments that are used by the debugging. The debugging provides algorithms for repairing defects in alignments and possibly add new knowledge.

We will continue exploring the interactions between ontology alignment and debugging. We will include and investigate the benefits when using structure-based alignment algorithms and partial-alignment-based techniques. Further, we will investigate the debugging problem for ontologies represented in more expressive formalisms.

Acknowledgements. We thank the Swedish Research Council (Vetenskapsrådet) and the Swedish e-Science Research Centre (SeRC) for financial support.

References

1. F Baader, D Calvanese, D McGuinness, D Nardi, and P Patel-Schneider. *The description logic handbook*. Cambridge University Press, 2003.
2. Ph Cimiano, P Buitelaar, and B Magnini. *Ontology Learning from Text: Methods, Evaluation and Applications*. IOS Press, 2005.
3. C Conroy, R Brennan, D O'Sullivan, and D Lewis. User Evaluation Study of a Tagging Approach to Semantic Mapping. In *6th European Semantic Web Conference, LNCS 5554*, pages 623–637, 2009.
4. O Corcho, C Roussey, L M Vilches, and I Pérez. Pattern-based OWL ontology debugging guidelines. In *Workshop on Ontology Patterns*, pages 68–82, 2009.
5. H-H Do and E Rahm. Matching large schemas: approaches and evaluation. *Information Systems*, 32:857–885, 2007.
6. S Falconer and M-A Storey. A cognitive support framework for ontology mapping. In *6th International Semantic Web Conference and 2nd Asian Semantic Web Conference, LNCS 4825*, pages 114–127, 2007.
7. G Flouris, D Manakanatas, H Kondylakis, D Plexousakis, and G Antoniou. Ontology change: Classification and survey. *Knowledge Engineering Review*, 23(2):117–152, 2008.
8. P Haase and L Stojanovic. Consistent Evolution of OWL Ontologies. In *2nd European Semantic Web Conference, LNCS 3532*, pages 182–197. 2005.

9. M Hearst. Automatic acquisition of hyponyms from large text corpora. In *14th International Conference on Computational Linguistics*, pages 539–545, 1992.
10. V Ivanova and P Lambrix. A system for aligning taxonomies and debugging taxonomies and their alignments. In *10th Extended Semantic Web Conference - ESWC 2013 Satellite Events, LNCS 7955*, pages 152–156, 2013.
11. V Ivanova and P Lambrix. A unified approach for aligning taxonomies and debugging taxonomies and their alignments. In *10th Extended Semantic Web Conference, LNCS 7882*, pages 1–15, 2013.
12. YR Jean-Mary, EP Shironoshita, and MR Kabuka. Ontology matching with semantic verification. *Journal of Web Semantics*, 7(3):235–251, 2009.
13. Q Ji, P Haase, G Qi, P Hitzler, and S Stadtmuller. RaDON - repair and diagnosis in ontology networks. In *6th European Semantic Web Conference, LNCS 5554*, pages 863–867, 2009.
14. E Jimenez-Ruiz, B Cuenca-Grau, Y Zhou, and I Horrocks. Large-scale interactive ontology matching: Algorithms and implementation. In *20th European Conference on Artificial Intelligence*, pages 444–449, 2012.
15. E Jimenez-Ruiz, B Cuenca Grau, I Horrocks, and R Berlanga. Ontology integration using mappings: Towards getting the right logical consequences. In *6th European Semantic Web Conference, LNCS 5554*, pages 173–187, 2009.
16. A Kalyanpur, B Parsia, E Sirin, and B Cuenca-Gray. Repairing Unsatisfiable Concepts in OWL Ontologies. In *3rd European Semantic Web Conference, LNCS 4011*, pages 170–184, 2006.
17. A Kalyanpur, B Parsia, E Sirin, and J Hendler. Debugging Unsatisfiable Classes in OWL Ontologies. *Journal of Web Semantics*, 3(4):268–293, 2006.
18. P Lambrix, Z Dragisic, and V Ivanova. Get my pizza right: Repairing missing is-a relations in ALC ontologies. In *2nd Joint International Semantic Technology Conference*, 2012.
19. P Lambrix and Q Liu. Using partial reference alignments to align ontologies. In *6th European Semantic Web Conference, LNCS 5554*, pages 188–202, 2009.
20. P Lambrix and Q Liu. Debugging is-a structure in networked taxonomies. In *4th International Workshop on Semantic Web Applications and Tools for Life Sciences*, pages 58–65, 2011.
21. P Lambrix, Q Liu, and H Tan. Repairing the missing is-a structure of ontologies. In *4th Asian Semantic Web Conference, LNCS 5926*, pages 76–90, 2009.
22. P Lambrix, G Qi, and M Horridge. *Proceedings of the 1st International Workshop on Debugging Ontologies and Ontology Mappings*. LiU E-Press, LECP 79, 2012.
23. P Lambrix and H Tan. SAMBO - a system for aligning and merging biomedical ontologies. *Journal of Web Semantics*, 4(3):196–206, 2006.
24. C Meilicke, H Stuckenschmidt, and A Tamilin. Repairing Ontology Mappings. In *22th Conference on Artificial Intelligence*, pages 1408–1413, 2007.
25. G Qi, Q Ji, and P Haase. A Conflict-Based Operator for Mapping Revision. In *8th International Semantic Web Conference, LNCS 5823*, pages 521–536, 2009.
26. S Schlobach. Debugging and Semantic Clarification by Pinpointing. In *2nd European Semantic Web Conference, LNCS 3532*, pages 226–240, 2005.
27. P Schvaiko and J Euzenat. Ontology matching: state of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering*, 25(1):158–176, 2013.
28. K Shchekotykhin, G Friedrich, Ph Fleiss, and P Rodler. Interactive ontology debugging: Two query strategies for efficient fault localization. *Journal of Web Semantics*, 12-13:88–103, 2012.
29. UMLS. Unified medical language system. `http://www.nlm.nih.gov/research/umls/about_umls.html`.
30. P Wang and B Xu. Debugging ontology mappings: a static approach. *Computing and Informatics*, 27:21–36, 2008.

31. WordNet. http://wordnet.princeton.edu/.