Debugging the missing is-a structure within taxonomies networked by partial reference alignments

Patrick Lambrix[1,2*], Qiang Liu[1]

(1) Department of Computer and Information Science
(2) Swedish e-Science Research Centre
Linköping University
SE-581 83 Linköping, Sweden

Abstract

With the proliferation of ontologies and their use in semantically-enabled applications, the issue of finding and repairing defects in ontologies has become increasingly important. Current work mostly targets debugging semantic defects in ontologies. In our work, we focus on another kind of severe defects, modeling defects, which require domain knowledge to detect and resolve. In particular, we debug the missing structural relations (is-a hierarchy) in a fundamental kind of ontologies, i.e. taxonomies. The context of our study is an ontology network consisting of several taxonomies networked by partial reference alignments. We use the ontology network as domain knowledge to detect the missing is-a relations in these ontologies. We also propose algorithms to generate possible repairing actions, rank missing is-a relations, recommend and execute repairing actions. Further, we discuss an implemented system RepOSE and experiments on ontologies of the Ontology Alignment Evaluation Initiative and the Finnish Ontology Library Service.

Keywords: Ontologies, Ontology engineering, Ontology debugging

This is a preprint of an article published in Data and Knowledge Engineering 86:179-205, 2013. http://dx.doi.org/10.1016/j.datak.2013.03.003

1. Introduction

Developing ontologies is not an easy task and often the resulting ontologies are not consistent or complete. Such ontologies, although often useful, also lead to problems when used in semantically-enabled applications. Wrong conclusions may be derived or valid conclusions may be missed. Semantically-enabled applications require high-quality ontologies and mappings. A key step towards this is debugging, i.e., detecting and repairing defects in, the ontologies and their alignment. It has been realized that

*corresponding author, Patrick.Lambrix@liu.se, Tel: +46 13 282605

this is an important issue and ontology debugging is currently establishing itself as a sub-field of ontology engineering with the first workshop on debugging ontologies and ontology mappings having been held in 2012 [40].

Defects in ontologies can take different forms (e.g. [35]). Syntactic defects are usually easy to find and to resolve. Defects regarding style include such things as unintended redundancy. More interesting and severe defects are the modeling defects which require domain knowledge to detect and resolve, and semantic defects such as unsatisfiable concepts and inconsistent ontologies. Most work up to date has focused on detecting and repairing the semantic defects in an ontology (e.g. [35, 34, 33, 54]). Recent work has also started looking at repairing semantic defects in a set of mapped ontologies [30, 29] or the mappings between ontologies themselves [44, 63, 52]. In this work we tackle the problem of repairing modeling defects and in particular, the repairing of the is-a structure of ontologies.

In addition to its importance for the correct modeling of a domain, the structural information in ontologies is also important in semantically-enabled applications. For instance, the is-a structure is used in ontology-based search and annotation. In ontology-based search, queries are refined and expanded by moving up and down the hierarchy of concepts. Incomplete structure in ontologies influences the quality of the search results. As an example, suppose we want to find articles in the MeSH (Medical Subject Headings [45], controlled vocabulary of the National Library of Medicine, US) Database of PubMed [51] using the term *Scleral Diseases* in MeSH. By default the query will follow the hierarchy of MeSH and include more specific terms for searching, such as *Scleritis*. If the relation between *Scleral Diseases* and *Scleritis* is missing in MeSH, we will miss 738 articles in the search result, which is about 55% of the original result. The structural information is also important information in ontology engineering research. For instance, most current ontology alignment systems use structure-based strategies to find mappings between the terms in different ontologies (e.g. overview in [41]) and the modeling defects in the structure of the ontologies have an important influence on the quality of the ontology alignment results [1].

As the ontologies grow in size, it is difficult to ensure the correctness and completeness of the structure of the ontologies. Some structural relations may be missing or some existing or derivable relations may be unintended. Detecting and resolving these defects requires, in contrast to semantic defects, the use of domain knowledge. One interesting kind of domain knowledge is the other ontologies and information about connections between these ontologies. For instance, in the case of the Anatomy track in the 2008 and 2009 Ontology Alignment Evaluation Initiative (OAEI) two ontologies, Adult Mouse Anatomy Dictionary [3] (MA, 2744 concepts) and the NCI Thesaurus anatomy [46] (NCI-A, 3304 concepts), and a partial reference alignment (PRA, a set of correct mappings between the terms of the ontologies) containing 988 mappings are given. Using one ontology and the mappings as domain knowledge for the other ontology (and vice versa), it was shown in [38] that at least 121 is-a relations in MA and 83 in NCI-A are missing and should be repaired. This is not an uncommon case. It is well-known that people that are not expert in knowledge representation often misuse and confuse equivalence, is-a and part-of (e.g. [10]), which leads to problems in the structure of the ontologies.

Once the missing is-a relations are found, the structure of the ontology can be

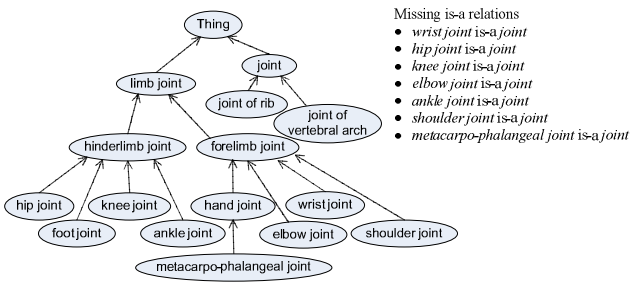

Figure 1: A part of MA concerning the concept *joint*.

repaired by adding the missing is-a relations themselves, but this is not always the most interesting solution for a domain expert. For instance, Figure 1 shows a part of MA regarding the concept *joint* (is-a relations shown with arrows). Using NCI-A and the PRA as domain knowledge, 7 missing is-a relations are found. The ontology could be repaired by adding these missing is-a relations themselves. However, for the missing is-a relation "*wrist joint* is-a *joint*", knowing that there is an is-a relation between *wrist joint* and *limb joint*, a domain expert will most likely prefer to add the is-a relation "*limb joint* is-a *joint*" instead. This is correct from a modeling perspective as well as more informative and would lead to the fact that the missing is-a relation between *wrist joint* and *joint* can be derived. In this particular case, using "*limb joint* is-a *joint*" would actually also lead to the repairing of the other 6 missing is-a relations, as well as others that were not found before (e.g. "*hand joint* is-a *joint*"). In general, such a decision should be made by domain experts.

In this paper, we deal with detecting and repairing the missing is-a structure in ontologies in the context of domain knowledge represented by the ontology network. Assuming that the existing is-a relations in the ontologies are correct, as well as the mappings in the PRAs, we use them as domain knowledge to detect the missing is-a relations in these ontologies. We also develop algorithms to generate and recommend possible ways of repairing, which are relevant for domain experts, as well as algorithms to rank missing is-a relations and execute the repairing. Further, we develop the system RepOSE (*Rep*air of *O*ntological *S*tructure *E*nvironment), which allows a domain expert to debug the missing is-a structure of ontologies in a semi-automatic way.

Before we introduce our work, we note that the 'is-a relation' is still not well-understood and/or used. For instance, in [6], an analysis of links in semantic networks, different kinds of is-a were identified including set/superset, generalization/specialization (based on predicates), 'a kind of', and conceptual containment (related to lambda-abstraction). In [31] the authors argue for four kinds of is-a: genus-subsumption, determinable-subsumption, specification and specialization. For the genus-subsumption the classes in the is-a relation have monadic qualities by which they can be characterized (e.g. classification of trees). Determinable-subsumption deals with qualities and characterization is based on similarity relations to other qualities (e.g. scarlet is-a red). Specification covers things such as 'careful painting' is-a 'painting' while specialization covers 'house painting' is-a 'painting'. In this case multiple inheritance does not

make sense for all kinds of is-a relations and in many information systems the different kinds are mixed. The latter is also addressed in [19] where the author discusses the problem of is-a overloading. Based on the notions of identity, rigidity and dependence, it is shown that not all is-a relations in existing ontologies make sense. These difficulties are not always recognized by ontology builders, while some may decide to use one kind of is-a relation. For instance, the Relation Ontology [58] for OBO defined the is-a relation for OBO ontologies, but is now superseded by RO [53] in which no more definition for is-a is given, but instead the subclass construct of OWL is used. The work in this paper is based on logic and we assume that the is-a relation is reflexive, antisymmetric and transitive. The detection and repairing of missing is-a relations in our work is based on logical reasoning on the ontologies and their PRAs. Our debugging tool does not take into account different kinds of is-a relation. Instead, it provides support for detecting and repairing missing structure that logically follows from decisions that were made by the developers of the ontologies and the PRAs.

The remainder of this paper is organized as follows. In Section 2 we present the theory for our debugging approach. The overview of the whole process is given in Section 3. Section 4 introduces the algorithms for the detection process, while Section 5 explains the algorithms for the repairing process which involves generating repairing actions, ranking missing is-a relations, recommending and executing repairing actions. Our system RepOSE and its use are described in Section 6. Further, we describe experiments in Section 7 and discuss lessons learned and advantages and limitations of our approach in Section 8. Related work is presented in Section 9 and the paper concludes in Section 10.

2. Theory

Our approach for debugging missing is-a relations in an ontology network contains two parts, i.e. detecting and repairing. The former deals with the identification of the missing is-a relations in the networked ontologies, while the latter deals with repairing the structure of the ontologies.

2.1. Preliminaries

The setting that we study is the case where the ontologies are defined using named concepts and subsumption axioms. Most ontologies contain this case and many of the most well-known and used ontologies, e.g. in the life sciences, are covered by this setting.

Definition 1. *An ontology $\mathcal{O}$ is represented by a tuple $(\mathcal{C}, \mathcal{I})$ with $\mathcal{C}$ its set of named concepts and $\mathcal{I} \subseteq \mathcal{C} \times \mathcal{C}$ a representation of its is-a structure.*

A PRA between two ontologies contains a set of correct mappings between the concepts of different ontologies. In this paper we consider *equivalent* ($\equiv$), *subsumed-by* ($\rightarrow$) and *subsumes* ($\leftarrow$) mappings.[1] We assume that concepts can participate in multiple mappings.

[1]We note that for a PRA between $\mathcal{O}_i$ and $\mathcal{O}_j$, there is a corresponding PRA between $\mathcal{O}_j$ and $\mathcal{O}_i$, such that there is a mapping $c_i r c_j$ in the former iff there is a corresponding mapping $c_j r^{-1} c_i$ in the latter, where

Definition 2. *A* **Partial Reference Alignment** *(PRA) between $\mathcal{O}_i$ and $\mathcal{O}_j$, is represented by a set $\mathcal{P}_{ij}$ of pairs representing is-a relations, such that for each mapping in the PRA with c_i a concept in $\mathcal{O}_i$ and c_j a concept in $\mathcal{O}_j$: $c_i \rightarrow c_j$ is represented by (c_i, c_j) in $\mathcal{P}_{ij}$; $c_i \leftarrow c_j$ 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).*

An ontology network contains ontologies and PRAs between ontologies. The domain knowledge of an ontology network is represented by its *induced ontology*.

Definition 3. *An* ***ontology network*** *$\mathcal{N}$ is a tuple $(\mathbb{O}, \mathbb{P})$ with $\mathbb{O} = \{\mathcal{O}_k\}_{k=1}^{n}$ where $\mathcal{O}_k = (\mathcal{C}_k, \mathcal{I}_k)$, the set of the ontologies in the network (called the networked ontologies) and $\mathbb{P} = \{\mathcal{P}_{ij}\}_{i,j=1;i<j}^{n}$ the set of representations for the PRAs between these ontologies. Further, the* ***induced ontology*** *$\mathcal{O}_N$ for $\mathcal{N}$ is an ontology $\mathcal{O}_N = (\mathcal{C}_N, \mathcal{I}_N)$ such that $\mathcal{C}_N = \cup_{k=1}^{n} \mathcal{C}_k$ and $\mathcal{I}_N = \cup_{k=1}^{n} \mathcal{I}_k \cup \cup_{i,j=1;i<j}^{n} \mathcal{P}_{ij}$.*

In the remainder of the paper we assume that the sets of named concepts for the different ontologies in the network are disjoint.

2.2. Theory for detecting

Given an ontology network, assuming that all existing is-a relations in the ontologies are correct, we use the domain knowledge of the ontology network to detect the missing is-a relations in these networked ontologies. For each ontology in the network, the set of missing is-a relations 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 induced ontology of the network, but not from the networked ontology alone.

Definition 4. *Given an ontology network $\mathcal{N} = (\mathbb{O}, \mathbb{P})$ where $\mathbb{O} = \{\mathcal{O}_k\}_{k=1}^{n}$ and $\mathcal{O}_k = (\mathcal{C}_k, \mathcal{I}_k)$. Then, the set of* ***missing is-a relations for the networked ontology $\mathcal{O}_k$ derivable from the ontology network*** *$\mathcal{N}$, denoted by $\mathcal{M}_k$, is the set of is-a relations $\{(a, b) \in \mathcal{C}_k \times \mathcal{C}_k | \, \mathcal{O}_N \models a \rightarrow b \wedge \mathcal{O}_k \not\models a \rightarrow b\}$. Further, the set of* ***missing is-a relations for the networked ontologies $\mathbb{O}$ derivable from the ontology network*** *$\mathcal{N}$, denoted by $\mathcal{M}_N$, is the set of is-a relations $\cup_{k=1}^{n} \mathcal{M}_k$.*

As an example, consider the ontology network in Figure 2. It contains two ontologies with their is-a hierarchies (marked by the solid arrows), which are related via 3 mappings with equivalence relations (marked by the dashed lines). According to the definition above, there are two missing is-a relations derivable from the network, $(ankle\_joint_1, joint_1)$ in ontology 1 and $(ankle\_joint_2, limb\_joint_2)$ in ontology 2 (marked by the dashed arrows). Domain experts may argue that some is-a relations, such as $(knee\_joint_1, joint_1)$ and $(hip\_joint_2, limb\_joint_2)$, are also missing is-a relations. However, these cannot be found using logical derivation within the network

r^{-1} denotes the inverse relation of r. The inverse relation for *equivalent* is *equivalent*, and *subsumes* and *subsumed-by* are each other's inverse relation.

and are thus not missing is-a relations *derivable* from the network as defined in Definition 4. From now on in this paper, whenever missing is-a relations are mentioned, we mean the missing is-a relations derivable from the network, unless explicitly stated otherwise.[2]

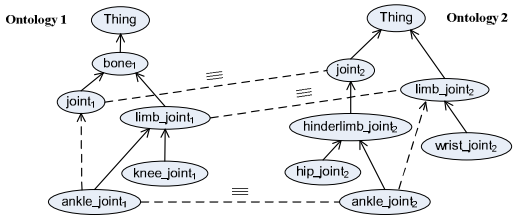

Figure 2: An example ontology network.

2.3. Theory for repairing

Our goal for repairing is to repair the original ontologies by adding a set of is-a relations, called a *structural repair*, to each ontology such that the missing is-a relations can be derived from the ontology extended with the newly-added is-a relations. Therefore, the structural repair only contains is-a relations between concepts within the same ontology. The elements in a structural repair are called *repairing actions*.

Definition 5. *Let $\mathcal{M}_N = \cup_{k=1}^{n} \mathcal{M}_k$ be the set of missing is-a relations for an ontology network $\mathcal{N} = (\mathbb{O}, \mathbb{P})$ where $\mathbb{O} = \{\mathcal{O}_k\}_{k=1}^{n}$, $\mathcal{O}_k = (\mathcal{C}_k, \mathcal{I}_k)$ and $\mathcal{M}_k$ the set of missing is-a relations for $\mathcal{O}_k$ derivable from $\mathcal{N}$. Then, a **structural repair for the networked ontology** $\mathcal{O}_k$ with respect to $\mathcal{M}_k$, denoted by $\mathcal{R}_k$, is a set of is-a relations such that $\mathcal{R}_k \subseteq \mathcal{C}_k \times \mathcal{C}_k$ and for each missing is-a relation $(a, b) \in \mathcal{M}_k$, $(\mathcal{C}_k, \mathcal{I}_k \cup \mathcal{R}_k) \models a \rightarrow b$. Further, a **structural repair for the networked ontologies** $\mathbb{O}$ with respect to $\mathcal{M}_N$, denoted by $\mathcal{R}_N$, is a set of is-a relations such that $\mathcal{R}_N = \cup_{k=1}^{n} \mathcal{R}_k$, where $\mathcal{R}_k$ is a structural repair for $\mathcal{O}_k$ with respect to $\mathcal{M}_k$.*

An immediate consequence of the definition is that, for the networked ontologies, the set of missing is-a relations is in itself a structural repair. Another consequence is that adding is-a relations between concepts of any single ontology in the network to a structural repair for the networked ontologies also constitutes a structural repair.

As mentioned in Section 1, not all structural repairs are equally useful or interesting for a domain expert. Therefore, we define a number of heuristics and preference relations between structural repairs for the networked ontologies.

Not all repairing actions are always needed in a structural repair. For example, in the case of Figure 2, $\{(ankle\_joint_1, joint_1), (knee\_joint_1, joint_1), (ankle\_joint_2, limb\_joint_2)\}$ is a structural repair, but repairing action $(knee\_joint_1, joint_1)$ is not

[2]For an overview of the terminology in this paper regarding 'missing is-a relation', see Table 19 in the appendix.

needed for repairing the missing is-a relations. Therefore, $\{(ankle\_joint_1, joint_1), (ankle\_joint_2, limb\_joint_2)\}$ which is also a structural repair, is preferred to $\{(ankle\_joint_1, joint_1), (knee\_joint_1, joint_1), (ankle\_joint_2, limb\_joint_2)\}$. Further, as we saw before in relation to the example in Figure 1, it may also happen that several missing is-a relations can be repaired by the same repairing action. For instance, for a network including the ontology in Figure 1, a structural repair that includes repairing action $(limb\_joint, joint)$ would repair both $(wrist\_joint, joint)$ and $(knee\_joint, joint)$. Therefore, in a structural repair that includes both the repairing actions $(limb\_joint, joint)$ and $(knee\_joint, joint)$ the latter is redundant although it does repair a missing is-a relation. The first preference relation that we define prefers to use structural repairs where every repairing action is needed within the structural repair. If a subset of a structural repair is also a structural repair, then the subset is preferred to its superset.

Definition 6. *Given an ontology network $\mathcal{N} = (\mathbb{O}, \mathbb{P})$, let $\mathcal{R}_N$ and $\mathcal{R}'_N$ be structural repairs for the networked ontologies $\mathbb{O}$ with respect to $\mathcal{M}_N$, then $\mathcal{R}_N$ is **axiom-preferred** to $\mathcal{R}'_N$ (notation $\mathcal{R}_N \ll_A \mathcal{R}'_N$) iff $\mathcal{R}_N \subseteq \mathcal{R}'_N$.*

The set of missing is-a relations is not always the most interesting structural repair for the domain expert. For instance, in the case of Figure 2, the structural repair $\{(limb\_joint_1, joint_1), (ankle\_joint_2, limb\_joint_2)\}$ is, for a domain expert, a more preferred way to repair the ontologies than the structural repair $\{(ankle\_joint_1, joint_1), (ankle\_joint_2, limb\_joint_2)\}$ which only contains the missing is-a relations. The former also repairs the ontologies, is correct according to the domain and is more informative. We define the notion of 'more informative than' for repairing actions as follows.

Definition 7. *Let (x_1, y_1) and (x_2, y_2) be two different repairing actions for 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 $\mathcal{O} \models x_2 \rightarrow x_1 \wedge y_1 \rightarrow y_2$.*

This definition implies that if (x_1, y_1) is more informative than (x_2, y_2), then adding (x_1, y_1) to the ontology will also allow us to derive (x_2, y_2) (and possibly more). Indeed, when (x_1, y_1) is added, then we know that in the extended ontology $x_2 \rightarrow x_1$ (from (x_1, y_1) is more informative than (x_2, y_2)), $x_1 \rightarrow y_1$ (added) and $y_1 \rightarrow y_2$ (from (x_1, y_1) is more informative than (x_2, y_2)), and thus $x_2 \rightarrow y_2$.

By using more informative repairing actions, we are able to add more (and sometimes within the network previously unknown) knowledge to our ontology. For instance, in the case of Figure 2, $(limb\_joint_1, joint_1)$ is more informative than $(ankle\_joint_1, joint_1)$, and by adding $limb\_joint_1 \rightarrow joint_1$, we have also introduced $ankle\_joint_1 \rightarrow joint_1$ as well as $knee\_joint_1 \rightarrow joint_1$ (which was also missing, but could not be derived from the network).

Another example is that $(hinderlimb\_joint_2, wrist\_joint_2)$ is more informative than $(ankle\_joint_2, limb\_joint_2)$. Indeed, adding $hinderlimb\_joint_2 \rightarrow wrist\_joint_2$[3]

[3]Note that, although this is a possible repairing action, it is not correct according to the domain. See also our comment immediately after Definition 8.

to the ontology will lead to the derivation of $ankle\_joint_2 \rightarrow limb\_joint_2$.

Our second preference relation is based on this notion.

Definition 8. *Given an ontology network $\mathcal{N} = (\mathbb{O}, \mathbb{P})$, let $\mathcal{R}_N$ and $\mathcal{R}'_N$ be structural repairs for the networked ontologies $\mathbb{O}$ with respect to $\mathcal{M}_N$. Then $\mathcal{R}_N$ is **information-preferred** to $\mathcal{R}'_N$ (notation $\mathcal{R}_N \ll_I \mathcal{R}'_N$) iff $\exists\, (x_1, y_1) \in \mathcal{R}_N$, $(x_2, y_2) \in \mathcal{R}'_N$: (x_1, y_1) is more informative than (x_2, y_2).*

We note, however, that the most preferred structural repairs according to $\ll_I$ are not necessarily correct according to the domain. For instance, $\{(ankle\_joint_1, joint_1)$, $(hinderlimb\_joint_2, wrist\_joint_2)\}$ is more preferred according to $\ll_I$ than $\{(ankle\_joint_1, joint_1)$, $(ankle\_joint_2, limb\_joint_2)\}$, but the former structural repair contains a repairing action that is not correct according to the domain (i.e. a hinderlimb joint is not a wrist joint). Therefore, in contrast to $\ll_A$ where minimality is desired, $\ll_I$ only gives a preference between different structural repairs, but the domain expert will need to decide on the correctness and essentially will choose the most preferred structural repairs among the correct ones.

Further, some structural repairs may introduce equivalence relations between concepts in some ontology which are only connected by an is-a relation in the original ontology. For example, in the case of Figure 2, the structural repair $\{(bone_1, joint_1)$, $(ankle\_joint_2, limb\_joint_2)\}$ will change the original is-a relation $(joint_1, bone_1)$ in ontology 1 into an equivalence relation. Although such a structural repair may result in a consistent ontology, this is usually not desired from a modeling perspective. The third preference relation prefers not to change is-a relations in an original ontology into equivalence relations.

Definition 9. *Given an ontology network $\mathcal{N} = (\mathbb{O}, \mathbb{P})$, let $\mathcal{R}_N = \cup_{k=1}^{n} \mathcal{R}_k$ and $\mathcal{R}'_N = \cup_{k=1}^{n} \mathcal{R}'_k$ be structural repairs for the networked ontologies $\mathbb{O} = \{\mathcal{O}_k\}_{k=1}^{n}$ with respect to $\mathcal{M}_N$, where for every k, $\mathcal{O}_k = (\mathcal{C}_k, \mathcal{I}_k)$. Then $\mathcal{R}_N$ is **strict-hierarchy-preferred** to $\mathcal{R}'_N$ (notation $\mathcal{R}_N \ll_{SH} \mathcal{R}'_N$) iff $\exists \mathcal{O}_i \in \mathbb{O}$ and $(a, b) \in \mathcal{I}_i : \mathcal{O}_i \not\models a \equiv b$ and $(\mathcal{C}_i, \mathcal{I}_i \cup \mathcal{R}_i) \not\models a \equiv b$ and $(\mathcal{C}_i, \mathcal{I}_i \cup \mathcal{R}'_i) \models a \equiv b$.*

We note that, according to our definitions, it is possible that one structural repair is preferred to a second structural repair, while at the same time the second structural repair is preferred to the first one. For example, in the case of Figure 2, let $\mathcal{R}_1$ be the structural repair $\{(limb\_joint_1, joint_1), (ankle\_joint_2, limb\_joint_2)\}$ and $\mathcal{R}_2$ be the structural repair $\{(ankle\_joint_1, joint_1), (hinderlimb\_joint_2, limb\_joint_2)\}$. Then $\mathcal{R}_1 \ll_I \mathcal{R}_2$ and $\mathcal{R}_2 \ll_I \mathcal{R}_1$. The first preference is based on the fact that $(limb\_joint_1$, $joint_1)$ is more informative than $(ankle\_joint_1, joint_1)$, while the second preference is based on the fact that $(hinderlimb\_joint_2, limb\_joint_2)$ is more informative than $(ankle\_joint_2, limb\_joint_2)$. In this case it is, however, possible to find a third structural repair, e.g $\{(limb\_joint_1, joint_1)$, $(hinderlimb\_joint_2, limb\_joint_2)\}$, that is strictly more information-preferred than both.

As explained in [14], $\ll_A$ is one way to capture Occam's razor. Another way to introduce a notion of simplicity of the solutions is the following heuristic of *single relations*. We assume that it is more likely that the ontology developers have missed to add single is-a relations, rather than a chain of is-a relations. For instance, it is

more likely that $(ankle\_joint_2, limb\_joint_2)$ is missing than $(ankle\_joint_2, x_1)$ and (x_1, x_2), and ... and $(x_k, limb\_joint_2)$.

3. Overview of the approach

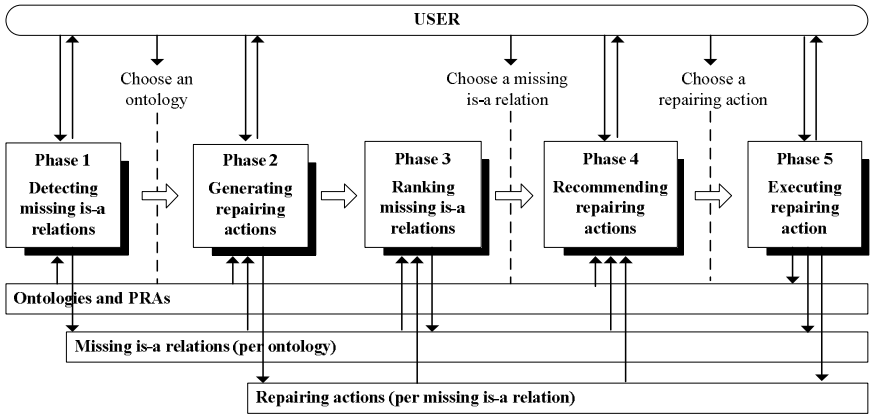

Figure 3: Approach for debugging missing is-a structure in ontologies networked by PRAs.

In this section, we give an overview of our debugging approach. As illustrated in Figure 3, the process consists of 5 phases (of which phases 3 and 4 are optional) and is driven by a domain expert. The input is a set of ontologies networked by a set of PRAs.

The user[4] starts with detecting missing is-a relations for all the networked ontologies **(Phase 1)**. The algorithm for detecting missing is-a relations is described in Section 4.

A naive way of repairing would be to compute all possible structural repairs for the networked ontologies. This is in practice infeasible as it involves all the ontologies and all the missing is-a relations 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 at once. Therefore, in our approach, we repair ontologies one at a time. After one ontology is chosen for repairing, we generate a set of possible repairing actions for each missing is-a relation in the chosen ontology **(Phase 2)** so that the user can repair the missing is-a relations one by one. The algorithm for generating possible repairing actions takes into account the preferences defined in Section 2.3. In general, there will be many missing is-a relations that need to be repaired and some of them may be easier to start with such as the ones with fewer repairing actions. Therefore,

[4]As for most ontology engineering tools, our aim is that a domain expert with ontology engineering expertise can use tools based on our approach without much introduction. Otherwise, an ontology engineer may assist a domain expert (or vice versa). The domain expert needs to make the final decisions on the repairing, while the ontology engineer may help with understanding is-a (e.g. as opposed to part-of) and understanding the consequences of the repairing. In work for the Swedish National Food Agency [26] the domain expert had some expertise in ontology engineering and few help from us was needed.

as an extra aid, we rank them with respect to the number of possible repairing actions **(Phase 3)**.

After this, the user can select a missing is-a relation to repair and choose between possible repairing actions. As an option, to facilitate this process, the user can ask for recommendations for repairing actions. We developed a method that recommends the most informative repairing actions supported by domain knowledge **(Phase 4)**. Once the user chooses a repairing action to execute, the chosen repairing action is then added to the ontology and the consequences are computed **(Phase 5)**. Some missing is-a relations may be repaired by the executed repairing action. Some missing is-a relations may have their repairing actions changed. Further, some new missing is-a relations may be found.

At any time during the process, the user can switch to another ontology or start earlier phases.

4. Detecting the missing is-a relations

The missing is-a relations derivable from the network could be found by checking the is-a relations between all concepts in every single ontology. If an is-a relation is not derivable from the ontology but derivable from the network, it is a missing is-a relation. However, some of these missing is-a relations are redundant in the sense that they can be repaired by the repairing of other missing is-a relations. It can be shown that only the missing is-a relations whose concepts appear in the mappings of the PRAs are necessary for repairing. (As a shorthand, we call the concepts appearing in the mappings of the PRAs *PRA concepts*.)

Proposition 1. *For each missing is-a relation in the network, there must exist a missing is-a relation whose concepts are PRA concepts, such that the repairing of the latter also repairs the former.*

PROOF. Suppose in an ontology network $\mathcal{N}$ as defined in Definition 3, there is a missing is-a relation (a, b) in an ontology $\mathcal{O}$. According to Definition 4, the relation $a \rightarrow b$ is not derivable from $\mathcal{O}$ 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 in $\mathcal{O}$ and a concept in another ontology in the network, for instance $x \rightarrow x'$ (or $x \equiv x'$), satisfying $\mathcal{O}_N \models a \rightarrow x \rightarrow x' \rightarrow z$, where x is a PRA concept in ontology $\mathcal{O}$. Likewise, for concepts z and b, the relation $z \rightarrow b$ must also be supported by a mapping between a concept in $\mathcal{O}$ and a concept in another ontology in the network, for instance $y' \rightarrow y$ (or $y' \equiv y$), satisfying $\mathcal{O}_N \models z \rightarrow y' \rightarrow y \rightarrow b$, where y is a PRA concept in ontology $\mathcal{O}$. 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 $\mathcal{O}$, the relation $x \rightarrow y$ can not be inferred from $\mathcal{O}$ either. This means that (x, y) is also a missing is-a relation in the network, and the repairing of missing is-a relation (x, y) also repairs (a, b). ♣

Based on Proposition 1, repairing the missing is-a relations between PRA concepts also repairs all other missing is-a relations derivable from the ontology network. Therefore, our algorithm in Figure 4 considers only missing is-a relations between PRA concepts.

Input
The ontology network $\mathcal{N} = (\mathbb{O}, \mathbb{P})$, the induced ontology $\mathcal{O}_N$,
the set of missing is-a relations $\mathcal{M}_N$, the set of is-a relations to check $\mathcal{M}_N^\star$.
Output
The updated set of missing is-a relations $\mathcal{M}_N$,
the updated set of is-a relations to check $\mathcal{M}_N^\star$.
Algorithm
For each ontology $\mathcal{O} \in \mathbb{O}$ where $\mathcal{O} = (\mathcal{C}, \mathcal{I})$:
    For every $(a, b) \in \mathcal{M}_N^\star$ and $a, b \in \mathcal{C}$:
        If $\mathcal{O} \models a \rightarrow b$ then remove (a, b) from $\mathcal{M}_N^\star$;
        If $\mathcal{O} \not\models a \rightarrow b$ and $\mathcal{O}_N \models a \rightarrow b$ then:
            Add (a, b) as a missing is-a relation to $\mathcal{M}_N$;
            Remove (a, b) from $\mathcal{M}_N^\star$.

Figure 4: Algorithm for detecting missing is-a relations.

In the algorithm in Figure 4, the global variable $\mathcal{M}_N$ represents the set of missing is-a relations. Before the algorithm is run for the first time, $\mathcal{M}_N$ is initialized to be the empty set. The global variable $\mathcal{M}_N^\star$ represents the set of is-relations which we need to check to find missing is-a relations in the networked ontologies. Before the algorithm is run for the first time, $\mathcal{M}_N^\star$ is initialized to be the set of pairs (a, b) where a and b are PRA concepts in the same ontology. For each element (a, b) in $\mathcal{M}_N^\star$ we then check whether $a \rightarrow b$ can be derived in the ontology to which a and b belong. If so, then this is not a missing is-a relation and (a, b) is removed from $\mathcal{M}_N^\star$. Otherwise, we check whether $a \rightarrow b$ can be derived from the network. If so, then it is a missing is-a relation and we add (a, b) to $\mathcal{M}_N$, and remove it from $\mathcal{M}_N^\star$. Otherwise, $a \rightarrow b$ can neither be derived from the ontology nor from the network. It then remains in $\mathcal{M}_N^\star$ as it may become derivable later when we have repaired part of the network. As all pairs of PRA concepts are checked, our algorithm ensures that all missing is-a relations between PRA concepts that can be derived from the current network will be found.

After the missing is-a relations are found, they will be repaired in later phases of the debugging process and this will bring changes to the is-a structures of the repaired ontologies and the induced ontology. Therefore, it is possible that some new is-a relations become derivable from the network and thus generate new missing is-a relations. For example, in the case of Figure 2, suppose we repair $(ankle\_joint_1, joint_1)$ by adding the is-a relation $(limb\_joint_1, joint_1)$ in ontology 1. Then, when re-running the detection algorithm, we find a new missing is-a relation $(limb\_joint_2, joint_2)$, since $(limb\_joint_2, joint_2)$ has now become inferrable from the induced ontology and it is still not inferrable from ontology 2. Therefore, after executing repairing actions, we need to re-run the detection algorithm to find new missing is-a relations. The initial

value of $\mathcal{M}_N^\star$ when re-running the detection algorithm is the same as the final value of $\mathcal{M}_N^\star$ in the previous run. This is because we do not change the mappings between the ontologies when repairing. Therefore, the set of PRA concepts does not change and thus there are no additions to $\mathcal{M}_N^\star$.

5. Repairing the missing is-a relations

As explained in Section 3, our approach deals with the networked ontologies one at a time. For the ontology under repair, possible repairing actions are generated for all its missing is-a relations. After ranking these missing is-a relations, the user can select a missing is-a relation to repair, and we provide an algorithm that recommends repairing actions. Further, we developed an algorithm that, upon the repairing of a missing is-a relation, detects for which missing is-a relations the set of repairing actions needs to be updated, and updates these.

5.1. Generating repairing actions

5.1.1. Basic algorithm

In our basic algorithm (see Figure 5), when generating repairing actions for a missing is-a relation, we take into consideration that all missing is-a relations will be repaired (least informative repairing action), but we do not take into account the consequences of the actual (possibly more informative) repairing actions that will be performed for other missing is-a relations.

Input
The ontology under repair $\mathcal{O}$, its set of missing is-a relations $\mathcal{M}$.
Output
Repairing actions.
Algorithm
1. Initialize KB with ontology;
2. For every missing is-a relation $(a, b) \in \mathcal{M}$: add the axiom $a \rightarrow b$ to the KB;
3. For each $(a, b) \in \mathcal{M}$:
   $Source(a, b)$:= super-concepts(a) $-$ super-concepts(b);
   $Target(a, b)$:= sub-concepts(b) $-$ sub-concepts(a);
4. Missing is-a relation (a, b) can be repaired by choosing an element from $Source(a, b) \times Target(a, b)$.

Figure 5: Basic algorithm for generating repairing actions.

In this algorithm, we store the ontology in a knowledge base and add the missing is-a relations. As we know that these missing is-a relations are derivable from the network, adding them will introduce the desired connections. It guarantees that, for the ontology under repair, all inferrable is-a relations between its concepts in the network will also become inferrable from the ontology. Essentially, this conforms to a structural repair containing the least informative repairing actions for each of the missing is-a relations in the ontology. Then, for each missing is-a relation, we generate its possible

repairing actions by computing two sets of concepts, called Source and Target sets. A possible repairing action regarding missing is-a relation (a, b) is an is-a relation (s, t) where s is an element from its Source set and t is an element from its Target set.

The algorithm computes repairing actions for each of the missing is-a relations. Structural repairs can be constructed by selecting one repairing action per missing is-a relation (and this conforms to the single relation heuristic in Section 2.3). The computation of Source and Target sets ensures that an element from $Source(a, b) \times Target(a, b)$ repairs missing is-a relation (a, b).[5] Therefore, a set consisting of one element from $Source(a, b) \times Target(a, b)$ for each missing is-a relation (a, b) is a structural repair. The algorithm terminates as we assume that there are a finite number of concepts in the ontologies and the computation of the Source and Target sets is based on computing super-concepts and sub-concepts in a finite taxonomy.

Further, every repairing action in such a structural repair repairs at least one missing is-a relation (preference $\ll_A$ in Definition 6). We note, however, that these may not always be the most preferred structural repairs according to $\ll_A$. It may happen that a repairing action selected for a missing is-a relation also repairs several other missing is-a relations. Therefore, the repairing actions related to these other missing is-a relations are redundant and when these repairing actions are removed from the structural repair, we have a structural repair that is more preferred according to $\ll_A$ than the original structural repair. The algorithm could easily be adapted to construct these most preferred structural repairs by for every structural repair generated by the algorithm, checking whether the subsets are still structural repairs and taking the minimal subsets. This is an expensive step and in our implemented system, we have not implemented this. However, as we repair one missing is-a relation at the time in our implemented system (Section 6), this is not a problem in practice. When a repairing action repairs several missing is-a relations, then all repaired missing is-a relations will be marked as repaired and will not be considered further.

Every possible repairing action (s, t) computed by the algorithm satisfies $a \rightarrow s$ and $t \rightarrow b$. This means that for every missing is-a relation (a, b) the selected repairing action from $Source(a, b) \times Target(a, b)$ in the structural repair is (a, b) itself or a repairing action that is more informative than (a, b). Thus, the generated structural repairs are the set of missing is-a relations itself as well as structural repairs that are more preferred according to $\ll_I$ (Definition 8) than the set of missing is-a relations. We note that the algorithm does not only compute the most preferred structural repairs according to $\ll_I$. As explained in Section 2.3, although in general, we prefer more informative repairing actions, these should still be validated by a domain expert. When a domain expert rejects a more informative repairing action for a missing is-a relation, a less informative will still repair the missing is-a relation. For instance, the least informative repairing action that will repair a missing is-a relation is the missing is-a relation itself.

Further, it is guaranteed that for missing is-relation (a, b) repairing actions of the form (a, t) or (s, b) do not introduce new equivalence relations, where in the source

[5] Observe that we consider that all missing is-a relations will be repaired, and it is under this consideration that it is guaranteed that the repairing action repairs the missing is-a relation.

ontology[6] we have only is-a relations (preference $\ll_{SH}$ in Definition 9). Let us prove this for repairing actions of the form (a, t). Assume a new equivalence relation has been introduced by (a, t) where previously there was only an is-a relation. This means that there exist a u and v such that $u \rightarrow v$ in the source ontology and after adding $a \rightarrow t$, we now also have $v \rightarrow u$. As adding $a \rightarrow t$ leads to $v \rightarrow u$, we have that $v \rightarrow a$ and $t \rightarrow u$ in the source ontology. This would mean that in the source ontology $t \rightarrow u$, $u \rightarrow v$ and $v \rightarrow a$, and thus $t \rightarrow a$. However, this would make t a sub-concept of a in the source ontology and thus, according to the algorithm, t could not have been in the Target set for (a, b). This yields a contradiction and thus (a, t) does not introduce new equivalence relations. Similar reasoning leads to the fact that repairing actions of the form (s, b) do not introduce new equivalence relations. We note, however, that a choice of repairing action (s, t) where $t \rightarrow s$ in the source ontology, will lead to the introduction of equivalence relations. It is easy to adapt the algorithm in step 4 to check this for each chosen repairing action. In the implemented system (Section 6) we have not implemented this to make the visualization of *all* generated repairing actions for a missing is-a relation at the same time (using Source and Target sets) as simple as possible. However, when a user selects a repairing action we check whether an equivalence is introduced and in such case a notification is given to the user.

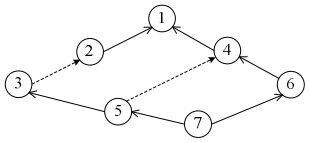

Figure 6: Generating repairing actions - Example 1 - Ontology and missing is-a relations.

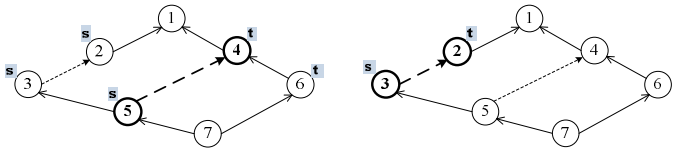

Figure 7: Generating repairing actions - Example 1 - Source and Target sets for (5,4) and (3.2).

As an example, consider the case presented in Figure 6, where $\mathcal{O} = (\mathcal{C}, \mathcal{I})$ is an ontology with concepts $\mathcal{C} = \{1, 2, 3, 4, 5, 6, 7\}$ and is-a relations (shown in full lines in Figure 6) $\mathcal{I} = \{(7, 5), (7, 6), (5, 3), (2, 1), (6, 4), (4, 1)\}$. ($\mathcal{I}$ represents the is-a hierarchy and thus also all is-a relations derived from the elements in $\mathcal{I}$.) The set of missing is-a relations (shown in dashed lines in Figure 6) is $\mathcal{M} = \{(5, 4), (3, 2)\}$. The algorithm then generates the following Source and Target sets.

$Source(5, 4) = \{5, 3, 2, 1, 4\} - \{4, 1\} = \{5, 3, 2\}$

[6] with missing is-a relations added.

$Target(5,4) = \{4,6,7,5\} - \{5,7\} = \{4,6\}$
$Source(3,2) = \{3,2,1\} - \{2,1\} = \{3\}$
$Target(3,2) = \{2,3,5,7\} - \{3,5,7\} = \{2\}$

These sets are visualized in Figure 7. On the left hand side of Figure 7 the Source and Target sets are shown for missing is-a relation (5,4) and on the right hand side we have the Source and Target sets for the missing is-a relation (3,2). The missing is-a relation under consideration is highlighted in bold. Elements in a Source set are annotated with 's', while elements in a Target set are annotated with 't'. For missing is-a relation $(3,2)$ the only generated repairing action is $(3,2)$. For missing is-a relation $(5,4)$ any of the repairing actions $(5,4)$, $(5,6)$, $(3,4)$, $(3,6)$, $(2,4)$, $(2,6)$ together with (any of) the generated repairing action(s) for $(3,2)$ leads to the derivation of the missing is-a relation $(5,4)$ in the extended ontology. The example also shows the importance of initially adding the missing is-a relations to the knowledge base. The possible repairing action $(2,4)$ for missing is-a relation $(5,4)$ would not be generated when we do not take into account that missing is-a relation $(3,2)$ will be repaired.[7] Further, the example also shows that we do not introduce repairing actions that would turn is-a relations in the original ontology into equivalence relations. For instance, adding $(1,4)$ would lead to the fact that missing is-a relation $(5,4)$ would be derivable in the extended ontology, but also leads to making 1 and 4 equivalent.

Input
The ontology under repair $\mathcal{O}$, its set of missing is-a relations $\mathcal{M}$.
Output
Repairing actions.
Algorithm
1. Initialize KB with ontology ;
2. For every missing is-a relation $(a,b) \in \mathcal{M}$:
   Create two new concepts x and y in the KB;
   Add the axioms $a \rightarrow x$, $x \rightarrow y$, $y \rightarrow b$ to the KB;
3. For each $(a,b) \in \mathcal{M}$:
   $Source\text{-}ext(a,b)$:= super-concepts(a) $-$ super-concepts(x);
   $Target\text{-}ext(a,b)$:= sub-concepts(b) $-$ sub-concepts(y);
4. Missing is-a relation (a,b) can be repaired by choosing an original ontology element from $Source\text{-}ext(a,b)$ and an original ontology element from $Target\text{-}ext(a,b)$.

Figure 8: Extended algorithm for generating repairing actions.

[7] So this means that repairing one is-a relation may influence the repairing actions for other missing is-a relations. However, when *generating* repairing actions in the algorithm in Figure 5 the only influence that is taken into consideration is the fact that missing is-a relations are or will be repaired (least informative repairing action), but not the actual (possibly more informative) repairing actions that will be performed.

5.1.2. Extended algorithm

Our extended algorithm (see Figure 8) for finding repairing actions for a particular missing is-a relation takes into account influences of other missing is-a relations that are valid for all possible choices for repairing actions for the other missing is-a relations. Before computing the Source and Target sets, we introduce two new concepts x and y for each missing is-a relation (a, b) in the knowledge base as well as the axioms $a \rightarrow x$, $x \rightarrow y$, $y \rightarrow b$. (x, y) satisfies the requirements that each possible repairing action for (a, b) should satisfy. As they are new concepts in the knowledge base, the properties and relations of x, respectively y, to other concepts in the knowledge base represent the properties and relations that are common to the Source concepts, respectively Target concepts, of the possible repairing actions for (a, b). The Source and Target sets are now computed relative to the x and y.

Consider the case presented in Figure 9, where $\mathcal{O} = (\mathcal{C}, \mathcal{I})$ is an ontology with concepts $\mathcal{C} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and is-a relations (shown in full lines in Figure 9) $\mathcal{I} = \{(7, 6), (6, 5), (5, 2), (2, 1), (7, 4), (10, 4), (10, 9), (9, 8), (8, 3), (3, 1), (4, 1)\}$. (As before, $\mathcal{I}$ represents the is-a hierarchy and thus also all is-a relations derived from the elements in $\mathcal{I}$.) The set of missing is-a relations (shown in dashed lines in Figure 9) is $\mathcal{M} = \{(5, 4), (8, 4)\}$.

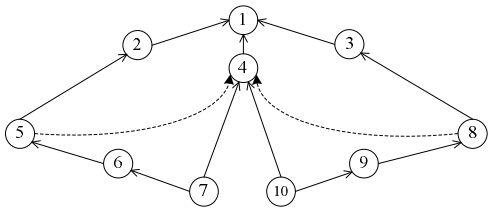

Figure 9: Generating repairing actions - Example 2 - Ontology and missing is-a relations.

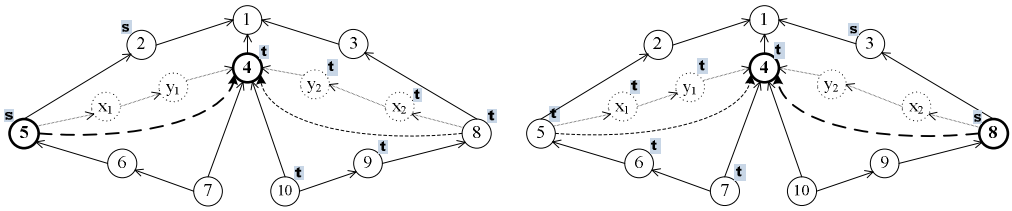

Figure 10: Generating repairing actions - Example 2 - Source and Target sets for (5,4) and (8,4).

The basic algorithm in Figure 5 generates the following Source and Target sets.
$Source(5, 4) = \{5, 4, 1, 2\} - \{4, 1\} = \{5, 2\}$
$Target(5, 4) = \{4, 8, 9, 10, 5, 6, 7\} - \{5, 6, 7\} = \{4, 8, 9, 10\}$
$Source(8, 4) = \{8, 4, 1, 3\} - \{4, 1\} = \{8, 3\}$
$Target(8, 4) = \{4, 8, 9, 10, 5, 6, 7\} - \{8, 9, 10\} = \{4, 5, 6, 7\}$
The extended algorithm in Figure 8 adds the concepts x_1, y_1, x_2, y_2 and the is-a rela-

tions $5 \rightarrow x_1, x_1 \rightarrow y_1, y_1 \rightarrow 4, 8 \rightarrow x_2, x_2 \rightarrow y_2$ and $y_2 \rightarrow 4$ (shown in dotted lines in Figure 9) in the knowledge base. It then generates the following Source and Target sets (Figure 10).

$$\begin{aligned}
Source\text{-}ext(5,4) &= \{5,4,1,2,x_1,y_1\} - \{4,1,x_1,y_1\} = \{5,2\} \\
Target\text{-}ext(5,4) &= \{4,8,9,10,5,6,7,x_1,y_1,x_2,y_2\} - \{5,6,7,x_1,y_1\} \\
&= \{4,8,9,10,x_2,y_2\} \\
Source\text{-}ext(8,4) &= \{8,4,1,3,x_2,y_2\} - \{4,1,x_2,y_2\} = \{8,3\} \\
Target\text{-}ext(8,4) &= \{4,8,9,10,5,6,7,x_1,y_1,x_2,y_2\} - \{8,9,10,x_2,y_2\} \\
&= \{4,5,6,7,x_1,y_1\}
\end{aligned}$$

The sets generated by the extended algorithm indicate that there is an influence between the two missing is-a relations. Indeed, when a choice is made for repairing the first missing is-a relation, we have essentially added equivalence relations between x_1, respectively y_1, and concepts in the ontology. The appearance of x_1 and y_1 in the Target-ext set for the second missing is-a relation indicates that the concept chosen to be equivalent to x_1 (and all concepts between this concept and 5) are now also candidates for the Target-ext for the second missing is-a relation. For example, when choosing $(2,4)$ as a repairing action for missing is-a relation $(5,4)$ then $(3,2)$ is a possible repairing action for missing is-a relation $(8,4)$.

Similarly to the basic algorithm, the proposed repairing actions for a missing is-a relation (a,b) all lead to the derivation of (a,b) in the extended ontology. In general, a user may repair the ontology by choosing for each missing is-a relation (a,b) an original ontology element from $Source\text{-}ext(a,b)$ and an original ontology element from $Target\text{-}ext(a,b)$. However, as the algorithm only takes into account influences that are common to all possible choices for repairing actions, a user may want to repair one missing is-a relation and recompute repairing actions for the other missing is-a relations.

5.2. Ranking repairing actions

In general, there may be many missing is-a relations that need to be repaired. Although it is possible to repair the missing is-a relations in any order, some orders may be more important or make it easier for the user. For instance, it may be important to first repair is-a structure in the top level of the ontology, or it may be easiest to deal with the missing is-a relations with the fewest repairing choices. In this paper we use a ranking algorithm that allows the user to start with the is-a relations where there are the fewest choices. These are usually easiest to visualize and resolve. Therefore, our ranking algorithm ranks the missing is-a relations according to the number of their possible repairing actions. For a missing is-a relation, this is calculated as the product of the Source set size and Target set size for the basic algorithm. For the extended algorithm, it is calculated in the same manner but without counting the extra-added new concepts. The user can choose to use the ranking or ignore it.

5.3. Recommending repairing actions

For a missing is-a relation under repair, there may be many possible repairing actions to choose from. Therefore, as an option, the user can ask for recommendations for repairing actions. We developed an algorithm that recommends the most informative repairing actions (see Definition 7) that are supported by some external domain

Input:
The ontology $\mathcal{O}$ and the missing is-a relation (a, b) under repair,
$Source(a, b)$ and $Target(a, b)$ computed by the basic algorithm.
Output
Recommended repairing actions.
Algorithm
Global Variable $visited$: stores already processed repairing actions.
Global Variable $recommended$: stores recommended repairing actions.
1. Set $visited$ = $\{(a, b)\}$;
2. Set $recommended$ = $\{(a, b)\}$;
3. Set X_e = $\{x_e : x_e \in Source(a, b) \bigwedge \forall x \in Source(a, b)$: if $x_e \rightarrow x$ then $x = x_e\}$;
4. Set Y_e = $\{y_e : y_e \in Target(a, b) \bigwedge \forall\ y \in Target(a, b)$: if $y \rightarrow y_e$ then $y = y_e\}$;
5. For each pair $(x_e, y_e) \in X_e \times Y_e$: call `FindRec`$(x_e, y_e)$;
6. Return $recommended$;
Function `FindRec(concept x, concept y)`
i. If $(x, y) \in visited$ then return;
ii. Add (x, y) to $visited$;
iii. If $\exists\ (x_r, y_r) \in recommended$: $x \rightarrow x_r \wedge y_r \rightarrow y$ then return;
iv. If x is a sub-concept of y according to the domain knowledge, then
    Remove all (x_r, y_r) from $recommended$ for which $x_r \rightarrow x$ and $y \rightarrow y_r$;
    Add (x, y) to $recommended$;
  else
    Let Y_{sup} be the set of direct super-concepts of y;
    For each $y_s \in Y_{sup} \cap Target(a, b)$: call `FindRec`$(x, y_s)$;
    Let X_{sub} be the set of direct sub-concepts of x;
    For each $x_s \in X_{sub} \cap Source(a, b)$: call `FindRec`$(x_s, y)$;

Figure 11: Algorithm for recommending repairing actions.

knowledge. We assume that there is domain knowledge which we can query regarding subsumption between concepts. There are several such sources such as general thesauri (e.g. WordNet) or specialized domain-specific sources (e.g. the Unified Medical Language System).

Essentially, the recommendation algorithm selects, among the structural repairs computed by the algorithm that generates repairing actions (Section 5.1), the structural repairs that contain the most informative repairing actions (preference $\ll_I$ in Definition 8) that are also supported by domain knowledge. The user can choose to accept the recommendations or not.

In our algorithm (see Figure 11) we generate recommended repairing actions for a missing is-a relation starting from the Source and Target sets generated by the algorithm in Figure 5[8]. The algorithm selects the most informative repairing actions that are

[8]We have also extended the algorithm in Figure 11 to deal with Source-ext and Target-ext sets derived by the algorithm in Figure 8.

supported by evidence in the domain knowledge. The variable $visited$ keeps track of already processed repairing actions. The variable $recommended$ stores recommended repairing actions at each step and its final value is returned as output. It is initialized with the repairing action from the missing is-a relation itself. This is the least informative repairing action which is ensured to be correct. Steps 3 and 4 compute the set X_e of maximal elements with respect to the is-a relation in the Source set and the set Y_e of minimal elements with respect to the is-a relation in the Target set. The elements from $X_e \times Y_e$ are then the most informative repairing actions. For each of these elements (x, y) we check whether there is support in the domain knowledge in step 5. Steps i and ii in the function `FindRec` do bookkeeping regarding the already processed repairing actions. Step iii assures that we do not recommend is-a relations that are less informative than others already recommended. In step iv we check whether there is support in the domain knowledge for the repairing action. If so, then the repairing action is recommended and all less informative repairing actions are removed from the recommendation set. If not, then we check whether there is support in the domain knowledge for the repairing actions that are less informative than (x, y). Among these we start with the most informative repairing actions.

5.4. Executing repairing actions

For a missing is-a relation under repair, the user selects from the generated repairing actions, possibly based on a recommendation, a repairing action to repair the missing is-a relation. We note that, whenever the user selects a more informative repairing action than the missing is-a relation itself, we have actually also detected a missing is-a relation (i.e. the more informative repairing action) that could not be derived from the ontology network. When a user executes a repairing action for a particular missing is-a relation, it may influence the set of possible repairing actions for other missing is-a relations. Therefore, the repairing actions for the other missing is-a relations need to be recomputed based on the ontology extended with the chosen repairing action.

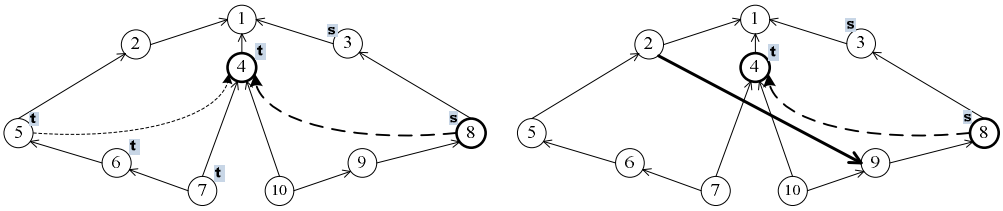

Figure 12: Updating repairing actions - Example 2 - Ontology, missing is-a relations and Source and Target sets for (8,4); before and after repairing missing is-a relation (5,4) using repairing action (2,9).

For instance, Figure 12 shows on the left hand side the original situation of the example in Figure 9, and on the right hand side the new situation after having repaired missing is-a relation $(5, 4)$ using repairing action $(2, 9)$ (shown in thick line). In this case the Source and Target sets for missing is-a relation $(8, 4)$ become the following for the basic algorithm:

$Source(8, 4) = \{8, 4, 1, 3\} - \{4, 1\} = \{8, 3\}$

$Target(8,4) = \{4,8,9,10,2,5,6,7\} - \{8,9,10,2,5,6,7\} = \{4\}$
and the following for the extended algorithm:
$Source\text{-}ext(8,4) = \{8,4,1,3,x_2,y_2\} - \{4,1,x_2,y_2\} = \{8,3\}$
$Target\text{-}ext(8,4) = \{4,8,9,10,2,5,6,7,x_2,y_2\} - \{8,9,10,2,5,6,7,x_2,y_2\}$
$= \{4\}$
When we compare the computed repairing actions after the choice of $(2,9)$ for repairing $(5,4)$ with the repairing actions computed before the choice (see Section 5.1), we note that the repairing actions that introduce equivalence relations (e.g. $(3,5)$, $(3,6)$, $(3,7)$, $(8,5)$, $(8,6)$ and $(8,7)$ for the basic algorithm) are removed after the choice of $(2,9)$ (preference $\ll_{SH}$ in Definition 9). However, before $(2,9)$ is chosen these repairing actions do not necessarily introduce equivalence relations. For instance, we could have repaired $(8,4)$ first using one of these repairing actions, and afterwards repaired $(5,4)$.

Input
The ontology under repair $\mathcal{O}$, the repaired missing is-a relation (a_r, b_r), the repairing action (x_r, y_r) taken for (a_r, b_r), the set of non-repaired missing relations $\mathcal{M}_r$.
Output
Updated Source and Target sets.
Algorithm
1. Add (x_r, y_r) to the KB;
2. For each missing is-a relation $(a,b) \in \mathcal{M}_r$:
If $a \rightarrow x_r$ then recompute super-concepts(a);
If $b \rightarrow x_r$ then recompute super-concepts(b);
If $a \rightarrow x_r$ or $b \rightarrow x_r$ then $Source(a,b)$:= super-concepts(a) $-$ super-concepts(b);
If $y_r \rightarrow a$ then recompute sub-concepts(a);
If $y_r \rightarrow b$ then recompute sub-concepts(b);
If $y_r \rightarrow a$ or $y_r \rightarrow b$ then $Target(a,b)$:= sub-concepts(b) $-$ sub-concepts(a);

Figure 13: Algorithm for updating Source and Target sets.

For small ontologies, computing the repairing actions does not take much time and the approach is feasible in a real setting. For large ontologies the computation time may not be small enough to guarantee immediate updates in an implemented tool for repairing. Therefore, in the algorithm[9] in Figure 13 we introduced a way to keep track of the influences between different missing is-a relations. The missing is-a relations for which the Source or Target sets can change are the missing is-a relations for which at least one of the concepts is a sub-concept or super-concept of at least one of the concepts in the chosen repairing action for the repaired missing is-a relation. We only update the Source and Target sets for these missing is-a relations. In addition, we also remove the other missing is-a relations that have been repaired by the current repairing action. This is essentially updating the global variable $\mathcal{M}_N$ as described in

[9]Like the algorithm in Figure 11, this algorithm is applicable for cases using the basic algorithm. We also have a version with similar strategy for when we use the extended algorithm.

the detection algorithm in Figure 4.

6. Implemented System

We implemented a system RepOSE (*Rep*air of *O*ntological *S*tructure *E*nvironment) in Java based on our approach described in the previous section. We use a framework and reasoner provided by Jena (version 2.5.7) [28]. Here, we show its use using pieces of MA and NCI-A regarding the concept *joint*, as well as a PRA with 8 equivalence mappings.

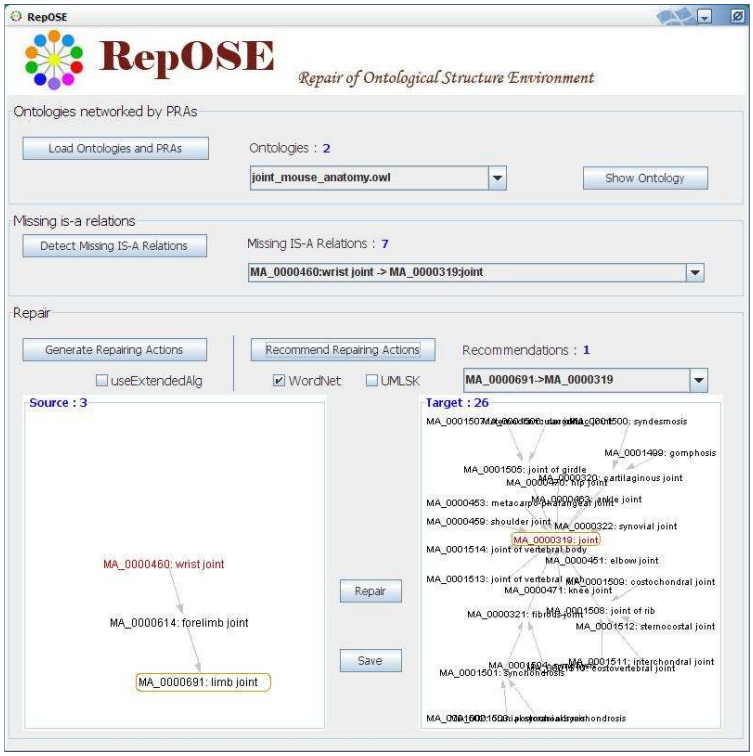

Figure 14: User interface of RepOSE.

As input our system takes a set of ontologies in OWL format as well as a set of PRAs in RDF format. The ontologies and PRAs can be imported using the *Load Ontologies and PRAs* button. The user can see the list of ontologies in the *Ontologies* menu (see Figure 14). Once the *Detect Missing IS-A Relations* button is clicked, missing is-a relations are detected in all ontologies. Then, the user can select an ontology to repair, and the *Missing IS-A Relations* menu shows the missing is-a relations of the currently selected ontology. In this case the ontology *joint_mouse_anatomy.owl* is selected and it contains 7 missing is-a relations (same as the case in Figure 1).

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, which is preceded by a two-stage preprocessing step. During the preprocessing, one stage is to identify and repair the missing is-a relations which are actually equivalence relations. This refers to cases where in the original ontology there is an is-a relation $a \rightarrow b$, and according to the network $b \rightarrow a$ is missing, as well as cases where both $a \rightarrow b$ and $b \rightarrow a$ are missing according to the network. As the structure in the ontologies and the mappings are assumed to be correct, this means that a and b are equivalent. These missing is-a relations are therefore immediately repaired by adding them to the ontology and essentially introducing an equivalence between the concepts. The other stage is to identify and remove the redundant missing is-a relations which are derivable from the ontology extended with other missing is-a relations. After preprocessing, repairing actions for each missing is-a relation are computed and presented as Source and Target sets. The selection of the *useExtendedAlg* checkbox makes the computation use our extended algorithm, otherwise our basic algorithm is used.

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, the Source and Target sets for the repairing actions are shown in the panels on the left and the right, respectively (as shown in Figure 14). Both these panels have zoom control and could be opened in a separate window by double clicking. The concepts in the missing is-a relation are highlighted in red. In this case, the repairing actions of the missing is-a relations are generated using the basic algorithm. The selection of the missing is-a relation "*wrist joint* is-a *joint*" displays its Source and Target sets in the panels. They contain 3 and 26 concepts respectively.

For the selected missing is-a relation, the user can also ask for recommended repairing actions by clicking the *Recommend Repairing Actions* button. For the query of domain knowledge, we currently implemented two methods. The first method is based on WordNet, making use of the WordNet senses and hypernym relations to verify the subsumption relation between concepts. The second one is based on UMLS Knowledge Source Server, checking whether one concept is defined as an ancestor of another in UMLS Metathesaurus. On the interface, the two checkboxes allow the user to specify the external domain knowledge used for generating recommendations. In our case, the system uses *WordNet* and recommends to add an is-a relation between *limb joint* and *joint*. In general, the system presents a list of recommendations. By selecting an element in the list, the concepts in the recommended repairing action are identified by round boxes in the panels. 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. The repairing action is then added to the ontology, and other missing is-a relations are updated, as well as the set of missing is-a relations of every ontology in the network.

At all times during the process the user can inspect the ontology under repair by clicking the *Show Ontology* button. The is-a structure of the repaired ontology will be shown in a separate window with newly added is-a relations being highlighted. The user can save the repaired ontology into an OWL file by clicking the *Save* button,

or select another ontology to repair. At all times the user can also switch to another ontology in the network. The whole debugging process runs semi-automatically until no more missing is-a relations are found or unrepaired in the network.

7. Experiments

In this section, we present experiments using our debugging approach. We use different cases for which we run complete debugging sessions. In the 'OAEI Anatomy' experiments we debug a network in the biomedical domain consisting of the two anatomy ontologies and a PRA in the OAEI Anatomy track. We use both the basic and extended algorithms for the generation of repairing actions. In the 'OAEI Bibliography' experiments we use 5 ontologies of the OAEI Benchmark track with four PRAs. We run debugging sessions on the full network of 5 ontologies and 4 PRAs, as well as on subnetworks consisting of 2 ontologies and 1 PRA. In the 'ONKI' experiment we debug two ontologies from the Finnish Ontology Library Service ONKI, a maritime ontology and an ontology which is an integrated ontology based on several core ontologies and domain extensions, and a PRA.

The experiments for OAEI Anatomy and OAEI Bibliography were performed on an AMD Dual Core Processor 2.90GHZ desktop machine with 4 GB DDR2 memory under Windows Vista Business operating system (SP2) and Java 1.6 compiler. The ONKI experiment was run on an Intel Xeon Processor 3.46Ghz server with 12 GB DDR3 memory under Debian 6.0.6 and Java 1.6 compiler.

7.1. OAEI Anatomy

In the OAEI Anatomy experiments we debug a network consisting of the two ontologies and the PRA from the 2008 Anatomy track in OAEI. As described before, the two ontologies, MA and NCI-A, contain 2744 and 3304 concepts respectively, while the PRA between them contains 988 equivalence mappings (see Table 1). Our debugging leads to the detection of 205 missing is-a relations in MA and 177 in NCI-A. These were repaired by the addition of 101 is-a relations in MA and 87 in NCI-A.

	number of concepts	PRA - total mappings	PRA - equivalence mappings	PRA - subsumption mappings
MA	2744	-	-	-
NCI-A	3304	-	-	-
	-	988	988	0

Table 1: Anatomy ontologies network.

The test runs for this experiment were done by the authors. As we are not domain experts for this experiment, we have used [15] to decide on possible choices and used the recommendation algorithm.

7.1.1. Run with basic algorithm for generation of repairing actions

The total debugging session took about 3 hours. This includes the computation by RepOSE, the user interaction with RepOSE as well as the looking up of domain knowledge by the user.

As a first step in the session the two ontologies and the PRA were loaded. Then the detection algorithm was run. This took about 2 minutes. As a result we found 199 missing is-a relations in MA and 167 in NCI-A. This is shown in Table 2 where the column 'total' shows the total number of detected missing is-a relations during the whole debugging session and in parentheses the number of initially detected missing is-a relations. As discussed before, all these missing is-a relations are between PRA concepts. A further check shows that some missing is-a relations are actually equivalence relations (column 'equivalence'), and some are redundant (column 'redundant). When all these missing is-a relations are preprocessed before generating repairing actions, we have 115 missing is-a relations in MA and 80 in NCI-A (column 'to repair').

	total	equivalence	redundant	to repair
MA	205 (199)	6 (6)	79 (78)	120 (115)
NCI-A	177 (167)	3 (3)	87 (84)	87 (80)

Table 2: Anatomy - Missing is-a relations detected during the whole debugging session. In parentheses the missing is-a relations that are initially detected.

The next step is to generate repairing actions for the remaining missing is-a relations in all ontologies. For MA, our basic algorithm generates for 9 missing is-a relations only 1 repairing action (which is then the missing is-a relation itself). This means that these could be immediately repaired. For NCI-A this number is 5. Of the remaining missing is-a relations there are 61 missing is-a relations for MA that have only 1 element in the Source set and 2 missing is-relations that have 1 element in the Target set. For NCI-A these numbers are 20 and 3, respectively. These are likely to be good starting points for repairing. Tables 3 and 4 show for different ranges how many Source and Targets sets had a size in that range. For most of the missing is-a relations these sets are small and thus can be easily visualized in the panels of our system.

	total	1	2-10	11-20	21-30	31-40	41-50	51-100
MA - Source	115	70	45	0	0	0	0	0
MA - Target	115	11	50	5	9	4	6	5
NCI-A - Source	80	25	55	0	0	0	0	0
NCI-A - Target	80	8	52	6	2	0	0	5

Table 3: Anatomy - Sizes of Source and Target sets when generating repairing actions for the first time - part 1.

Table 5 shows the results of the repairing. Most of the missing is-a relations were repaired explicitly by the user through interaction with RepOSE (column 'explicitly repaired'), while some in MA were repaired as a result of the repairing of others (column 'repaired by others'). In some cases it was immediately clear which repairing

	total	101-200	201-300	301-400	>400
MA - Source	115	0	0	0	0
MA - Target	115	18	3	0	4
NCI-A - Source	80	0	0	0	0
NCI-A - Target	80	4	1	2	0

Table 4: Anatomy - Sizes of Source and Target sets when generating repairing actions for the first time - part 2.

action to use. The chosen repairing action in this case was always the missing is-a relation itself (columns 'obvious choice self' and 'obvious choice more informative'). For the other cases the recommendation algorithm was used. RepOSE generated recommendations using WordNet. The running time was circa 4 minutes for MA and circa 2 minutes for NCI-A. The results are shown in Table 6. In most cases, there was only 1 recommendation for repairing a missing is-a relation. Sometimes, there were 2 or 3 recommendations. The recommendations can come from small sets of repairing actions or from large sets. For instance, for MA the system recommends for the missing is-a relation *(mandible, bone)* the following three repairing actions *(oral region cartilage/bone, bone)*, *(viscerocranium bone, bone)*, and *(mandible, lower jaw)*. The repairing actions are recommended from a Source set of 177 concepts and a Target set of 3 concepts. In almost all cases the recommendation by RepOSE was used (columns with 'use recommended' in Table 6) and in some cases the recommendation was ignored (columns with 'ignore recommended'). In many cases the missing is-a relation itself was used as repairing action (columns with 'self') but for 18 missing is-a relations in MA and 7 in NCI-A a more informative repairing action was used (columns with 'more informative')[10] thereby adding new knowledge to the network.

	total	explicitly repaired	repaired by others	obvious choice self	obvious choice more informative	ask recommendation
MA	120	101	19	28	0	73
NCI-A	87	87	0	7	0	80

Table 5: Anatomy - Repaired missing is-a relations.

After repairing the initially detected missing is-a relations we started a new round of detection. We found 6 new missing is-a relations (of which 5 needed to be repaired) in MA and 10 (of which 7 needed to be repaired) in NCI-A. Every newly derived missing is-a relation was caused by the repairing of a missing is-a relation in the other ontology for which the repairing action was more informative than the missing is-a relation itself. Among these new missing is-a relations, 4 in MA and 4 in NCI-A appear separately

[10] An expert in the anatomy domain might have been able to find additional more informative repairing actions.

	total	use recommended self	use recommended more informative	ignore recommended self	ignore recommended more informative
MA	73	52	16	3	2
NCI-A	80	73	6	0	1

Table 6: Anatomy - Recommendations.

after the execution of the relevant repairing action. Common to these new missing is-a relations is that each concept in the new missing is-a relation in one ontology is equivalent to a concept in the executed repairing action in the other ontology. For instance, the new missing is-a relation *(Thoracic_Aorta, Artery)* in NCI-A was caused by the execution of repairing action *(thoracic aorta, artery)* in MA, and the new missing is-a relation *(venule endothelium, vein endothelium)* in MA was caused by the execution of *(Venule_Endothelium, Vein_Endothelium)* in NCI-A. Other new missing is-a relations appeared together in the respective ontology, of which 1 and 3 were redundant in MA and NCI-A, respectively. For example, in NCI-A, after the missing is-a relation *(Myocardium, Muscle)* in MA was repaired by *(Striated_Muscle_Tissue, Muscle)*, the new missing is-a relations *(skeletal muscle tissue, muscle)* and *(striated muscle tissue, muscle)* appeared together, in which the former was redundant with respect to the latter.

After these missing is-a relations were repaired, no more were detected and the debugging session was concluded.

7.1.2. Using the extended algorithm for generation of repairing actions

We have also experimented with the extended algorithm for generating repairing actions. Table 7 shows the influences between different missing is-a relations that can be computed using our extended algorithm. The last column (ST) shows the number of missing is-a relations where x's and y's of other missing is-a relations occur in both Source and Target sets. For the other columns the x's and y's only occur in Source or Target, but not in both. For instance, for MA there are 22 missing is-a relations whose Source or Target set contain x and y from one other missing is-a relation. We see that for a majority of the missing is-a relations detected initially (94/115 for MA and 67/80 for NCI-A) there are influences. An interesting observation is that in several cases missing is-a relations that have the same number of influences from other missing is-a relations, actually influence each other. For instance, in NCI-A we find missing is-a relations between each of *Bronchus_Basement_Membrane*, *Bronchus_Cartilage*, *Bronchus_Lamina_Propria*, *Bronchus_Submucosa*, and the concept *Bronchus_Connective_Tissue*. Repairing one of these missing is-a relations influences the repairing actions of all the others. We found several such clusters, among others for instance, in MA concerning *body cavity/lining*, *lymphoid tissue*, and *brain nucleus* with 7, 4 and 6 missing is-a relations, respectively.

7.2. OAEI Bibliography

The OAEI Bibliography experiments deal with ontologies from the 2010 Benchmark track in OAEI. We use the ontologies in the bibliography domain called 101, 301,

	total	1	2	3	4	5	6	7	8	9	10	11-15	16-35	ST
MA	94	22	5	3	5	19	10	8	0	4	0	14	0	4
NCI-A	67	14	22	3	1	2	0	0	0	0	0	6	6	13

Table 7: Influence between repairing actions of different missing is-a relations - in Source or Target.

302, 303 and 304 and retain from these ontologies only the taxonomic part[11]. There are mappings (equivalence and subsumption) between 101 and the other ontologies. The number of concepts and mappings is shown in Table 8.

	number of concepts	PRA - total mappings with 101	PRA - equivalence mappings with 101	PRA - subsumption mappings with 101
101	33	-	-	-
301	15	22	14	8
302	13	23	12	11
303	54	18	16	2
304	39	30	28	2

Table 8: Bibliography ontologies network.

We consider 2 cases. In the first case we debug 4 small networks, each consisting of 101 and one other ontology and their PRA. This led to the detection of 2, 18, 1 and 9 missing is-a relations for the respective networks and these were repaired by the addition of 2, 7, 1 and 7 is-a relations, respectively. In the second case we consider the five ontologies and the four PRAs as one network. This lead to the detection of 48 missing is-a relations in the ontologies and these were repaired by the addition of 16 is-a relations.

The test runs for this experiment were done by the second author. As he is familiar with the domain, we considered him as a domain expert. As it was obvious which repairing actions to choose, the recommendation algorithm was not used. The largest session for this case took less than 5 minutes.

7.2.1. Case 1 - four small networks

For the first case, the running time for the detection algorithm was around 10 seconds per network. The results for this case are given in Table 9. Information about the sizes of the Source and Target sets for the missing is-a relations to repair is given in Table 10. Many of the Source and Target sets only contain one element. For 4 of the missing is-a relations there is only 1 repairing action. Table 11 shows the results of the repairing. All missing is-a relations except one in 101 were repaired explicitly. More informative repairing actions were used for 101 in network 101-302 and for 101 and

[11]This means we only keep internal concepts and the subClassOf statements. Further, we only retain mappings between concepts (but not between relations).

304 in network 101-304. After having repaired these missing is-a relations, no more were detected and the debugging session was concluded.

	total	equivalence	redundant	to repair
101	1 (1)	0 (0)	0 (0)	1 (1)
301	1 (1)	0 (0)	0 (0)	1 (1)
101	17 (17)	0 (0)	11 (11)	6 (6)
302	1 (1)	0 (0)	0 (0)	1 (1)
101	0 (0)	0 (0)	0 (0)	0 (0)
303	1 (1)	0 (0)	0 (0)	1 (1)
101	4 (4)	0 (0)	0 (0)	4 (4)
304	5 (5)	0 (0)	1 (1)	4 (4)

Table 9: Bibliography case 1 - Missing is-a relations detected during the whole debugging session. In parentheses the missing is-a relations that are initially detected.

	total	1	2-10	11-20	>20
101 - Source	1	1	0	0	0
101 - Target	1	1	0	0	0
301 - Source	1	1	0	0	0
301 - Target	1	1	0	0	0
101 - Source	6	6	0	0	0
101 - Target	6	1	0	5	0
302 - Source	1	1	0	0	0
302 - Target	1	1	0	0	0
101 - Source	0	0	0	0	0
101 - Target	0	0	0	0	0
303 - Source	1	1	0	0	0
303 - Target	1	1	0	0	0
101 - Source	4	1	3	0	0
101 - Target	4	0	4	0	0
304 - Source	4	3	1	0	0
304 - Target	4	0	4	0	0

Table 10: Bibliography case 1 - Sizes of Source and Target sets when generating repairing actions for the first time.

7.2.2. Case 2 - one network

The running time for the detection algorithm for case 2 was around 40 seconds. For this case (Table 12) for 101 we find 22 missing is-a relations of which 12 are redundant. Of the remaining 10 we have 2 that according to the network are equivalence relations (*(Chapter:BookPart, InBook:InBook)* and *(InBook:InBook, Chapter:BookPart)*). For 301, 302 and 303 we find 1 missing is-a relation each. For 304 we find 23 missing

	total	explicitly repaired	repaired by others	obvious choice self	obvious choice more informative
101	1	1	0	1	0
301	1	1	0	1	0
101	6	6	0	3	3
302	1	1	0	1	0
101	0	0	0	0	0
303	1	1	0	1	0
101	4	3	1	2	1
304	4	4	0	3	1

Table 11: Bibliography case 1 - Repaired missing is-a relations.

is-a relations of which 13 are redundant[12]. Information about the sizes of the Source and Target sets for the missing is-a relations to repair is given in Table 13. For each of 301, 302 and 303 there is a missing is-a relation with only 1 repairing action. Table 14 shows the results of the repairing. All missing is-a relations except two in 101 were repaired explicitly. More informative repairing actions appear in 101 and 304. After having repaired these missing is-a relations, we found 3 new missing is-a relations of which 1 needed to be repaired.

	total	equivalence	redundant	to repair
101	22 (22)	2 (2)	12 (12)	8 (8)
301	1 (1)	0 (0)	0 (0)	1 (1)
302	1 (1)	0 (0)	0 (0)	1 (1)
303	1 (1)	0 (0)	0 (0)	1 (1)
304	23 (20)	0 (0)	15 (13)	7 (7(*))

Table 12: Bibliography case 2 - Missing is-a relations detected during the whole debugging session. In parentheses the missing is-a relations that are initially detected.

7.3. ONKI

In the ONKI experiment we debug a network consisting of two ontologies and their PRA as available in the Finnish Ontology Library Service ONKI [48, 25]. The maritime ontology MERO contains ca 1400 concepts and is maintained by the Finnish Transport Agency. KOKO is Finnish national resource created by aligning an upper ontology and several domain ontologies. It contains ca 32,000 concepts. The PRA between KOKO and MERO contains 266 equivalence mappings. (See Table 15.)

[12]The (*) in Table 12 marks the fact that of the 7 missing is-a relations to repair initially, one will become redundant later on in the debugging process. Further, an additional missing is-a relation will be found later on in the debugging process.

	total	1	2-10	11-20	>20
101 - Source	8	5	3	0	0
101 - Target	8	0	4	4	0
301 - Source	1	1	0	0	0
301 - Target	1	1	0	0	0
302 - Source	1	1	0	0	0
302 - Target	1	1	0	0	0
303 - Source	1	1	0	0	0
303 - Target	1	1	0	0	0
304 - Source	7	5	2	0	0
304 - Target	7	0	4	3	0

Table 13: Bibliography case 2 - Sizes of Source and Target sets when generating repairing actions for the first time.

	total	explicitly repaired	repaired by others	obvious choice self	obvious choice more informative
101	8	6	2	4	2
301	1	1	0	1	0
302	1	1	0	1	0
303	1	1	0	1	0
304	7	7	0	6	1

Table 14: Bibliography case 2 - Repaired missing is-a relations.

Our debugging leads to the detection of 25 missing is-a relations in KOKO and 37 in MERO . These were repaired by the addition of 23 is-a relations in KOKO and 34 in MERO.

	number of concepts	PRA - total mappings	PRA - equivalence mappings	PRA - subsumption mappings
KOKO	32044	-	-	-
MERO	1448	-	-	-
	-	266	266	0

Table 15: ONKI ontologies network.

The loading of the ontologies and PRA took ca 12 minutes. The running time for the detection algorithm was 140 seconds. Initially, we find 24 missing is-a relations for KOKO and MERO each (Table 16). The running time for the generation of repairing actions was ca 100 seconds for KOKO and 1 second for MERO. Table 18 shows the results of the repairing. Most missing is-a relations were repaired by using the missing is-a relation itself, 5 were repaired by a more informative repairing action and 2 were repaired by repairing of others. After having repaired these missing is-a relations, we found 1 new missing is-a relation for KOKO and 13 for MERO. All new missing is-a relations for MERO were between a sub-concept of *ship* and *ship* itself. Most were repaired by using the missing is-a relation itself and 2 were repaired by a more informative repairing action. The 2 more information repairing actions *(cargo-ship, ship)* and *(special-purpose ship, ship)* led to the repairing of 3 other missing is-a relations. No more new missing is-a relations were detected after this.

Information about the sizes of the Source and Target sets for the missing is-a relations to repair is given in Table 17. Many of the larger Source and Target sets were related to is-a relations between a sub-concept of *ship* and *ship* itself. For 10 missing is-a relations in MERO and 1 in KOKO there was only 1 repairing action. We used the recommendation for all missing is-a relations. In 2 cases a more informative repairing action was recommended and these were accepted.

	total	equivalence	redundant	to repair
KOKO	25 (24)	0 (0)	0 (0)	25 (24)
MERO	37 (24)	0 (0)	0 (0)	37 (24)

Table 16: ONKI - Missing is-a relations detected during the whole debugging session. In parentheses the missing is-a relations that are initially detected.

8. Discussion

8.1. Detecting missing is-a relations

In general, detecting defects in ontologies without the support of a dedicated system is cumbersome and unreliable. In the experiments outlined in this paper RepOSE clearly provided a necessary support.

	total	1	2-10	11-20	>20
KOKO - Source	23	4	19	0	0
KOKO - Target	23	6	14	2	1
MERO - Source	34	26	8	0	0
MERO - Target	34	11	9	4	10

Table 17: ONKI - Sizes of Source and Target sets

	total	explicitly repaired	repaired by others	self	more informative
KOKO	25	23	2	21	2
MERO	37	34	3	31	5

Table 18: ONKI - Repaired missing is-a relations.

The detection of missing is-a relations as proposed in this paper is based on the knowledge inherent in the network. This is both an advantage and a limitation. It is advantageous that no external knowledge is needed. However, there are also other methods that may identify missing is-a relations. As we note in Section 9, these approaches are complementary to our approach and could be integrated in a future version of RepOSE.

Another advantage of our logic-based approach is, that it is guaranteed that all missing is-a relations derivable from the network are found. Further, the detection algorithm is fast, with a running time of only 2 minutes for the Anatomy case.

A limitation of the approach described in this paper in practice is the assumption that the existing structure in the ontologies and the mappings are correct. In general, this assumption may not be satisfied. In this case missing is-a relations may be derived based on wrong information in the ontologies. For instance, in the Anatomy and ONKI experiments we found several missing is-a relations that may not be correct. A solution to this limitation is to consider the computed missing is-a relations as *candidate* missing is-a relations and introduce a validation step where a domain expert classifies these as missing or wrong. The missing is-a relations are then treated in the same way as in our approach, while the wrong is-a relations would lead to a debugging opportunity for removing wrong information from the ontologies or alignments.

8.2. Generating repairing actions

As generating all possible structural repairs in general is infeasible, our algorithms support a number of heuristics. In Section 5.1 we have shown how our algorithms for generating repairing actions support $\ll_A$, $\ll_{SH}$ and the single relation heuristic. Further, by showing the is-a relations between the concepts in the Source and Target sets, we have implicitly ordered the solutions for a missing is-a relation with respect to $\ll_I$.

8.3. Executing repairing actions

RepOSE stores information about all changes made, and computes and stores the consequences. Again, without the support of a dedicated system this is cumbersome and unreliable.

Repairing influenced the number of repairing actions for other missing is-a relations. For Anatomy, during the repairing process, for 25 missing is-a relations in MA and 11 in NCI-A the number of repairing actions changed. For each of these missing is-a relations the size of the Target sets increased while the Source sets remained the same. The increase of the number of possible repairing actions ranged from 1 to 35 with average 8 in MA, and from 4 to 24 with average 12 in NCI-A. For Bibliography case 1, there were no changes in the number of repairing actions for missing is-a relations. For Bibliography case 2, in 101, for 1 missing is-a relation the Target set decreased. For 304, for 1 missing is-a relation the Target set decreased and for 1 missing is-a relation the Target set increased. The Source sets remained the same.

Repairing also influenced the set of missing is-a relations. In Anatomy for MA 19 missing is-a relations were repaired due to the repairing of other missing is-a relations, in Bibliography case 2, there were 2 such is-a relations in 101, and in ONKI there were 2 such relations for KOKO and 3 for MERO. For Bibliography case 1, there were no changes in this respect.

Further, repairing leads to the further detection of missing is-a relations. Regarding Anatomy, we found 6 new missing is-a relations (of which 5 needed to be repaired) in MA and 10 (of which 7 needed to be repaired) in NCI-A. In Bibliography case 2 we found 3 new missing is-a relations of which 1 needed to be repaired. In ONKI we found 1 new missing is-a relation for KOKO and 13 for ONKI. No new missing is-a relations were found in Bibliography case 1. New missing is-a relations may be derived when more informative repairing actions are used and thus new information is added to the network. If the missing is-a relation itself is used for repairing, the induced ontology for the ontology network does not change and thus no new missing is-a relations would be derived from the ontology network. A new round of detection therefore only needs to be started when more informative repairing actions are used.

8.4. Recommending repairing actions

The recommendation algorithm computes the most informative repairing actions that are supported by domain knowledge. We used the recommendation in the Anatomy and ONKI experiments. In almost all cases the recommendation was used.

8.5. User interface

For the experiments described in this paper, RepOSE was very responsive and interaction with the user was easy.

The Source and Target set panels seem to be a convenient way to show all possible computed choices for repairing a missing is-a relations with implicit ordering with respect to $\ll_I$. For small ontologies all Source and Target sets are small enough to have a good visualization in the tool (e.g Bibliography experiment). However, for larger ontologies there will likely be some missing is-a relations for which these sets

are too large for a good visualization. For instance, in the Anatomy experiment, this happened for 27 missing is-a relations in MA and 10 in NCI-A.

Further, the visualization of the results of the recommendation algorithm is integrated in the Source and Targets panels.

Visualization of the justifications (derivation paths) of possible defects would be helpful to have at hand as well as a graphical display of the possible defects within their contexts in the ontologies addressed. This has been added to a later version of RepOSE and the usefulness was confirmed in work for the Swedish National Food Agency [26] in which we debugged a toxicology ontology created by the Swedish National Food Agency based on an alignment to MeSH [45].

An identified constraint of RepOSE is the fact that adding is-a relations not appearing in the computations in RepOSE can be a demanding undertaking. This could happen, for instance, when a domain expert has identified a missing is-a relation manually. Currently, these changes need to be conducted in the ontology files, but it would be useful to allow a user to do this via the system.

8.6. Influence of additional ontologies and PRAs in the network

The Bibliography experiment allows us to investigate the influence of additional ontologies and PRAs in the network. Indeed, the small networks in Bibliography case 1 are sub-networks of the network in Bibliography case 2.

When we compare the results of Bibliography cases 1 and 2, we notice that for 301, 302 and 303 the same missing is-a relations are found. The additional information given by the connections to the other ontologies than 101, had no influence in this case. For 304, however, we find 3 additional missing is-a relations in case 2. For 101, we find for the four networks in case 1 together, 11 missing is-a relations that need to be repaired. For case 2 we have 2 equivalence relations and 8 missing is-a relations that need to be repaired. Of the 11 missing is-a relations in case 1, there are 2 corresponding equivalence relations in case 2. Of the remaining 9, 8 also appear in case 2. The last missing is-a relation in case 1 (that was found in 101-302) is redundant in case 2 as it can be derived by combining knowledge from 304 with knowledge from 101 and 302. The experiment shows thus that debugging a network (e.g. the five ontologies in the experiment) gives better results than debugging only part of the network (e.g. two of the ontologies).

9. Related Work

9.1. Debugging missing is-a relations

There is not much work on debugging is-a relations in networked ontologies.

9.1.1. Detecting missing is-a relations

The work closest to our own is [4], in which the authors deal with the missing is-a relations as ontology nonalignments in the context of ontology enrichment. Given two pairs of terms between two ontologies which are linked by the same kind of relationship, if the two terms in one ontology are linked by an is-a relation while the corresponding terms in the other are not, it is deemed as a nonalignment. However,

the authors note that, depending on the specific relationship, not all nonalignments are defects and there is no conclusive solution for repairing. Further, in [38] the use of a PRA in the setting of ontology alignment is discussed. One of the approaches includes detecting missing is-a relations by using the structure of the ontologies and the PRA. Missing is-a relations are found by looking at pairs of equivalence mappings. If there is an is-a relation between the terms in the mappings belonging to one ontology, but there is no is-a relation between the corresponding terms in the other ontology, then it is concluded that an is-a relation is missing in the second ontology. The detected missing is-a relations are then added to the ontologies before starting the actual alignment process.

Detecting missing is-a relations may be seen as a special case of detecting relations. There is much work on finding relationships between terms in the ontology learning area [7]. In this setting, new ontology elements are derived from text using knowledge acquisition techniques. There is, however, also work specifically focused on the discovery of is-a relations. One paradigm is based on linguistics using lexico-syntactic patterns. The pioneering research conducted in this line is in [23], which defines a set of patterns indicating is-a relationships between words in the text. However, depending on the chosen corpora, these patterns may occur rarely. Thus, though the approach has a reasonable precision, its recall is very low. To overcome this, an approach proposed in [9] collects a corpus from the World Wide Web using Google and identifies patterns from this collective knowledge. An evaluation and extension of this work is in [62]. In [12] the authors propose an approach for detecting defects within an ontology based on patterns and antipatterns. Another paradigm is based on machine learning and statistical methods, such as k-nearest neighbors approach [42], association rules [43], bottom-up hierarchical clustering techniques [64], supervised classification [60] and formal concept analysis [8]. In contrast to the approach described in this paper, these detection approaches using knowledge external to the network, while our detection method uses the ontology network itself as the domain knowledge for the detection of is-a relations. Our approach is also able to deal with multiple ontologies at the same time rather than a single ontology. However, these detection approaches are complementary to ours, and results from them, after validation, could be used as supplement to the domain knowledge or for repairing.

9.1.2. Repairing missing is-a relations

Most approaches just add the detected missing is-a relations to the ontologies. This is, in our context, the simplest kind of structural repair. It essentially means that after removing redundancy the least information-preferred ($\ll_I$) solution is chosen.

The work in [39] focused on repairing a given set of missing is-a relations in a *single* ontology. The work contains algorithms for generating, recommending and executing repairing actions. However, there is no investigation on how to detect missing is-a relations using networked ontologies and no attempt to find new missing is-a relations during the repairing. The work in this paper can be seen as an extension of the work in [39] in two ways. A first extension is that we deal with a network of ontologies. The second extension is that the algorithms for generating, recommending and executing repairing actions described in this paper are extensions of the algorithms in [39] that can be used for single ontologies when a set of missing is-a relations is given.

In [37] algorithms are proposed to generate different ways to repair the missing is-a structure in single ontologies that can be represented as $\mathcal{ALC}$ acyclic terminologies. The algorithms are based on a tableau-based algorithm for satisfiability checking with unfolding on demand as well as our Source and Target sets approach. The current implementation is, however, only feasible for small ontologies.

9.2. Other relevant topics

In this subsection we describe work that addresses related topics, although not the actual problem described in this paper.

9.2.1. Debugging semantic defects

Most of the work in ontology debugging has addressed semantic defects and aims at identifying and removing logical contradictions, i.e. inconsistencies and incoherencies, from an ontology. Standard reasoners are used to identify the existence of a contradiction, and provide support for resolving and eliminating it [16]. Most work focuses on single and isolated ontologies, and there is little support for ontologies connected by mappings [20].

In [54] minimal sets of axioms are identified which need to be removed to render an ontology coherent. In [35, 34, 33] strategies are described for repairing unsatisfiable concepts detected by reasoners, explanation of errors, ranking erroneous axioms, and generating repair plans. Strategies for reducing the set of possible solutions are given in [47, 5]. In [21] the focus is on maintaining the consistency as the ontology evolves. It formalized the semantics of change for the OWL ontologies and proposed methods for detecting and resolving inconsistency at three different levels. An interactive approach for ontology debugging is presented in [56]. Complexity issues are discussed in [50].

In [44] and [29] 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 [52] further characterizes the problem as mapping revision. Another approach for debugging mappings is proposed in [63] where the authors focus on the detection of certain kinds of defects and redundancy. Using the theory of belief revision [22], it gives a rationality analysis for the logical properties of the revision algorithms. The approach in [30] deals with the inconsistencies introduced by the integration of ontologies, and unintended entailments validated by the user. Further, the work in [27] addresses the detection and repair of missing and wrong is-a relations and mappings. The repairing of missing is-a relations in that system is based on the basic algorithm in this paper. The approach was used for the alignment and debugging of ontologies for the Swedish National Food Agency [26].

There has been some work in the area of modular ontologies, that is relevant for the problem discussed in this paper. Modular ontologies can be seen as a set of local ontologies that are connected by directional mappings, which are called bridge rules, and where the intuition is that an ontology can import knowledge from another ontology. The mappings are often equivalence and subsumption relations. The main difference

with the mappings in our ontology networks is that these bridge rules are directional. This means that knowledge propagation only occurs in one direction. Regarding the detection of semantic defects, within a framework based on distributed description logics, it is possible to restrict the propagation of local inconsistency to the whole set of ontologies (e.g. [55]).

9.2.2. Debugging modeling defects

The properties of is-a can be used for detecting modeling defects. For instance, as mentioned in the introduction, based on the notions of identity, rigidity and dependence, not all is-a relations in existing ontologies make sense [19]. These is-a relations can be detected by checking these properties. In [59] shallow semantic representations are constructed for the concepts in the ontologies and four hypotheses regarding the semantic representations and the is-a relation are proposed and tested. Three of the hypotheses were confirmed in the tests. The authors claim that the proposed hypotheses can be used for detection of inappropriate is-a relations.

In [36] two reasoning services are proposed for detecting flaws in OWL property expressions. The defects relate to the property is-a hierarchy, domain and range axioms and property chains.

9.2.3. Logic-based approaches

Logic-based abduction. The generation of repairing actions in this paper can be seen as an abductive problem. A general description of the problem of logic-based abduction is the following [13]. Given a logical theory T formalizing a domain, a set M of atomic formula describing manifestations and a set H of formulae containing possible hypotheses, find an explanation S for M such that S $\subseteq$ H and T $\cup$ S is consistent and logically entails M. In our case, the domain theory T would be represented by the union of the ontologies. The manifestations in M are the missing is-a relations. The set H contains all is-a relations between concepts within the same ontology that are not derivable from the ontology network. Further, preference $\ll_A$ is essentially the subset or irredundancy criterion that is often used in logic-based abduction.

In general, finding all or multiple solutions is a hard problem [14]. In our setting, for generating repairing actions, we compute multiple solutions that satisfy the constraint that for every element in M, this element occurs in S or there is a more informative element in S (single relation heuristic in Section 2.3). This constraint disallows solutions for which a missing is-a relation (a, b) would be repaired by the combination of new is-a relations (a, x_1), (x_1, x_2), ..., (x_k, b); and thus reduces the search space drastically. For our application, this is, however, a reasonable assumption. Further, we do not require minimality according to $\ll_A$ and $\ll_{SH}$ during the generation, but do an additional check when it is needed. A consequence of these choices is that our algorithms perform well in practice and response times in RepOSE are low.

Recently, for single ontologies it has been shown in [37] that the problem of finding possible ways to repair the missing is-a structure in an ontology in general can be formalized as a generalized version of the TBox abduction problem as follows. Given a knowledge base KB in language $\mathcal{L}$, concepts C_i, D_i for $1 \leq i \leq m$, that are satisfiable w.r.t. KB, and such that KB $\cup \{ C_i \rightarrow D_i \mid 1 \leq i \leq m\}$ is coherent. Then a solution to the generalized TBox abduction problem for (KB, $\{(C_i, D_i) \mid 1 \leq i \leq m\}$) is any

finite set S_{GT} = $\{G_j \rightarrow H_j \mid j \leq n\}$ of TBox assertions in $\mathcal{L}'$ such that $\forall$ i: KB $\cup$ S_{GT} $\models$ $C_i \rightarrow D_i$. In our setting the languages $\mathcal{L}$ and $\mathcal{L}'$ only allow *named* concepts and we only consider is-a relations between those concepts. The (C_i, D_i) are the missing is-a relations within the ontology. A solution S_{GT} is a structural repair.

There is very few work related to TBox abduction. The work in [24] proposes an automata-based approach to TBox abduction using abduciles representing axioms that can appear in solutions. It is based on a reduction to the axiom pinpointing problem which is then solved with automata-based methods. A PTIME algorithm is proposed for the language $\mathcal{EL}$.

Belief revision. From the perspective of belief revision [2, 17], our repairing approach could be deemed as an instantiation of belief expansion. It accommodates the structural repair as a new piece of information, and adds it to the ontology without checking consistency. Since in our setting the ontologies contain only named concepts and subsumption axioms, our repairing approach does not introduce logical contradictions.

9.2.4. Ontology engineering

Ontology engineering deals with the ontology development process, the ontology life cycle, and the methodologies, tools and languages for building ontologies [18][13]. There are several ontology engineering methodologies and several of these are reviewed in, for instance, [32, 11, 57]. The methodologies have been developed based on experiences in different areas. There are also methods for modular ontology development (e.g. [49]).

According to [18] a methodology should define the activities management (which includes a feasibility study - pre-development), development (which includes domain analysis, conceptualization and implementation) and support (which includes maintenance and use of the ontology - post-development). Several of the methodologies are developed with a particular application in mind [32, 11] and the feasibility study would, for instance, answer the question whether the application would benefit from using an ontology. Most of the methodologies cover the development process, although the actual instantiation of the different steps may differ. Domain analysis may cover such things as developing scenarios and competency questions. In the conceptualization step domain terms are identified and possibly definitions from other ontologies can be integrated. The implementation phase can include a formalization. The post-development activities include adaptation of the ontology according to new requirements and using the ontology in applications. Unfortunately, although there has been work on methodologies for several years, the existing methodologies are not yet mature from a software engineering point of view [11]. They lack some project management processes, ontology development-oriented processes or integral processes.

As discussed in [32], many of the existing methodologies have separate stages in which first an informal description of the ontology is developed and then the ontology is represented in a formal knowledge representation language. The debugging that we propose in this paper would take place after the representation of the ontology in a

[13] In this book the term 'ontological engineering' is used.

formal knowledge representation language. We would argue that debugging should be an iterative and integrated part of the implementation step of the development phase.

The study in [57] collected data using interviews and questionnaires from 148 ontology engineering projects from industry and academia in different areas such as information systems, commerce, multimedia and tourism. They found that the larger projects used a methodology and among those METHONTOLOGY was most popular. Two of the major findings in the study were that there is not yet much tool support and tool support for the different steps in the methodologies was deemed necessary both for the development of ontologies as well as for gaining more acceptance for the methodologies. Further, quality of the developed ontologies is a major concern. Most projects did minor testing but tools for aiding in ontology evaluation may result in major efficiency gains. The work in this paper addresses these concerns at their core.

10. Conclusion

Semantically-enabled applications require high-quality ontologies and a key step towards this is the debugging of the ontologies. In this paper we have focused on one of the important kinds of defects in ontologies, namely the modeling defects, and in particular, defects in the is-a structure of ontologies. In Section 1 we have shown the influence of missing is-a relations on the quality of the results of semantically-enabled applications. We have also shown that missing is-a relations are a common problem (e.g. our experiments) even in small ontologies, although this has not received much attention yet. It is clear that tool support for debugging is needed and this paper presents a first step towards this. Later, ontology debugging may become an integrated part of ontology engineering and become a natural part of steps in ontology engineering methodologies.

In this paper we have proposed an approach for debugging the missing is-a structure of ontologies that are networked by a set of PRAs. We defined important notions and developed algorithms for detection and repair of missing is-a structure of ontologies. We also implemented a system and described and discussed experiments using ontologies from the OAEI and ONKI. We have shown that system support for detection and repairing of missing is-a relations is needed. Our approach is logic-based using knowledge that is inherent in the ontology network. The approach guarantees to find all logically derivable missing is-a relations. Further, as generating all possible structural repairs is infeasible and not very useful, our approach finds structural repairs that satisfy reasonable heuristics ($\ll_A$, $\ll_{SH}$ and the single relation heuristic). The extended algorithm also shows influences between the missing is-a relations and their solutions. Further, the solutions for a missing is-a relation are ordered with respect to $\ll_I$ and our recommendation algorithm recommends among these the most informative that are supported by domain knowledge. Our approach is the first that allows to add new knowledge to the ontologies in the repairing phase by using more informative repairing actions. The visualization of the possible repairing actions, their ranking and the recommendation was very useful.

The approach and system have, however, some limitations which require further work. Our detection algorithm is based on the knowledge inherent in the network, but,

there are methods that use external knowledge. As these approaches are complementary to our approach, they could be integrated in a future version of RepOSE. Further, we assume that the existing structure and mappings are correct. In the case where this does not hold, we need to deal with semantic defects both in the structure and in the mappings. As discussed before, the detection algorithm would generate candidate missing is-a relations which need to be validated by a domain expert. A candidate missing is-a relation validated to be correct would be treated in the same way as in this paper. A candidate missing is-a relation validated to be wrong would be based on wrong information in the ontologies or PRAs and lead to the detection and repairing of semantic defects. We would also need to investigate possible influences between semantic defects and modeling effects. Regarding visualization we want to add justifications of missing is-a relations as well as the possibility for users to add their manually detected missing is-a relations.

Further, we are interested in investigating the following issues. One issue is to deal with ontologies represented in more expressive representation languages. The techniques described in this paper may be partly used for these ontologies, but a number of side conditions (such as the consequences of negation and disjointness) will need to be taken into account. Further, with the availability of more and more Linked Open Data (LOD), it is interesting to investigate how this can be used in the debugging process. It would be possible to use the LOD as instances of concepts in the ontologies for detection of missing is-a relations. The detection could be based on set inclusion or based on similarities between concepts that take into account the instances (e.g. [61]). The LOD could also be used for validation. We also want to find ways to optimize the generation of results. For instance, for large ontologies the generation of structural repairs may take a lot of time, and we want to investigate ways to partition the set of missing is-a relations into subsets that can be processed independently. We also want to study the influence between repairing actions. For instance, it may be more important to repair the top level in the ontology first and in this case, the ranking approach should reflect this.

Acknowledgements

We thank the reviewers for their insightful comments and suggestions and Rajaram Kaliyaperumal for help with the ONKI experiment. Further, we acknowledge financial support from the Swedish National Graduate School in Computer Science (CUGS), the Swedish Research Council (VR) and the Swedish e-Science Research Centre (SeRC).

Availability

The system described in this paper is available for download at http://www.ida.liu.se/∼patla/research/RepOSE/.

References

[1] Preliminary results of the OAEI 2009. http://oaei.ontologymatching.org/2009/results.

[2] C E Alchourrón, P Gärdenfors, and D Makinson. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic*, 50(2):510–530, 1985.

[3] AMA. Adult mouse anatomical dictionary. http://www.informatics.jax.org/searches/AMA_form.shtml.

[4] M Bada and L Hunter. Identification of OBO nonalignments and its implications for OBO enrichment. *Bioinformatics*, 24(12):1448–1455, 2008.

[5] S Bail, B Parsia, and U Sattler. Declutter your justifications: Determining similarity between OWL explanations. In *1st International Workshop on Debugging Ontologies and Ontology Mappings*, pages 13–24, 2012.

[6] RJ Brachman. What IS-A is and isn't: An analysis of taxonomic links in semantic networks. *IEEE Computer*, 16(10):30–36, 1983.

[7] Ph Cimiano, P Buitelaar, and B Magnini. *Ontology Learning from Text: Methods, Evaluation and Applications*. IOS Press, 2005.

[8] Ph Cimiano, A Hotho, and S Staab. Learning concept hierarchies from text corpora using formal concept analysis. *Journal of Artificial Intelligence Research*, 24:305–339, 2005.

[9] Ph Cimiano and S Staab. Learning by googling. *ACM SIGKDD Explorations Newsletter*, 6(2):24–33, 2004.

[10] 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*, pages 623–637, 2009.

[11] O Corcho, M Fernandez-Lopez, and A Gomez-Perez. Methodologies, tools and languages for building ontologies. Where is their meeting point? *Data & Knowledge Engineering*, 46:41–64, 2003.

[12] 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.

[13] T Eiter and G Gottlob. The complexity of logic-based abduction. *Journal of the Association for Computing Machinery*, 42(1):3–42, 1995.

[14] T Eiter and K Makino. On computing all abductive explanations from a propositional Horn theory. *Journal of the Association for Computing Machinery*, 54(5):Article 24, 2007.

[15] F Feneis and W Dauber. *Pocket Atlas of Human Anatomy, 4th ed.* Thieme Verlag, 2000.

[16] G Flouris, D Manakanatas, H Kondylakis, D Plexousakis, and G Antoniou. Ontology change: classification and survey. *Knowledge Engineering Review*, 23(2):117–152, 2008.

[17] P Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. MIT Press, 1988.

[18] A Gomez-Perez, M Fernandez-Lopez, and O Corcho. *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web*. Springer, 2003.

[19] N Guarino. Some ontological principles for designing upper level lexical resources. In *1st International Conference on Language Resources and Evaluation*, 1998.

[20] P Haase and G Qi. An analysis of approaches to resolving inconsistencies in dl-based ontologies. In *International Workshop on Ontology Dynamics*, pages 188–202, 2007.

[21] P Haase and L Stojanovic. Consistent evolution of OWL ontologies. In *2nd European Semantic Web Conference*, pages 182–197, 2005.

[22] S O Hansson. *A Textbook of Belief Dynamics Theory: Theory Change and Database Updating*. Kluwer Academic Publishers, 1999.

[23] M Hearst. Automatic acquisition of hyponyms from large text corpora. In *14th International Conference on Computational Linguistics*, pages 539–545, 1992.

[24] T Hubauer, S Lamparter, and M Pirker. Automata-based abduction for tractable diagnosis. In *International Workshop on Description Logics*, pages 360–371, 2010.

[25] E Hyvönen, K Viljanen, J Tuominen, and K Seppälä. Building a national semantic web ontology and ontology service infrastructure - the FinnONTO approach. In *5th European Semantic Web Conference*, pages 95–109, 2008.

[26] V Ivanova, J Laurila Bergman, U Hammerling, and P Lambrix. Debugging taxonomies and their alignments: the ToxOntology - MeSH use case. In *1st International Workshop on Debugging Ontologies and Ontology Mappings*, pages 25–36, 2012.

[27] V Ivanova and P Lambrix. A unified approach for aligning taxonomies and debugging taxonomies and their alignments. In *10th Extended Semantic Web Conference*, 2013.

[28] Jena. http://jena.sourceforge.net/.

[29] Q Ji, P Haase, G Qi, P Hitzler, and S Stadtmuller. RaDON - repair and diagnosis in ontology networks. In *6th European Semantic Web Conference*, pages 863–867, 2009.

[30] 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*, pages 173–187, 2009.

[31] I Johansson and B Klein. Four kinds of ”is-a” relations: genus-subsumption, determinable subsumption, specification, and specialization. In *3rd International Workshop on Philosopy and Informatics*, 2006.

[32] D Jones, T Bench-Capon, and P Visser. Methodologies for ontology development. In *IT and KNOWS Conference of the 15th IFIP World Computer Congress*, pages 62–75, 1998.

[33] A Kalyanpur. *Debugging and Repair of OWL Ontologies*. PhD thesis, University of Maryland, College Park, 2006.

[34] A Kalyanpur, B Parsia, E Sirin, and B Cuenca-Gray. Repairing unsatisfiable concepts in OWL ontologies. In *3rd European Semantic Web Conference*, pages 170–184, 2006.

[35] A Kalyanpur, B Parsia, E Sirin, and J Hendler. Debugging unsatisfiable classes in OWL ontologies. *Journal of Web Semantics*, 3(4):268–293, 2006.

[36] M Keet. Detecting and revising flaws in OWL object property expressions. In *18th International Conference on Knowledge Engineering and Knowledge Management*, pages 252–266, 2012.

[37] 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*, pages 17–32, 2012.

[38] P Lambrix and Q Liu. Using partial reference alignments to align ontologies. In *6th European Semantic Web Conference*, pages 188–202, 2009.

[39] P Lambrix, Q Liu, and H Tan. Repairing the missing is-a structure of ontologies. In *4th Asian Semantic Web Conference*, pages 371–386, 2009.

[40] P Lambrix, G Qi, and M Horridge. *Proceedings of the 1st International Workshop on Debugging Ontologies and Ontology Mappings*. Linköping University Electronic Press, 2012.

[41] P Lambrix, L Strömbäck, and H Tan. Information integration in bioinformatics with ontologies and standards. In Bry and Maluszynski, editors, *Semantic Techniques for the Web: The REWERSE perspective*, pages 343–376. Springer, 2009.

[42] A Maedche, V Pekar, and S Staab. Ontology learning part one - on discovering taxonomic relations from the web. In Zhong, Liu, and Yao, editors, *Web Intelligence*, pages 301–320. Springer, 2003.

[43] A Maedche and S Staab. Discovering conceptual relations from text. In *14th European Conference on Artificial Intelligence*, pages 321–325, 2000.

[44] C Meilicke, H Stuckenschmidt, and A Tamilin. Repairing ontology mappings. In *20th National Conference on Artificial Intelligence*, pages 1408–1413, 2007.

[45] MeSH. Medical subject headings. http://www.nlm.nih.gov/mesh/.

[46] NCI-A. National cancer institute - anatomy. http://www.cancer.gov/cancerinfo/terminologyresources/.

[47] T Nguyen, R Power, P Piwek, and S Williams. Measuring the understandability of deduction rules for OWL. In *1st International Workshop on Debugging Ontologies and Ontology Mappings*, pages 1–12, 2012.

[48] ONKI. http://onki.fi/en/.

[49] T Özacar, Ö Öztürk, and MO Ünalir. ANEMONE: An environment for modular ontology development. *Data & Knowledge Engineering*, 70:504–526, 2011.

[50] R Penaloza and B Sertkaya. On the complexity of axiom pinpointing in the EL family of description logics. In *12th International Conference on Principles of Knowledge Representation and Reasoning*, pages 280–289, 2010.

[51] PubMed. http://www.ncbi.nlm.nih.gov/pubmed/.

[52] G Qi, Q Ji, and P Haase. A conflict-based operator for mapping revision. In *8th International Semantic Web Conference*, pages 521–536, 2009.

[53] OBO RO. http://code.google.com/p/obo-relations/.

[54] S Schlobach. Debugging and semantic clarification by pinpointing. In *2nd European Semantic Web Conference*, pages 226–240, 2005.

[55] L Serafini, A Borgida, and A Tamilin. Aspects of distributed and modular ontology reasoning. In *19th International Joint Conference on Artificial Intelligence*, pages 570–575, 2005.

[56] 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.

[57] E Simperl, M Mochol, and T Bürger. Achieving maturity: the state of practice in ontology engineering in 2009. *International Journal of Computer Science and Applications*, 7:45–65, 2010.

[58] B Smith, W Ceusters, B Klagges, J Köhler, A Kumar, J Lomax, C Mugall, F Neuhaus, AL Rector, and C Rosse. Relations in biomedical ontologies. *Genome Biology*, 6:R46, 2005.

[59] G Solskinnsbakk, JA Gulla, V Haderlein, P Myrseth, and O Cerrato. Quality of hierarchies in ontologies and folksonomies. *Data & Knowledge Engineering*, 74:13–25, 2012.

[60] V Spiliopoulos, G Vouros, and V Karkaletsis. On the discovery of subsumption relations for the alignment of ontologies. *Journal of Web Semantics*, 8:69–88, 2010.

[61] H Tan, V Jakoniene, P Lambrix, J Aberg, and N Shahmehri. Alignment of biomedical ontologies using life science literature. In *International Workshop on Knowledge Discovery in Life Science Literature*, pages 1–17, 2006.

[62] WR van Hage, S Katrenko, and G Schreiber. A method to combine linguistic ontology-mapping techniques. In *4th International Semantic Web Conference*, pages 732–744, 2005.

[63] P Wang and B Xu. Debugging ontology mappings: a static approach. *Computing and Informatics*, 27:21–36, 2008.

[64] E Zavitsanos, G Paliouras, G A Vouros, and S Petridis. Discovering subsumption hierarchies of ontology concepts from text corpora. In *IEEE / WIC / ACM International Conference on Web Intelligence*, pages 402–408, 2007.

Appendix

Missing is-a relation: is-a relation that should be in the ontology according to the domain model, but is not. Only used when explicitly stated.
Missing is-a relation derivable from the ontology network: missing is-a relation that is detected by using the domain knowledge inherent in the ontology network (see Definition 4). Unless explictly mentioned, when using 'missing is-a relation' in the paper, we mean this kind.
Regarding detection, we refer to: *Initially detected missing is-a relation:* missing is-a relation detected by running the detection algorithm before any repairing is done. *Additionally detected missing is-a relation* or *missing is-a relation detected later in the debugging process*: missing is-a relation detected by running the detection algorithm after some repairing is done.
Regarding repairing, we refer to: *Missing is-a relation for which the domain expert explicitly selected a repairing action:* the repairing is done by the domain expert through the selection of a solution from Source $\times$ Target. *Missing is-a relation that was repaired through repairing actions of other missing is-a relations:* the missing is-a relation became derivable after other missing is-a relations were repaired, and therefore did not need to be explicitly repaired.

Table 19: Missing is-a relations - terminology.