Repairing missing is-a structure in ontologies is an abductive reasoning problem

Patrick Lambrix[1,2], Fang Wei-Kleiner[1], Zlatan Dragisic[1,2], Valentina Ivanova[1,2]

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

Abstract. With the increased use of ontologies in semantically-enabled applications, the issue of debugging defects in ontologies has become increasingly important. These defects can lead to wrong or incomplete results for the applications. Debugging consists of the phases of detection and repairing. In this paper we focus on the repairing phase of a particular kind of defects, i.e., the missing relations in the is-a hierarchy. We show that this can be formalized as an abduction problem. Further, we define properties for the ontology, the set of is-a relations to repair and the domain expert, as well as preference criteria on solutions and discuss the influences of these properties and criteria on the existence of solutions for the abduction problem. We also discuss the consequences of our analyses of the repairing problem for the development and use of debugging systems.

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. Defects in ontologies can take different forms (e.g., [16]). 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. Debugging consists of two phases - detection and repair. Most work up to date has focused on debugging the semantic defects in an ontology (see related work in Section 5).

Modeling defects have mainly been discussed for taxonomies, i.e., from a knowledge representation point of view, a simple kind of ontologies. The focus has been on defects regarding the is-a structure (Section 5). In addition to its importance for the correct modeling of a domain, the structural information in ontologies is also important in semantically-enabled applications such as ontology-based search and annotation. In this paper we formalize the problem of repairing the is-a structure of ontologies.

There are different ways to detect missing is-a relations (Section 5). One way is inspection by domain experts. Another way is to use ontology learning techniques or patterns. When the ontology is part of a network of ontologies connected by mappings, missing is-a relations may be detected using logical derivation in the network. However, although there are many approaches to detect missing is-a relations, these approaches,

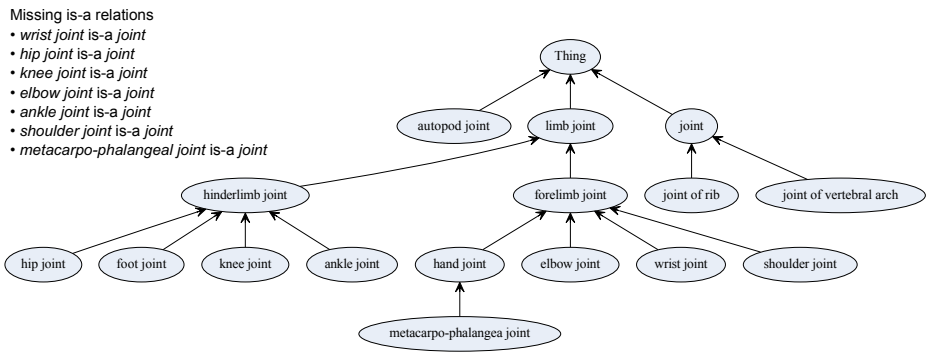

Fig. 1: A part of MA concerning the concept *joint*.

in general, do not detect *all* missing is-a relations. For instance, although the precision for the linguistic patterns approaches is high, their recall is usually very low.

In this paper we assume that the detection phase has been performed. We assume that we have obtained a set of missing is-a relations for a given ontology (validated or not) and focus on the repairing phase. In the ideal case where our set of missing is-a relations contains *all* missing is-a relations, the repairing phase is easy. We just add all missing is-a relations to the ontology and a reasoner can compute all logical consequences. However, when the set of missing is-a relations does not contain all missing is-a relations - and this is the common case - there are different ways to repair the ontology.

For instance, Figures 1 and 2 (T) show a small ontology representing a part of the Adult Mouse Anatomy (MA) ontology concerning joint, that is relevant for our discussion. M is a set of detected missing is-a relations. Adding these relations to the ontology will repair the missing is-a structure. However, there are other more interesting possibilities. For instance, adding limb-joint $\dot{\sqsubseteq}$ joint also repairs the missing is-a structure. Further, this is-a relation is correct according to the domain and constitutes a new is-a relation that was not derivable from the ontology and not originally detected by the detection algorithm.

The contributions of this paper are the following. First, in Section 2 we formalize the problem of repairing missing is-a structure as an abduction problem (extension of [18]) and introduce two decision problems - (i) do solutions exist, and (ii) if so, find a solution. We also define different properties for the ontology, the set of is-a relations to repair, and the domain expert and discuss the influences of these properties on the existence of solutions for the abduction problem. In general, when solutions exist, there may be many solutions. As not all solutions are equally interesting, in Section 3 we propose two preference criteria on the solutions as well as different ways to combine these. We also discuss the decision problems for the criteria and their preferences. Further, in Section 4 we discuss the consequences of our analyses for debugging in practice.

C = { autopod-joint, limb-joint, hinderlimb-joint, hip-joint, foot-joint, knee-joint, ankle-joint, forelimb-joint, hand-joint, elbow-joint, wrist-joint, shoulder-joint, metacarpo-phalangeal-joint, joint, joint-of-rib, joint-of-vertebral-arch }

T = { autopod-joint $\dot{\sqsubseteq}$ $\top$, limb-joint $\dot{\sqsubseteq}$ $\top$, hinderlimb-joint $\dot{\sqsubseteq}$ limb-joint , hip-joint $\dot{\sqsubseteq}$ hinderlimb-joint, foot-joint $\dot{\sqsubseteq}$ hinderlimb-joint, knee-joint $\dot{\sqsubseteq}$ hinderlimb-joint, ankle-joint $\dot{\sqsubseteq}$ hinderlimb-joint, forelimb-joint $\dot{\sqsubseteq}$ limb-joint, hand-joint $\dot{\sqsubseteq}$ forelimb-joint, elbow-joint $\dot{\sqsubseteq}$ forelimb-joint, wrist-joint $\dot{\sqsubseteq}$ forelimb-joint, shoulder-joint $\dot{\sqsubseteq}$ forelimb-joint, metacarpo-phalangeal-joint $\dot{\sqsubseteq}$ hand-joint, joint $\dot{\sqsubseteq}$ $\top$, joint-of-rib $\dot{\sqsubseteq}$ joint, joint-of-vertebral-arch $\dot{\sqsubseteq}$ joint }

M = { wrist-joint $\dot{\sqsubseteq}$ joint, hip-joint $\dot{\sqsubseteq}$ joint, knee-joint $\dot{\sqsubseteq}$ joint, elbow-joint $\dot{\sqsubseteq}$ joint, ankle-joint $\dot{\sqsubseteq}$ joint, shoulder-joint $\dot{\sqsubseteq}$ joint, metacarpo-phalangeal-joint $\dot{\sqsubseteq}$ joint }

H_1 = set of all is-a relations that are correct according to the domain
H_2 = $H_1 \setminus$ {autopod-joint $\dot{\sqsubseteq}$ limb-joint, limb-joint $\dot{\sqsubseteq}$ joint}
H_3 = $H_2 \cup$ {hinderlimb-joint $\dot{\sqsubseteq}$ joint-of-rib, forelimb-joint $\dot{\sqsubseteq}$ joint-of-vertebral-arch}
H_4 = { A $\dot{\sqsubseteq}$ B | A, B $\in C$ }

Let $\mathcal{P}_i$ = GTAP(T, C, H_i, M) for $1 < i < 4$

Fig. 2: Small example.

2 Abduction Framework

In the following we explain how the problem of finding possible ways to repair the missing is-a structure in an ontology is formalized as a generalized version of the TBox abduction problem (extension of [18]). We assume that our ontology is represented using a TBox T. The identified is-a relations to repair are then represented by a set M of atomic concept subsumptions. As discussed in Section 1, M usually does not contain *all* missing is-a relations. To repair the ontology, it should be extended with a set S of atomic concept subsumptions (repair) such that the extended ontology is consistent and the missing is-a relations are derivable from the extended ontology. However, the added atomic concept subsumptions should be correct according to the domain[1]. Therefore, we assume that a domain expert validates whether an atomic concept subsumption is correct and these validated to be correct atomic concept subsumptions are collected in a set H. We note that in practice H is not known beforehand, but acts as an oracle. It is then required that $S \subseteq H$. The following definition formalizes this.

Definition 1 (Generalized TBox Abduction) *Let T be a consistent TBox and C be a set of atomic concepts. Let $M = \{\ C_i \dot{\sqsubseteq} D_i \mid 1 \leq i \leq m\ \}$ be a set of TBox assertions where $C_i, D_i \in C$. Let $H = \{E_i \dot{\sqsubseteq} F_i \mid 1 \leq i \leq n\}$ where $E_i, F_i \in C$. A solution to the generalized TBox abduction problem (GTAP) (T, C, H, M) is any finite set $S \subseteq H$, such that $T \cup S$ is consistent and $T \cup S \models M$. The set of all such solutions is denoted as $\mathcal{S}(T, C, H, M)$.*

Moreover, we are interested in two problems which are useful in practice. The first problem is the so called existence problem. That is, the decision problem of whether $\mathcal{S}(T, C, H, M) \neq \emptyset$. Clearly, with a concrete debugging task the existence problem should be answered at the beginning. If the answer to the existence problem is positive,

[1] In the remainder of this paper when we say that concept subsumptions or is-a relations are *correct*, we mean correct according to the domain.

we are interested in finding *a* solution[2]. This is normally a realistic goal in practice, since the number of all solutions could be considerably big.

Next, we discuss different properties of T, H and M and how these properties and their combinations affect the existence and type of solutions. In this discussion we make the assumption that the domain is consistent.

The GTAP definition requires T to be consistent. If this would not be the case, it would mean that the original ontology is not consistent. In this case approaches for debugging semantic defects could be used to obtain a consistent ontology. We also note that if T is not consistent then there are no solutions satisfying the definition (as $T \cup S$ would be inconsistent). However, even if T is consistent, it is possible that T contains relations which are not correct. It would mean that the developers introduced a modeling defect. Therefore, we indentify two cases for T - all the is-a relations in T are correct ('T correct' in Table 1), or not ('T not correct' in Table 1).

For M there are 2 cases. In the first case we assume that all is-a relations in M are correct, and thus they are really missing is-a relations ('Missing' in Table 1). In the second case M may contain missing as well as wrong is-a relations ('Missing + Wrong' in Table 1). This is a common case when possible missing is-a relations are generated by detection algorithms (e.g., using patterns or ontology learning methods) and not validated by a domain expert. It may also occur when M is generated by domain experts (e.g., using inspection) - as it is an error-prone task, the experts may make mistakes.

For H we identified the following interesting cases. In the first case ('Complete Knowledge' in Table 1) H contains all correct is-a relations and no others. In this case we are sure that if an is-a relation belongs to H, it is correct and if not, it is not correct. This case represents the ideal situation of an all-knowing domain expert. In the second case ('Partial-Correct' in Table 1) H contains only correct is-a relations, but not necessarily all. This case represents a domain expert who knows a part of the domain well. If the domain expert validates an is-a relation as correct, it is correct. Otherwise, the is-a relation is wrong or the domain expert does not know. An approximation of this case is when using several domain experts and a skeptical approach. We only consider an is-a relation correct if all domain experts validate it as correct. In the third case ('Wrong' in Table 1) H may contain relations that are not correct. In this case, the domain expert can make mistakes regarding the validation of is-a relations. Some wrong is-a relations may be validated as correct. This is a common case as exemplified by the use case in [12]. The fourth and fifth cases represent situations where there is no domain expert. In the fourth case all possible is-a relations are validated as correct and thus $H = \{E_i \dot{\sqsubseteq} F_i \mid E_i, F_i \in C\}$ ('No Expert' in Table 1). In the fifth case (not in in Table 1) no is-a relation is validated as correct and thus $H = \emptyset$. For the fifth case there can be only 1 solution, i.e., S = $\emptyset$ and this only in the case where $T \models M$ (and thus the is-a relations in M were not actually missing). We have the following relations between the different cases. Let H_c, H_{pc}, H_w, H_{no} be sets corresponding to the cases 1-4, respectively and related to the same domain. Then $H_{pc} \subset H_c \subset H_{no}$ and $H_w \subset H_{no}$. Therefore, we also have that $\mathcal{S}(T, C, H_{pc}, M) \subset \mathcal{S}(T, C, H_c, M) \subset \mathcal{S}(T, C, H_{no}, M)$ and $\mathcal{S}(T, C, H_w, M) \subset \mathcal{S}(T, C, H_{no}, M)$. In our example in Figure 2 H_1, H_2, H_3 and H_4 are examples of H_c, H_{pc}, H_w and H_{no}, respectively.

[2] Often regarding various preference criteria, see Section 3.

M / H	Missing: T correct	Missing: T not correct
Complete Knowledge	$M \subseteq H$ M is solution All solutions are correct	$M \subseteq H$ No solution if $T \cup M$ inconsistent M is solution iff $T \cup M$ consistent All solutions are correct
Partial-Correct	$M \subseteq H$ or $M \not\subseteq H$ No solution if $M \not\subseteq H \wedge T \cup H \not\models M$ if $M \subseteq H$ then M is a solution if $M \not\subseteq H \wedge T \cup H \models M$ then H is a solution All solutions are correct	$M \subseteq H$ or $M \not\subseteq H$ No solution if $T \cup M$ inconsistent No solution if $M \not\subseteq H \wedge T \cup H \not\models M$ No solution if $\forall S : S \neq \emptyset \wedge S \subseteq H \rightarrow T \cup S$ inconsistent if $T \cup M$ consistent $\wedge$ $M \subseteq H$ then M is a solution if $T \cup H$ consistent $\wedge$ $M \not\subseteq H \wedge T \cup H \models M$ then H is a solution All solutions are correct
Wrong	$M \subseteq H$ or $M \not\subseteq H$ No solution if $M \not\subseteq H \wedge T \cup H \not\models M$ No solution if $\forall S : S \neq \emptyset \wedge S \subseteq H \rightarrow T \cup S$ inconsistent if $M \subseteq H$ then M is a solution if $M \not\subseteq H \wedge T \cup H \models M \wedge T \cup H$ consistent then H is a solution If M is solution, then correct, no guarantee otherwise	$M \subseteq H$ or $M \not\subseteq H$ No solution if $T \cup M$ inconsistent No solution if $M \not\subseteq H \wedge T \cup H \not\models M$ No solution if $\forall S : S \neq \emptyset \wedge S \subseteq H \rightarrow T \cup S$ inconsistent if $T \cup M$ consistent $\wedge$ $M \subseteq H$ then M is a solution if $T \cup H$ consistent $\wedge$ $M \not\subseteq H \wedge T \cup H \models M$ then H is a solution If M is solution, then correct (but not $T \cup M$), no guarantee otherwise
No Expert	$M \subseteq H$ M is solution If M is solution, then correct, no guarantee otherwise	$M \subseteq H$ M is solution iff $T \cup M$ consistent If M is solution, then correct (but not $T \cup M$), no guarantee otherwise

M / H	Missing + Wrong: T correct	Missing + Wrong: T not correct
Complete Knowledge	$M \not\subseteq H$ No solution	$M \not\subseteq H$ No solution if $T \cup M$ inconsistent No solution if $T \cup H \not\models M$ No solution if $\forall S : S \neq \emptyset \wedge S \subseteq H \rightarrow T \cup S$ inconsistent if $T \cup H$ consistent $\wedge$ $T \cup H \models M$ then H is a solution The solutions are not correct
Partial-Correct	$M \not\subseteq H$ No solution	$M \not\subseteq H$ No solution if $T \cup M$ inconsistent No solution if $T \cup H \not\models M$ No solution if $\forall S : S \neq \emptyset \wedge S \subseteq H \rightarrow T \cup S$ inconsistent if $T \cup H$ consistent $\wedge$ $T \cup H \models M$ then H is a solution The solutions are not correct
Wrong	$M \subseteq H$ or $M \not\subseteq H$ No solution if $T \cup M$ inconsistent No solution if $M \not\subseteq H \wedge T \cup H \not\models M$ No solution if $\forall S : S \neq \emptyset \wedge S \subseteq H \rightarrow T \cup S$ inconsistent if $T \cup M$ consistent $\wedge$ $M \subseteq H$ then M is a solution if $M \not\subseteq H \wedge T \cup H \models M \wedge T \cup H$ consistent then H is a solution The solutions are not correct	$M \subseteq H$ or $M \not\subseteq H$ No solution if $T \cup M$ inconsistent No solution if $M \not\subseteq H \wedge T \cup H \not\models M$ No solution if $\forall S : S \neq \emptyset \wedge S \subseteq H \rightarrow T \cup S$ inconsistent if $T \cup M$ consistent $\wedge$ $M \subseteq H$ then M is a solution if $T \cup H$ consistent $\wedge$ $M \not\subseteq H \wedge T \cup H \models M$ then H is a solution The solutions are not correct
No Expert	$M \subseteq H$ M is solution iff $T \cup M$ consistent The solutions are not correct	$M \subseteq H$ M is solution iff $T \cup M$ consistent The solutions are not correct

Table 1: Different combinations of cases for T, H and M.

Table 1 shows the properties for T, H, M and their combinations. For each combination we give information about the relationship between M and H, the existence of solutions and the correctness of the solutions. Here, we summarize the findings.

An ideal situation is the case where the domain expert has complete knowledge (H contains all correct is-a relations and no others) and T and M contain only correct is-a relations. In this case, $M \subseteq H$. Further, M is a solution and all solutions are correct.

For any case where $T \cup M$ is inconsistent, there is no solution. Indeed, for any solution S we have that $T \cup S \models M$ and thus $T \cup S$ would not be consistent.

In the cases where M contains wrong is-a relations, there may be no solutions. If there are solutions, these are not correct. Further, correctness of solutions is only guaranteed when M does not contain wrong is-a relations and H represents complete knowledge or partial-correct.

There are no solutions if $T \cup S$ is inconsistent for every non-empty subset S of H.

If $M \subseteq H$ and $T \cup M$ is consistent, then M is a solution. If $M \not\subseteq H$, $T \cup H$ is consistent and $T \cup H \models M$, then H is a solution.

In the case of no expert ($H = \{E_i \dot{\sqsubseteq} F_i \mid E_i, F_i \in C\}$) we have that $M \subseteq H$ and all is-a relations are allowed in the solution. Therefore, if $T \cup M$ is consistent, then M is a solution, otherwise there is no solution. However, as there is no domain expert, there is no guarantee that any solution other than M is correct. Further, in the cases where M contains wrong is-a relations, M is a solution, but not correct. As there is no validation, only logical consistency can be guaranteed, but no correctness.

3 Solutions with preference criteria

There can be many solutions for a GTAP and, as explained in Section 1, not all solutions are equally interesting. Therefore, we propose two preference criteria on the solutions.

Definition 2 (Subset Minimality) *A solution S to the GTAP (T, C, H, M) is said to be subset minimal iff there is no proper subset $S' \subsetneq S$ such that S' is a solution. The set of all subset minimal solutions is denoted as $\mathcal{S}_{min}(T, C, H, M)$.*

Examples of subset minimal solutions for $\mathcal{P}_1$ in Figure 2 are {limb-joint $\dot{\sqsubseteq}$ joint} and {hinderlimb-joint $\dot{\sqsubseteq}$ joint, forelimb-joint $\dot{\sqsubseteq}$ joint}.

Assuming there exist solutions, the answer to the existence problem for subset-minimal solutions is yes, if and only if $T \cup H \models M$. To find a solution S, we can start from H, and remove the is-a relations h stepwise, such that $T \cup H \setminus h \models M$ holds. The process continues until no is-a relation can be removed. Thus if the entailment problem for the underlying ontology is tractable, finding a solution can be done in polynomial time. This is indeed the case for the is-a taxonomy.

The second criterion prefers solutions that imply more information.

Definition 3 (More Informative) *Let S and S' be two solutions to the GTAP (T, C, H, M). S is said to be* more informative *than S' iff $T \cup S \models T \cup S'$ and there exists a ψ such that $T \cup S \models \psi$ and $T \cup S' \not\models \psi$. Further, we say that S is* equally informative *as S' iff $T \cup S \models S'$ and $T \cup S' \models S$.*

Consider two solutions to $\mathcal{P}_1$ in Figure 2, S = {limb-joint $\dot{\sqsubseteq}$ joint} and S'={hinderlimb-joint $\dot{\sqsubseteq}$ joint, hand-joint $\dot{\sqsubseteq}$ joint}. S is more informative than S' as T $\cup$ S entails limb-joint $\dot{\sqsubseteq}$ joint in addition to everything that T $\cup$ S' entails.

Definition 4 (Semantic Maximality) *A solution S to the GTAP (T, C, H, M) is said to be semantically maximal iff there is no solution S' which is more informative than S. The set of all semantically maximal solutions is denoted as $\mathcal{S}^{max}(T, C, H, M)$.*

Analogous to the subset minimality, assuming the existence of GTAP solutions, the answer to the existence problem for a semantically maximal solution is yes, if and only if $T \cup H \models M$ holds. Moreover, in the case where $M \subseteq H$, and $T \cup H$ is consistent, H is a semantically maximal solution.

In practice, both of the above two criteria are desirable. However, only with the semantic maximality we might obtain a solution with redundancy. Although subset minimality does not yield redundancy, there is no guarantee that the solution is the most informative. In the following we propose definitions on solutions by combining these criteria. There are diverse interpretations for the combination of subset minimality and semantic maximality, depending on what kind of priority we assign for the single preferences. A first interpretation implies a higher priority on subset minimality than the semantic maximality. As the second interpretation, higher priority for semantic maximality can be assigned to subset minimality. In the third interpretation, the skyline-style interpretation, we treat both preferences equally and the chosen solution is such that there does not exist another solution which is preferable on both criteria.

Definition 5 (Combining with priority for subset minimality) *A solution S to the GTAP (T, C, H, M) is said to be minmax optimal iff S is subset minimal and there does not exist another subset minimal solution S' such that S' is more informative than S. The set of all minmax optimal solutions is denoted as $\mathcal{S}^{max}_{\mathbf{min}}(T, C, H, M)$.*

Lemma 1. $\mathcal{S}^{max}_{\mathbf{min}}(T, C, H, M) \subseteq \mathcal{S}_{min}(T, C, H, M)$

As an example, $\{$limb-joint $\dot{\sqsubseteq}$ joint$\}$ is a minmax optimal solution for $\mathcal{P}_1$, while $\{$hinderlimb-joint $\dot{\sqsubseteq}$ joint, forelimb-joint $\dot{\sqsubseteq}$ joint$\}$ is a minmax optimal solution for $\mathcal{P}_2$.

The existence problem is equivalent to the existence problem of the subset minimal solutions, i.e., there exists a subset minimal solution if and only if there exists a minmax optimal solution. On the other hand, finding a minmax optimal solution tends to be a harder problem. One naive method is first collecting all the subset minimal solutions, then removing those which are less informative. Obviously this is intractable, because theoretically there could be an exponential number of subset minimal solutions already.

In practice, minmax optimal solutions ensure fewer is-a relations to be added, thus avoiding redundancy. This is desirable if the domain expert would prefer to look at as small solutions as possible. The disadvantage is that there may be redundant relations that are correct and not be derivable when they are not added.

Definition 6 (Combining with priority for semantic maximality) *A solution S to the GTAP (T, C, H, M) is said to be maxmin optimal iff S is semantically maximal and there does not exist another semantically maximal solution S' such that S' is a proper subset of S. The set of all maxmin optimal solutions is denoted as $\mathcal{S}^{\mathbf{max}}_{min}(T, C, H, M)$.*

Lemma 2. $\mathcal{S}^{\mathbf{max}}_{min}(T, C, H, M) \subseteq \mathcal{S}^{max}(T, C, H, M)$

As an example, $\{$limb-joint $\dot{\sqsubseteq}$ joint, autopod-joint $\dot{\sqsubseteq}$ limb-joint$\}$ is a maxmin optimal solution for $\mathcal{P}_1$.

Analogous to the case of minmax optimal, the existence problem of maxmin optimal is equivalent to the existence problem of the semantic maximal solutions. Moreover, if H is a semantically maximal solution, finding a maxmin optimal solution S can be done by starting from H, and stepwise removing the is-a relations h such that $T \cup S \setminus h \models H$ holds. Intuitively, the goal is to remove the redundant relations in H. Of course there might be multiple maxmin optimal solutions in this regard, but finding one such a solution is tractable as long as the reasoning task for the underlying logic is tractable.

The advantage of the maxmin optimal semantics is that a maximal body of correct information is added to the ontology. If the domain expert would prefer to look at as informative solutions as possible without (set) redundancy, maxmin optimal solutions is preferable than the minmax optimal solutions. This conclusion can even be strengthened from the efficiency point of view, as finding a maxmin optimal solution is more efficient than finding a minmax optimal one. The disadvantage is that more relations need to be validated.

For the skyline interpretation, we consider the subset minimality and the semantic maximality as two dimensions for a solution S. S is skyline optimal if it is not dominated by any other solution. A solution dominates another solution if it is as good or better in all dimensions and better in at least one dimension. Therefore regarding the above two dimensions we define that a solution S dominates another solution S' if one of the following conditions is fulfilled:

1. $S \subsetneq S'$ and S is more informative than S', or
2. $S = S'$ and S is more informative than S', or
3. $S \subsetneq S'$ and S is equally informative as S'.

It is easy to verify that condition 1 and 2 can never be fulfilled, due to the monotonicity property of the entailment. Therefore, a solution S dominates another solution S' if and only if condition 3 is fulfilled. Accordingly, we have the definition for the skyline optimality as follows.

Definition 7 (Skyline optimal) *A solution S to the GTAP (T, C, H, M) is said to be skyline optimal iff there does not exist another solution S' such that S' is a proper subset of S and S' is equally informative as S. The set of all skyline optimal solutions is denoted as $\mathcal{S}^{max}_{min}(T, C, H, M)$.*

Skyline optimal is a relaxed criterion. It requires subset minimality for some level of informativeness. It comprises all the subset minimal solutions – which in turn comprises all the minmax optimal solutions – and all the maxmin optimal solutions. This relationship can be easily verified.

Lemma 3. $\mathcal{S}_{min}(T, C, H, M) \cup \mathcal{S}^{\mathbf{max}}_{min}(T, C, H, M) \subseteq \mathcal{S}^{max}_{min}(T, C, H, M)$.

As an example, M in Figure 2 is a skyline optimal solution for $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$ and $\mathcal{P}_4$. All previous examples for subset mininal, minmax optimal and maxmin optimal solutions are also skyline optimal solutions. However, there are semantically maximal solutions that are not skyline optimal. For instance, {hinderlimb-joint $\dot{\sqsubseteq}$ joint, forelimb-joint $\dot{\sqsubseteq}$ joint, hand-joint $\dot{\sqsubseteq}$ joint} is a semantically maximal solution for $\mathcal{P}_2$, but it is not skyline optimal as its subset {hinderlimb-joint $\dot{\sqsubseteq}$ joint, forelimb-joint $\dot{\sqsubseteq}$ joint} is equally informative.

4 Debugging in practice

4.1 General observations

A system for repairing the missing is-a structure in ontologies, takes as input the ontology T and a set of is-a relations to repair M. C is implicit and can be computed using

T. Further, the system should be used by a domain expert who validates is-a relations (H)[3]. In general, however, when starting a debugging session, we do not know the properties of T, M and H. Further, H represents the knowledge about is-a relations from the domain expert, but is normally not available beforehand, but only through interaction of the domain expert with the debugging system. This means that even in the situations where H is a solution, this does not readily provide us a solution in practice. It also means that one cannot just take subsets of H and check whether they are solutions.

Table 1 provides us with some guidelines for the development and the use of debugging systems. First, it is clear that we prefer an all-knowing expert. The second best case for obtaining correct solutions is the partial-correct expert. As discussed in Section 2, this could be approximated by using multiple domain experts and a skeptical approach.

If there are wrong is-a relations in M, there will be no solution or solutions that are not correct. The repaired ontology will contain incorrect is-a relations. Therefore, the expert should validate M at the beginning of the debugging session. Those is-a relations which are identified to be incorrect should be removed from M.[4] Another advantage of the validation is that, after validation we have that $M_{validated} \subseteq H$.

Further, as we do not know whether T is correct according to the domain or not, it should be checked whether $T \cup M_{validated}$ is consistent. If not, then there are no solutions. Otherwise, we know that $M_{validated}$ is a solution. When we remove the redundancy from $M_{validated}$, then we also have a subset minimal solution. This solution could then be used as a basis for finding more informative solutions. The difficulty is in finding subsets S of H (which is not available) such that $T \cup S$ is consistent.

4.2 Lessons for an existing system

The system in [13] allows debugging the is-a structure of and mappings between taxonomies in a taxonomy network. The input to the system is an ontology network. In this discussion we focus on one of the ontologies in the network (T and thus also C). The debugging workflow consists of three phases: (1) detection (generation of M), (2) validation of M and (3) repair (solving the GTAP problem). The domain expert is involved in the validation of M as well as in phase 3 for validation of possible solutions (S). The domain expert can switch between the different phases at any time. The system was used in a real case for the Swedish National Food Agency [12] and in several experiments with ontologies from the Ontology Alignment Evaluation Initiative [19].

Although the system allows to switch between the different phases, in all our experiments we started with validating M, which is as suggested by our analysis in Section 4.1. If M contained wrong is-a relations, we used semantic debugging techniques to repair these. This allowed us to remove incorrect is-a relations in T. When all the wrong is-a relations are repaired and removed from M, we obtain a new $M_{validated}$. If the domain expert validated M in a correct way, we are in a situation in the upper part of Table 1. The is-a relations in $M_{validated}$ are then repaired. When they are repaired using

[3] If there would be no expert, as shown in Table 1, in the best case M could be a correct solution, but there is no guarantee for solutions. We do not discuss this case further in this section.

[4] Depending on the detection method to generate M, the wrong is-a relations in M may lead to other debugging opportunities for semantic defects (e.g., [13]).

solutions that are more informative than $M_{validated}$, then new knowledge is added to the network and a new round of detection was started, possibly leading to the detection, validation and repair of new is-a relations.

Initially, $M_{validated}$ is added to the ontology. This means that we start with a least informative solution. When removing redundancy from $M_{validated}$, it is also a subset minimal solution. Then, the system tries to generate more informative solutions. For this, the missing is-a relations are repaired one at the time. For each missing is-a relation m_i a set of is-a relations R_i is computed that guarantees that $T \cup \{r_i\} \models m_i$ for each $r_i \in R_i$. Thus, for each missing is-a relation, at most one is-a relation is added to the ontology. By removing redundancy subset minimal solutions can be guaranteed. Further, for each missing is-a relation on its own semantically maximal solutions are generated with the extra conditions that only one is-a relation is used for repairing and no unnecessary equivalences ($\ll_{SH}$ in [21]) are introduced in the ontology.

One immediate consequence of our analysis is that we should allow a domain expert to choose several elements of each R_i. This is an easy extension to the system that would provide more informative solutions. Another consequence is that it would be advantageous to allow a domain expert to deal with a previously repaired is-a relation again, when new knowlegde was added to the ontology. New more informative solutions may be found. Further, there should be a way for domain experts to add new is-a relations that do not occur within the repairing process.

An interesting observation during the debugging described in [12] was that the domain experts changed their mind about the correctness of some is-a relations after debugging some other is-a relations. This means that H may actually change during a session, and we may move upwards in Table 1.

5 Related Work

Repairing missing is-a relations. There is not much work on the repairing of missing is-a structure. In [21, 20] this was addressed in the setting of taxonomies where the problem as well as some preference criteria were defined. Further, an algorithm was given for finding a solution to the repairing problem and an implemented system was proposed. A later version of that system was then used for debugging ontologies related to a project for the Swedish National Food Agency [12]. The system was further extended to deal with missing and wrong is-a relations and mappings [19] and integrated with ontology alignment [13]. In [18] the problem was formalized as an abduction problem and an algorithm was given for finding solutions for $\mathcal{ALC}$ acyclic terminologies.

TBox abduction. Except for [18] in which GTAP without H was defined, there is no other work yet on GTAP. There is some work on TBox abduction. [11] proposes an automata-based approach to TBox abduction using abducibles. It is based on a reduction to the axiom pinpointing problem which is then solved with automata-based methods.

Related topics. There is work that addresses related topics but not directly the problem that is addressed in this paper. Regarding *detecting missing is-a relations* there is much work on finding relationships between terms in the ontology learning area [2]. Further, there is work on finding is-a relations based on different kinds of patterns (e.g., [9, 4]). When the ontology is part of a network of ontologies connected by mappings,

knowledge itrinsic to the ontology network can be used to detect missing is-a relations using logical derivation [21, 12]. These approaches, in general, do not detect *all* missing is-a relations. There is much work on *debugging semantic defects*. Most of the work on debugging semantic defects aims at identifying and removing logical contradictions from an ontology (e.g., [8, 26, 16, 10, 24, 27, 1, 23]). In [22, 28, 25, 14, 15] the setting is extended to repairing ontologies connected by mappings. Further, there is some work on *abductive reasoning in description logics*. In [7] four different abductive reasoning tasks are defined - concept, ABox, TBox and knowledge base abduction. Concept abduction deals with finding sub-concepts. Abox abduction deals with retrieving instances that, when added to the knowledge base, allow the entailment of a desired ABox assertion. Knowledge base abduction includes both ABox and TBox abduction. Most existing approaches focus on ABox [17, 6] and concept abduction [3, 5].

6 Conclusion

In this paper we formalized repairing missing is-a structure in ontologies as an abduction problem. We defined properties for the ontology, the set of is-a relations to repair and the domain expert, as well as preference criteria on solutions and discussed the influences of these properties and criteria on the existence of solutions for the abductive problem. We also discussed the consequences of our analyses for the development and use of debugging systems. One direction for future work is to analyze the complexity of the decision problems for different knowledge representation languages. Further, we want to investigate in algorithms that satisfy the preference criteria for different languages and that can be used in practice in a debugging system.

References

1. 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.
2. Ph Cimiano, P Buitelaar, and B Magnini. *Ontology Learning from Text: Methods, Evaluation and Applications*. IOS Press, 2005.
3. S Colucci, T Di Noia, E Di Sciascio, F Donini, and M Mongiello. A uniform tableaux-based approach to concept abduction and contraction in ALN. In *International Workshop on Description Logics*, pages 158–167, 2004.
4. O Corcho, C Roussey, L M Vilches, and I Pérez. Pattern-based OWL ontology debugging guidelines. In *Workshop on Ontology Patterns*, pages 68–82, 2009.
5. F Donini, S Colucci, T Di Noia, and E Di Sciasco. A tableaux-based method for computing least common subsumers for expressive description logics. In *21st International Joint Conference on Artificial Intelligence*, pages 739–745, 2009.
6. J Du, G Qiand Y-D Shen, and J Pan. Towards practical Abox abduction in large OWL DL ontologies. In *25th AAAI Conference on Artificial Intelligence*, pages 1160–1165, 2011.
7. C Elsenbroich, O Kutz, and U Sattler. A case for abductive reasoning over ontologies. In *OWL: Experiences and Directions*, 2006.
8. P Haase and L Stojanovic. Consistent Evolution of OWL Ontologies. In *2nd European Semantic Web Conference*, pages 182–197. 2005.

9. M Hearst. Automatic acquisition of hyponyms from large text corpora. In *14th International Conference on Computational Linguistics*, pages 539–545, 1992.
10. M Horridge, B Parsia, and U Sattler. Laconic and precise justifications in OWL. In *7th International Semantic Web Conference*, pages 323–338, 2008.
11. T Hubauer, S Lamparter, and M Pirker. Automata-based abduction for tractable diagnosis. In *International Workshop on Description Logics*, pages 360–371, 2010.
12. 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.
13. V Ivanova and P Lambrix. A unified approach for aligning taxonomies and debugging taxonomies and their alignments. In *10th Extended Semantic Web Conference*, pages 1–15, 2013.
14. 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.
15. E Jimenez-Ruiz, B Cuenca Grau, I Horrocks, and R Berlanga. Ontology Integration Using Mappings: Towards Getting the Right Logical Consequences. In *6th European Semantic Web Conference*, pages 173–187, 2009.
16. A Kalyanpur, B Parsia, E Sirin, and J Hendler. Debugging Unsatisfiable Classes in OWL Ontologies. *Journal of Web Semantics*, 3(4):268–293, 2006.
17. S Klarman, U Endriss, and S Schlobach. Abox abduction in the description logic ALC. *Journal of Automated Reasoning*, 46:43–80, 2011.
18. P Lambrix, Z Dragisic, and V Ivanova. Get my pizza right: Repairing missing is-a relations in ALC ontologies. In *2nd Joint International Semantic Technology Conference*, pages 17–32, 2012.
19. P Lambrix and V Ivanova. A unified approach for debugging is-a structure and mappings in networked taxonomies. *Journal of Biomedical Semantics*, 4:10, 2013.
20. P Lambrix and Q Liu. Debugging the missing is-a structure within taxonomies networked by partial reference alignments. *Data & Knowledge Engineering*, 2013.
21. P Lambrix, Q Liu, and H Tan. Repairing the Missing is-a Structure of Ontologies. In *4th Asian Semantic Web Conference*, pages 76–90, 2009.
22. C Meilicke, H Stuckenschmidt, and A Tamilin. Repairing Ontology Mappings. In *22th National Conference on Artificial Intelligence*, pages 1408–1413, 2007.
23. 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.
24. 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.
25. G Qi, Q Ji, and P Haase. A Conflict-Based Operator for Mapping Revision. In *8th International Semantic Web Conference*, pages 521–536, 2009.
26. S Schlobach. Debugging and Semantic Clarification by Pinpointing. In *2nd European Semantic Web Conference*, pages 226–240, 2005.
27. 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.
28. P Wang and B Xu. Debugging ontology mappings: a static approach. *Computing and Informatics*, 27:21–36, 2008.