



Repairing Networks of \mathcal{EL}_\perp Ontologies Using Weakening and Completing

Ying Li^{1,2(✉)}  and Patrick Lambrix^{1,2(✉)} 

¹ Linköping University, Linköping, Sweden
{Ying.Li,Patrick.Lambrix}@liu.se

² The Swedish e-Science Research Centre, Linköping, Sweden

Abstract. The quality of ontologies and their alignments is crucial for developing high-quality semantics-based applications. Traditional debugging techniques repair ontology networks by removing unwanted axioms and mappings, but may thereby remove consequences that are correct in the domain of the ontology network. In this paper we propose a framework for repairing ontology networks that deals with this issue. It defines basic operations such as debugging, weakening and completing. Further, it defines combination operators that reflect choices in how and when to use the basic operators, as well as choices regarding the autonomy level of the ontologies and alignments in the ontology network. We show the influence of the combination operators on the quality of the repaired network and present an implemented tool. By using our framework together with existing algorithms for debugging, weakening and completing, we essentially provide a blueprint for extending previous work and systems.

1 Introduction

Ontologies and ontology networks, i.e., a set of ontologies connected through alignments, are core components in application scenarios that involve searching, integrating, managing and extracting value from diverse sources of data at large scale. In the latter case they are the input for machine learning and data mining applications in diverse fields such as business analytics, health, crime analysis, and materials design. They are also the structural part of knowledge graphs which are used by, e.g., major data and database providers such as Google, Amazon, Meta, and Neo4j. The quality of ontologies, ontology networks and knowledge graphs is crucial for developing high-quality semantics-based applications. However, ensuring their quality, in particular regarding completeness (or coverage, all relevant information is modeled) and correctness (no wrong information is modeled), is a major challenge [4, 9, 18, 31, 32].

Repairing ontology networks is a natural requirement for any application that would need to use several ontologies. We have, for instance, experience in a number of settings where an ontology network is repaired. A first example is the case of ontologies in the Ontology Alignment Evaluation Initiative (OAEI) where the ontologies to be aligned and a partial alignment were used to detect and repair defects [21] (and the results were sent to the ontology owners). Another example is the development of modular ontologies where the modules and connections

between the modules have been repaired (e.g., [19]). Further, we have worked with companies that used publicly available ontologies as well as concern-wide ontologies and where the network of these were repaired (e.g., [10]).

The workflow for dealing with unwanted axioms in ontologies or networks consists of several steps including the detection and localization of the defects and the repairing. In this paper we assume we have detected defects and we now need to repair the ontologies and networks. In the classical approaches the end result is a set of axioms to remove from the ontology or network that is obtained after detection and localization, and the repairing consists solely of removing the suggested axioms. However, more knowledge than needed may be removed and approaches that alleviate the effect of removing unwanted axioms using weakening and completing have been proposed. In [24] a framework for repairing \mathcal{EL} ontologies, based on the basic operations removing, weakening and completing, was proposed. Further, different combination operators were introduced that relate to choices that can be made when combining the basic operations (e.g., in which order to perform the operations, and when to update the ontologies). It was shown that the choice of combination has an influence on the amount of validation work by a domain expert and the completeness of the final ontology. It was also shown that earlier work on weakening (without completing) only considered one of the possible combinations. Similarly, earlier work on completing (without weakening) also considered only one of the possible combinations.

Our contributions in this paper relate to different ways in which we extend this framework. (i) A first extension is that we add the basic operation of debugging into the framework of [24]. We introduce combination operators for debugging and show the influence of the choice with respect to completeness, incorrectness and validation work (Sect. 2.3). (ii) As our first main contribution, we extend the framework in [24] to deal with ontology *networks*, formalize the problem, and show that new choices appear in addition to the choices for ontologies, when ontologies are connected with alignments (Sect. 3) (iii) As our second main contribution, we introduce different levels of autonomy for the ontologies and the alignments in the ontology network, reflecting the policies of the ontology and alignment owners regarding updating and computing for their ontologies and alignments (Sect. 3). We define combination operators based on these autonomy levels and show the influences on the quality of the repair and the validation work (Sect. 4). In general, there is a trade-off between the quality of the repaired ontologies on the one hand, and the amount of validation work for the oracle (e.g., domain experts) and the autonomy level for the ontologies and alignments on the other hand. We also give recommendations about which combinations to use in different cases, and we show that current systems for alignment repair (a special case of ontology network repair) use one particular combination of choices. In Sect. 5 we give examples of repairing existing ontology networks and (iv) in Sect. 6 we briefly describe an implemented tool that allows users to make different choices for the combination operators.

The only other work that discusses the combination of basic operations for repairing is [24] where the focus is on ontologies and the basic operations

are removing, weakening and completing. For the basic operations different approaches can be used. As this paper is not about possible approaches for the basic operations, but rather about their combinations, we mention these approaches in Sect. 2.3 and refer for an overview that compares these approaches to [18]. This paper essentially treats the basic operations of debugging, weakening and completing as black boxes and does not put requirements on the actual implementation of these operations. Therefore, by using our framework and the combination operators together with existing algorithms for debugging, weakening and completing, we essentially provide a blueprint for extending previous work and systems.

Table 1. \mathcal{EL}_\perp syntax and semantics.

Name	Syntax	Semantics
top	\top	$\Delta^{\mathcal{I}}$
bottom	\perp	\emptyset
conjunction	$P \sqcap Q$	$P^{\mathcal{I}} \cap Q^{\mathcal{I}}$
existential restriction	$\exists r.P$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x, y) \in r^{\mathcal{I}} \wedge y \in P^{\mathcal{I}}\}$
GCI	$P \sqsubseteq Q$	$P^{\mathcal{I}} \subseteq Q^{\mathcal{I}}$

2 Preliminaries

2.1 Description Logics

Description logics (DL) [2] are knowledge representation languages where concept descriptions are constructed inductively from a set N_C of atomic concepts and a set N_R of atomic roles and (possibly) a set N_I of individual names. Different DLs allow for different constructors for defining complex concepts and roles. An interpretation \mathcal{I} consists of a non-empty set $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ which assigns to each atomic concept $P \in N_C$ a subset $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, to each atomic role $r \in N_R$ a relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and to each individual name¹ $i \in N_I$ an element $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The interpretation function is straightforwardly extended to complex concepts. A TBox is a finite set of axioms which in \mathcal{EL}_\perp are *general concept inclusions* (GCIs). The syntax and semantics for \mathcal{EL}_\perp are shown in Table 1.

An interpretation \mathcal{I} is a *model* of a TBox \mathcal{T} if for each GCI in \mathcal{T} , the semantic conditions are satisfied. We say that a TBox \mathcal{T} is *inconsistent* if there is no model for \mathcal{T} . Further, a concept P in a TBox \mathcal{T} is *unsatisfiable* if for all models \mathcal{I} of \mathcal{T} :

¹ As we do not deal with individuals in this paper, we do not use individuals in the later sections.

$P^{\mathcal{I}} = \emptyset$. We say that a TBox is *incoherent* if it contains an unsatisfiable concept. One of the main reasoning tasks for DLs is subsumption checking² in which the problem is to decide for a TBox \mathcal{T} and concepts P and Q whether $\mathcal{T} \models P \sqsubseteq Q$, i.e., whether $P^{\mathcal{I}} \subseteq Q^{\mathcal{I}}$ for every model \mathcal{I} of TBox \mathcal{T} . In this paper we update the TBox during the repairing and we always use subsumption with respect to the current TBox.

2.2 Ontology Networks

In this paper we assume that ontologies are represented using DL TBoxes. An alignment between two ontologies is a set of mappings between the ontologies. A mapping between two ontologies is represented by $P \sqsubseteq Q$ where P is a concept in the first ontology and Q is a concept in the second ontology. Note that equivalence mappings (e.g., P is equivalent to Q) are represented by two subsumption mappings ($P \sqsubseteq Q$ and $Q \sqsubseteq P$). Although we base our work and examples on \mathcal{EL}_{\perp} , the discussions hold for ontologies represented by DLs in general. An ontology network is a collection of ontologies together with their alignments and can be represented by a TBox (Definition 1).³ In the remainder we use the term axiom for the axioms in the ontologies and the mappings. When we mean axioms in the ontologies, we will explicitly state this.

Definition 1. Let $\mathcal{T}_1, \dots, \mathcal{T}_n$ be TBoxes representing ontologies $\mathcal{O}_1, \dots, \mathcal{O}_n$, respectively. For $i, j \in [1..n]$ with $i < j$, let \mathcal{A}_{ij} be an alignment between ontology \mathcal{O}_i and \mathcal{O}_j . The network of the ontologies and their alignments is then represented by TBox $\mathcal{T} = (\bigcup_{i=1..n} \mathcal{T}_i) \cup (\bigcup_{i,j=1..n, i < j} \mathcal{A}_{ij})$.

Our aim is to find repairs that remove as much wrong knowledge and add as much correct knowledge (back) to our ontology network as possible. Therefore, we use the preference relations *more complete* and *less incorrect* between TBoxes (Definitions 2 and 3) that formalize these intuitions [18]. Further, if a TBox representing an ontology (network) is more complete/less incorrect than the Tbox of an other ontology (network), then we say that the first ontology (network) is more complete/less incorrect than the second ontology (network). The definitions assume the existence of an oracle (representing a domain expert) that, when given an axiom, can answer whether this axiom is correct or wrong in the domain of interest of the ontology network.

Definition 2 (more complete). TBox \mathcal{T}_1 is *more complete* than TBox \mathcal{T}_2 (or \mathcal{T}_2 is *less complete* than \mathcal{T}_1) according to oracle Or iff $(\forall \psi : (\mathcal{T}_2 \models \psi \wedge Or(\psi) = true) \rightarrow \mathcal{T}_1 \models \psi) \wedge (\exists \psi : Or(\psi) = true \wedge \mathcal{T}_1 \models \psi \wedge \mathcal{T}_2 \not\models \psi)$. They are *equally complete* iff $\forall \psi : Or(\psi) = true \rightarrow (\mathcal{T}_1 \models \psi \leftrightarrow \mathcal{T}_2 \models \psi)$

² Note that unsatisfiability checking in \mathcal{EL}_{\perp} can be performed using subsumption checking. A concept P is unsatisfiable if $P \sqsubseteq \perp$. Further, we can express that two concepts P and Q are disjoint by requiring that $P \sqcap Q \sqsubseteq \perp$.

³ We do not require that the ontologies in the network have the same signature.

Definition 3 (less incorrect). TBox \mathcal{T}_1 is *less incorrect* than TBox \mathcal{T}_2 (\mathcal{T}_2 is *more incorrect* than \mathcal{T}_1) according to oracle Or iff $(\forall\psi : (\mathcal{T}_1 \models \psi \wedge Or(\psi) = false) \rightarrow \mathcal{T}_2 \models \psi) \wedge (\exists\psi : Or(\psi) = false \wedge \mathcal{T}_1 \not\models \psi \wedge \mathcal{T}_2 \models \psi)$. \mathcal{T}_1 and \mathcal{T}_2 are *equally incorrect* iff $\forall\psi : Or(\psi) = false \rightarrow (\mathcal{T}_1 \models \psi \leftrightarrow \mathcal{T}_2 \models \psi)$.

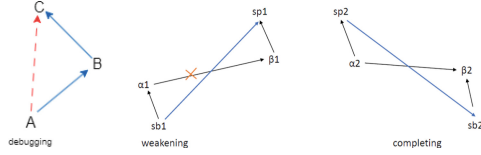
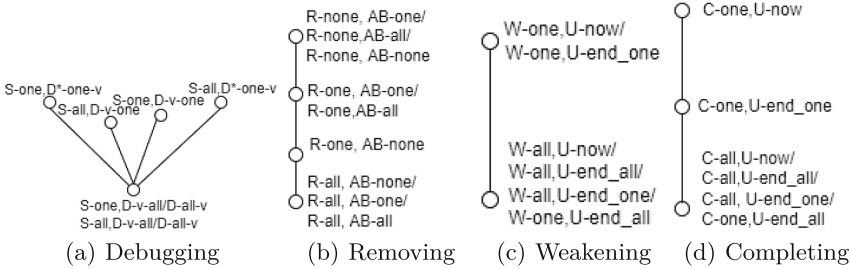
2.3 Basic Operations

To repair ontologies, algorithms can be developed using a number of basic operations such as removing, weakening and completing, and combining these in different ways [24]. In this paper we add a basic operation, debugging, and use variants of these operations to repair ontology networks.

Given a set of wrong axioms W , *debugging* aims to find a set of wrong **asserted** axioms D that when all axioms in D are removed from the network, the axioms in W cannot be derived anymore. Many debugging approaches have been proposed (e.g., [1, 7, 11, 15–17, 23, 27–30, 35, 37–41], overview in [18])⁴. A basic approach is based on the computation of justifications for the wrong axioms and then computing a Hitting set over the set of justifications. As an example, in Fig. 1 derived wrong axiom $A \sqsubseteq C$ needs to be removed. This can be done by removing asserted axiom $A \sqsubseteq B$ or asserted axiom $B \sqsubseteq C$. *Removing* deletes all the wrong asserted axioms in a given set D from the ontology network. Removing makes a network less or equally incorrect than it was before the operation. Given an axiom, *weakening* aims to find other axioms that are weaker than the given axiom, i.e., the given axiom logically implies the other axioms within the network. For the repairing this means that a wrong axiom $\alpha \sqsubseteq \beta$ can be replaced by a correct weaker axiom $sb \sqsubseteq sp$ such that sb is a sub-concept of α and sp is a super-concept of β , thereby mitigating the effect of removing the wrong axiom (Fig. 1). Algorithms for weakening have been provided in e.g., [3, 5, 24, 42]. *Completing* aims to find correct axioms that are not derivable from the ontology yet and that would make a given axiom derivable. For a given axiom $\alpha \sqsubseteq \beta$, it finds correct axioms $sp \sqsubseteq sb$ such that sp is a super-concept of α and sb is a sub-concept of β (Fig. 1). This means that if $sp \sqsubseteq sb$ is added to \mathcal{T} , then $\alpha \sqsubseteq \beta$ would be derivable. Completing is performed on correct axioms, and in repairing, it is applied to weakened axioms. Completing algorithms are proposed in, e.g., [6, 8, 20, 24, 43]. Note that weakening and completing are dual operations where the former finds weaker axioms and the latter stronger axioms. Both these operations make an ontology network more or equally complete.

These basic operations can be combined in different ways and there are choices to be made in terms of, e.g., the order in which the operations are performed, the order in which the axioms are processed, whether one axiom is dealt with at a time or all at once, and when the TBox is updated. For removing, weakening and completing, the different combinations were classified in Hasse diagrams [24], and here we add a Hasse diagram for debugging (Fig. 2). Explanations for removing, weakening and completing are from [24] and are given in the

⁴ Many of these approaches use axiom pinpointing [33]. Further, there are other approaches that take ABoxes into account.

**Fig. 1.** Debugging, weakening and completing.**Fig. 2.** Hasse diagrams (b-d from [24]). (a) selecting and debugging; (b) removing and adding back wrong axioms; (c) weakening and updating; (d) completing and updating.

supplemental material. For debugging, we introduce the combination operators S-one, S-all, D-one-v, D-v-one and D-all-v/D-v-all. S-one and S-all represent the choices to calculate the justifications for one wrong axiom at the time or for all at once, respectively. The operations with a name starting with D concern choices regarding using the justifications to generate asserted wrong axioms to remove from the ontology. When using D-one-v, one Hitting set is generated from the justifications, and then the asserted axioms in the generated Hitting set are validated. With D-v-one, one valid Hitting set in the justifications is validated. Using D-all-v/D-v-all, all asserted axioms from the justifications are validated. Note that using D-one-v may not guarantee that a repair is found as axioms in the Hitting set may be correct and therefore should not be removed. The proof of the Hasse diagram for debugging is given in the supplemental material.⁵

In general, operations higher up in the diagrams use more (but also more possibly wrong) information during the computations and lead to more (or equally) complete ontologies, more (or equally) incorrect ontologies as well as more validation work for the domain expert. Algorithms for repairing can be defined by which of the operators are used and in which order. These algorithms can then be compared using the Hasse diagrams. In general, if the sequence of operators for one algorithm can be transformed into the sequence of operators of a second

⁵ Note that the discussions and proofs for all Hasse diagrams in this paper hold for ontologies and ontology networks represented by DL TBoxes in general, and thus the results regarding the combination operators are language agnostic. When the framework is instantiated to create actual systems, then choices need to be made regarding which algorithms to use for the different basic operations, and the choice of these algorithms will, of course, also influence the completeness and correctness of an ontology or ontology network (but this is not the topic of this paper).

algorithm, by replacing some operators of the first algorithm using operators higher up in the Hasse diagrams in Fig. 2, then the ontologies repaired using the second algorithm are more or equally complete and incorrect than the ontologies repaired using the first algorithm. The work in [24] was the first to identify these different combinations. It also showed that previous work on combining removing with weakening used the combination (R-one, AB-none) with (W-one, U-now). Previous work on completing used (C-one, U-end-all). There was no work that used other combinations and no work that combined all basic operations.

Algorithm 1. Generate the justifications of each wrong axiom and validate all wrong asserted axioms from the generated justifications, weaken all wrong asserted axioms, complete all weakened axioms, add all completed axioms and remove all wrong asserted axioms at the end (S-one, D-v-all/R-none, AB-none/W-all, U-end_all/C-all, U-end_all).

Input: TBox \mathcal{T} , Oracle Or , set of unwanted axioms W

Output: A repaired TBox

```

1:  $D \leftarrow \emptyset$ 
2: for each  $\alpha \sqsubseteq \beta \in W$  do
3:   for each  $J \in \text{Justifications}(\alpha \sqsubseteq \beta)$  do
4:     for each  $axiom \in J$  do
5:       if  $\text{Validate\_axiom}(axiom, Or) = \text{false}$  then
6:          $D \leftarrow D \cup \{axiom\}$ 
7:       end if
8:     end for
9:   end for
10: end for
11: for each  $\alpha \sqsubseteq \beta \in D$  do
12:    $w_{\alpha \sqsubseteq \beta} \leftarrow \text{weakened-axiom-set}(\alpha \sqsubseteq \beta, \mathcal{T}, Or)$ 
13: end for
14:  $c_{\alpha \sqsubseteq \beta} \leftarrow \emptyset$ 
15: for each  $\alpha \sqsubseteq \beta \in D$  do
16:   for each  $sb \sqsubseteq sp \in w_{\alpha \sqsubseteq \beta}$  do
17:      $c_{sb \sqsubseteq sp} \leftarrow \text{completed-axiom-set}(sb \sqsubseteq sp, \mathcal{T}, Or)$ 
18:    $c_{\alpha \sqsubseteq \beta} \leftarrow c_{\alpha \sqsubseteq \beta} \cup c_{sb \sqsubseteq sp}$ 
19:   end for
20: end for
21:  $\mathcal{T}_r \leftarrow \text{Add-axioms}(\mathcal{T}, \bigcup_{\alpha \sqsubseteq \beta} c_{\alpha \sqsubseteq \beta})$ 
22:  $\mathcal{T}_r \leftarrow \text{Remove-axioms}(\mathcal{T}_r, D)$ 
23: return  $\mathcal{T}_r$ 

```

For the examples in this paper we have used Algorithm 1 as a basis for discussing repairing of ontology networks. It uses a particular combination of debugging, removing, weakening and completing with specific algorithms for the basic operations. Our discussion regarding the choices for *ontology networks* (Sect. 3) would still hold if we used other algorithms for the basic operations, or different combination operators for the basic operators.

3 Repairing Ontology Networks Problem Definition

In this section we define the repairing problem for ontology networks. Definition 4 is an extension of the definition of repair for single ontologies as defined in [24]. We are given a set of wrong axioms W that we want to remove from the ontology network, and when they are removed, they cannot be derived from the TBox representing the ontology network anymore. These axioms in W can be axioms in the ontologies or mappings. Further, to guarantee a high level of quality of the ontology (i.e., so that no correct information is removed or no incorrect information is added), domain expert validation is a necessity (e.g., [24, 34]). Therefore, we assume an oracle (representing a domain expert) that, when given an axiom, can answer whether this axiom is correct or wrong in the domain of interest of the ontology network. A repair (A, D) for the ontology network given the TBox \mathcal{T} , oracle Or , and a set of wrong axioms W , is a tuple containing two sets: a set A of axioms that are correct according to the oracle and should be added to the TBox, and a set D of asserted axioms that are not correct according to the oracle and should be removed from the TBox. We require that when the axioms in D are removed and the axioms in A are added, the wrong axioms in W cannot be derived anymore.

Definition 4 (Repair). Let TBox $\mathcal{T} = (\bigcup_{i=1..n} \mathcal{T}_i) \cup (\bigcup_{i,j=1..n, i < j} \mathcal{A}_{ij})$ represent a network of ontologies \mathcal{O}_i represented by TBoxes \mathcal{T}_i , and their alignments \mathcal{A}_{ij} . Let Or be an oracle that given a TBox axiom returns true or false. Let W be a finite set of TBox axioms in \mathcal{T} such that $\forall \psi \in W: Or(\psi) = \text{false}$. Then, a repair for Debug-Problem $DP(\mathcal{T}, Or, W)$ is a tuple (A, D) where A and D are finite sets of TBox axioms such that

- (i) $\forall \psi \in A: Or(\psi) = \text{true}$;
- (ii) D is a finite set of *asserted* axioms in \mathcal{T} ;
- (iii) $\forall \psi \in D: Or(\psi) = \text{false}$;
- (iv) $\forall \psi \in W: (\mathcal{T} \cup A) \setminus D \not\models \psi$.

From a theoretical point of view, as we have represented the ontology network as a TBox, and previous work has represented ontologies as TBoxes, we could reuse the algorithms and approaches in, e.g., [24]. However, from a practical point of view, the situation is more complex. When working with single ontologies, it is in the interest and mandate of the ontology owners to repair their ontologies. However, when ontologies are connected in a network, the ontology and alignment owners may want to retain different levels of autonomy and not necessarily allow others to change or propose changes to their ontologies and alignments. Also, computation time and validation work for repairing may be issues. In this case, computation time and validation work may be lower within an ontology or alignment than for the whole network. In this paper we discuss three levels of autonomy and show the influence of these different choices.

The first level consists of the cases ‘O’ (ontology) and ‘M’ (mappings). ‘O’ represents the choice where ontologies are completely autonomous. Essentially, this means that ontologies act on their own regarding detection and repairing of defects. W in Definition 4 contains only axioms in the ontology, only the

axioms within the ontology itself can be used for the computation of repairs, and solutions can only include axioms in the ontology itself. A dual case is ‘M’ where the owners of an alignment are autonomous. In this case W contains only mappings, and the alignment is repaired using only the mappings in the alignment. The second level consists of the cases ‘MO’ (materialized ontology) and ‘MM’ (materialized mappings). ‘MO’ uses the network to derive new axioms within the ontology and materializes these axioms.⁶ W contains only axioms in the ontology. For the computation of repairs the materialized ontology is used, and solutions contain only axioms within the ontology. This level accepts the fact that the network provides more knowledge about the own ontology than is represented by the ontology itself, and accepts this knowledge, but it does not use the network in repairing. ‘MM’, the dual case for mappings, computes derived mappings for the alignment, but then acts autonomously. W contains only mappings. The third level ‘ON’ (ontology network) considers the ontologies and alignments as integral parts of the network, uses the full network for the computation of repairs, and repairs defects using axioms within all ontologies and alignments. W can contain ontology axioms and mappings. We note that different ontology and alignment owners may make different choices regarding their level of autonomy.

These choices have an influence on the detection and repairing of effects. We give an example for the detection here, and focus on the repairing in further sections. For the network in Fig. 3 we have the following situation. Using the full network (level ‘ON’), we can derive that concepts E , F , e , and f are unsatisfiable.⁷ Using level ‘MO’ for both ontologies, we materialize $E \sqsubseteq D$ and $F \sqsubseteq E$ in the first ontology, and $e \sqcap b \sqsubseteq \perp$ in the second ontology. This allows us to obtain the same unsatisfiable concepts as in the ‘ON’ level (although the repairing will be different). When using level ‘O’ for both ontologies, we cannot detect any unsatisfiable concepts.

In general, there is a trade-off between the quality of the repaired ontologies on the one hand, and the amount of validation work for the oracle and the autonomy level for the ontologies and alignments on the other hand. In the next sections we define and exemplify these notions.

4 Repairing Ontology Networks Combination Operators

During the repairing process different levels of autonomy can be used at different stages. The choice has an influence on the quality of the repair. Here we show this influence for different stages.

⁶ We note that a choice may be made regarding which axioms to materialize. For taxonomies it may be feasible to materialize all. However, for \mathcal{EL}_\perp there is an infinite number of derivable axioms. In our experiments we restrict the space to axioms with concepts at the left- and right-hand side with non-nested operators (SCC in [24]).

⁷ E unsatisfiable because $D \sqcap E \sqsubseteq \perp$, $E \sqsubseteq e$, $e \sqsubseteq b$, $b \sqsubseteq D$. e unsatisfiable because E unsatisfiable and $e \sqsubseteq E$. f unsatisfiable because e unsatisfiable and $f \sqsubseteq e$. F unsatisfiable because f unsatisfiable and $F \sqsubseteq f$.

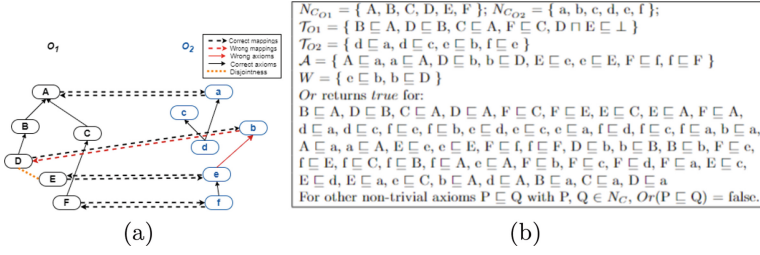


Fig. 3. The network contains unsatisfiable concepts E, F, e, and f. Asserted axiom $e \sqsubseteq b$ in O_2 , and mapping $b \sqsubseteq D$ between O_2 and O_1 lead to network incoherence.

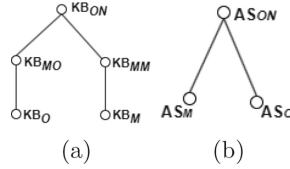


Fig. 4. Hasse diagrams regarding (a) the use of different knowledge bases during repairing (KB); (b) different restrictions on the add set (AS);

4.1 Use of Background

As discussed earlier, we can use different autonomy levels during the computation of repairs. In our algorithms, this is reflected by which TBox is used. We can combine particular choices for debugging, removing, weakening and completing algorithms with different autonomy levels. As an example, assume we have made the choices for the basic operations as in Algorithm 1. Then, for autonomy level ‘ON’ we can use Algorithm 1 with as input the TBox representing the whole network. For levels ‘O’ and ‘MO’ we can use Algorithm 1 repeatedly using each of the TBoxes of the ontologies or materialized ontologies, respectively. (Another possibility is to use as input the union of these TBoxes, i.e., there will be no axioms representing mappings in the network.) For ‘M’ and ‘MM’ we use the alignments or materialized alignments as TBoxes. The choices are explained in Table 2 where KB stands for ‘knowledge base’. Note that the choices can be performed on each of the basic operations. For instance, it is possible to debug using ‘ON’ and then weaken and complete using ‘MO’. We can organize the choices in a Hasse diagram as in Fig. 4(a). The proofs for the Hasse diagram are given in the supplemental material. In general, using more background knowledge leads to more (or equally) complete and more (or equally) incorrect networks, and more validation work by the oracle.

As an example, assume we have used ‘ON’ in the debugging step where we validate all axioms in the justifications, for the network in Fig. 3. This results in wrong axioms $e \sqsubseteq b$ and $b \sqsubseteq D$. We now look at the weakening step for the different autonomy levels. When weakening $e \sqsubseteq b$ in the second ontology, we use sub-concepts of e and super-concepts of b to find weakened axioms. These sets

Table 2. Choices for background knowledge (KB) and add sets (AS).

Choices	Description
KB_{ON}	Use the whole ontology network as the background knowledge base to compute the sub/sup-concepts when weakening and completing an axiom.
KB_{MO}	Materialize each ontology in the network. Disconnect each ontology from the ontology network. Use only the axioms within the materialized ontologies to compute the sub/sup-concepts when weakening and completing an axiom.
KB_O	Disconnect each ontology from the ontology network. Use the axioms within the respective ontologies to compute the sub/sup-concepts when weakening and completing an axiom.
KB_{MM}	Materialize each alignment in the network. Disconnect each alignment from the ontology network. Use only the mappings within the materialized alignments to compute the sub/sup-concepts when weakening and completing an axiom.
KB_M	Disconnect each alignment from the ontology network. Use the mappings within the respective alignments to compute the sub/sup-concepts when weakening and completing an axiom.
AS_{ON}	Add all axioms.
AS_O	Add only axioms within the respective ontologies.
AS_M	Add only mappings between the ontologies.

are different for different autonomy levels. We have $\text{Sub}(e, KB_O) = \{e, f\}$, $\text{Sub}(e, KB_{MO}) = \{e, f\}$, $\text{Sub}(e, KB_{ON}) = \{e, f, E, F\}$, and $\text{Sup}(b, KB_O) = \{b\}$, $\text{Sup}(b, KB_{MO}) = \{b, a\}$, $\text{Sup}(b, KB_{ON}) = \{b, a, D, B, A\}$. Therefore, there are 2, 4 and 20 candidates for the weakened axioms for ‘O’, ‘MO’ and ‘ON’, respectively. After validation we have as weakened axioms $f \sqsubseteq b$ for ‘O’, $f \sqsubseteq b$ and $e \sqsubseteq a$ for ‘MO’, and $f \sqsubseteq b$, $e \sqsubseteq a$, and $E \sqsubseteq A$ for ‘ON’ (Table 3). As expected from the Hasse diagram, ‘ON’ leads to the most complete network, followed by ‘MO’ and then ‘O’. When weakening the wrong mapping $b \sqsubseteq D$, we can choose between ‘M’, ‘MM’ and ‘ON’. We have $\text{Sub}(b, KB_M) = \{b, D\}$, $\text{Sub}(b, KB_{MM}) = \text{Sub}(b, KB_{ON}) = \{e, f, b, D, E, F\}$, $\text{Sup}(D, KB_M) = \{D, b\}$, and $\text{Sup}(D, KB_{MM}) = \text{Sup}(D, KB_{ON}) = \{D, A, B, b, a\}$.⁸ For ‘M’ there are no weakened axioms. For ‘MM’ we find $b \sqsubseteq B$. For ‘ON’ we find $b \sqsubseteq B$ and $e \sqsubseteq a$. The latter includes both mappings and ontology axioms. Table 3 also shows the results after completing the weakened axioms using the same autonomy level as for weakening. The network is repaired by removing the wrong axioms and adding the completed axioms.

⁸ All algorithms for debugging, weakening and completing use heuristics when computing the sub-concepts and super-concepts to prune the search space of solutions [18, 24]. Also for the networks different heuristics can be used. For instance, when computing the sub- and super-concepts for the concepts in mappings, one heuristic is to only allow sub- and super-concepts in the same ontology. In the example we have allowed sub- and super-concepts to belong to other ontologies as long as they are involved in mappings.

Table 3. Weakening and completing the wrong axioms $e \sqsubseteq b$ and $b \sqsubseteq D$.

Background knowledge base	KB_O $e \sqsubseteq b$	KB_{MO} $e \sqsubseteq b$	KB_{ON} $e \sqsubseteq b$	KB_M $b \sqsubseteq D$	KB_{MM} $b \sqsubseteq D$	KB_{ON} $b \sqsubseteq D$
$ \text{Sub}(\alpha, \mathcal{T}) $	2	2	4	2	6	6
$ \text{Sup}(\beta, \mathcal{T}) $	1	2	5	2	5	5
Weakened	$f \sqsubseteq b$	$f \sqsubseteq b, e \sqsubseteq a$	$f \sqsubseteq b, e \sqsubseteq a, E \sqsubseteq A$		$b \sqsubseteq B$	$b \sqsubseteq B, e \sqsubseteq a$
$ \text{Sup}(\alpha, \mathcal{T}) $	2	4 3	10 7 7		5	5 7
$ \text{Sub}(\beta, \mathcal{T}) $	1	3 5	6 11 11		7	7 11
Completed	$f \sqsubseteq b$	$f \sqsubseteq b, e \sqsubseteq d$	$f \sqsubseteq b, e \sqsubseteq d, E \sqsubseteq C$		$b \sqsubseteq B, B \sqsubseteq b$	$b \sqsubseteq B, B \sqsubseteq b$

4.2 Add Sets

Another stage where the choice of autonomy level influences the quality of the repair is when deciding what repairing solutions to retain for the final repair, i.e. A in Definition 4. The choices are summarized in Table 2 under ‘AS’ (Add Set). Choice AS_{ON} is the most general case and allows all kinds of axioms to be added to the network. This means that regardless which choice was used during the computation of the repairs, these repairs can be used without any adaptations. Choice AS_O only allows to add axioms within ontologies. This represents the case where ontology owners only focus on repairing their own ontologies. Similarly, choice AS_M only allows to add mappings and represents the case where alignment owners only focus on repairing their own mappings. In AS_O and AS_M , not all repairing suggestions may be retained for the final solution. Therefore, when ‘ON’ was used during the computation of repairs and ‘O’ or ‘M’ is used during this stage, this may lead to a lower level of completeness for the network than if ‘ON’ is used for this stage. The Hasse diagram for these choices is given in Fig. 4(b) and the proofs are given in the supplemental material.

4.3 Finalizing

In the case of ‘ON’ the repairing solutions may contain axioms in different ontologies as well as mappings. In this case the ontology and alignment owners may decide to materialize the knowledge derived from the network which regards their ontology or alignment. From the network point of view the network does not change logically as the same axioms can be derived. From the ontology or alignment point of view, the difference appears when they are disconnected from the network. The materialized versions are more (or equally) complete.

4.4 Discussion

When the network is owned by one entity (e.g., in the case of modular ontologies with mappings) or there is a tight cooperation between the owners of the

individual parts of the network (e.g., in a consortium such as EMMO (<https://github.com/emmo-repo/EMMO>) or OBO (<https://obofoundry.org/>)), and computation time and domain expert staff is not an issue, the recommended combination is to use $(KB_{ON}, AS_{ON}, \text{final materialization})$. This ensures the highest level of completeness for the network as well as for the ontologies and alignments when they are disconnected from the network.

The least complete networks come from the combinations (KB_O, AS_O) and (KB_M, AS_M) . Essentially, this means that mappings between the ontologies exist, but the network is not taken into account at all during repairing. The repairs are the same as when repairing is done without the network, and this seems to be a common case in practice.

When ontology or alignment owners require full control over the computation resources, while still taking into account the knowledge in the network, then the combinations with KB_{MO} and KB_{MM} may be a good choice. Maintaining full control about what is added leads to choices with AS_O and AS_M .

In the ontology alignment field there are alignment systems that also repair the alignment. The current systems require that the repair only contains mappings and thus use AS_M . Examples of such systems are ALCOMO [26], LogMap [13, 14], AgreementMakerLight [36]. Regarding the background knowledge, they all use the choice KB_{ON} . RaDON [12] is a system focusing on network repair, but makes the same choices as the alignment systems with repair functionality.

5 Experiments

As examples of the use of different choices for the combination operators, we performed experiments on 5 ontology networks. The ontologies (ekaw, sigkdd, iasted, cmt) used in these networks are from the conference track of the OAEI⁹. We have used the parts of these ontologies that are expressible in \mathcal{EL}_\perp in the sense that we removed the parts of axioms that used constructors not in \mathcal{EL}_\perp . We introduced wrong axioms in the ontologies and mappings between ontologies by replacing existing axioms with axioms where the left-hand or right-hand side concepts of the existing axioms were changed. Further, we also flagged some existing axioms as wrong in our full experiment set. All axioms were validated manually. The characteristics of the ontologies and the wrong axioms are shown in the supplemental material.

We repaired the networks using all choices regarding the use of background knowledge together with Algorithm 1. To repair ontology axioms we used KB_O , KB_{MO} and KB_{ON} . To repair mappings we used KB_M , KB_{MM} and KB_{ON} . We computed the sizes of the sets of sub-concepts and super-concepts used in weakening and completing. The sizes of these sets reflect the number of axioms that need to be validated by the oracle. We note, however, that using the visualization in our system (see below) these sets are shown together and thus the validation of many axioms can be done at the same time.

⁹ <http://oaei.ontologymatching.org/2023/conference/index.html>.

The full results for the experiments are shown in the supplemental material. In the experiments, the number of axioms to be validated for KB_{ON} is between 3 and 5 times higher than for KB_O/KB_M . The validation work for the materialized versions $\text{KB}_{MO}/\text{KB}_{MM}$ is between 6% and 40% higher than for KB_O/KB_M . For all networks there were axioms for which the repairing with an operator higher up in the Hasse diagram led to a strictly more complete ontology network.

6 Implemented System

We implemented a Java-based system which extends the \mathcal{EL} version of the RepOSE system ([22, 24, 43]) with full debugging, weakening and completing for \mathcal{EL}_\perp ontology network repairing. The system allows the user to choose different combinations, thereby giving a choice in the trade-off between validation work, incorrectness and completeness.

The system uses an interactive way to repair the ontology network. It takes as input the ontologies and alignments in the network as well as a set of wrong axioms. It is also possible to not give a set of wrong axioms but let the system deduce unsatisfiable concepts in the network. As basic algorithms, we have used the black-box algorithm in [15] for debugging, and the weakening and completing algorithms in [24]. With respect to the combination operators in the Hasse diagrams, the system supports (S-one, D-v-all) and (S-one, D-one-v) thereby providing the choice to validate all axioms in the justifications or to validate Hitting sets. Regarding weakening the combinations are (R-none, AB-none/W-all, U-end_all), (R-all, AB-none/W-all, U-end_all) and (R-one, AB-one/W-one, U-end_all). The first combination does not remove wrong axioms during the computation, weakens all axioms at once and updates at the end. The second removes all wrong axioms before the computation, weakens all axioms at once and updates at the end. The third removes wrong axioms one at a time during the computation and puts them back before dealing with the next wrong axiom. It weakens axioms one at a time and updates at the end. For completing, the choices are (C-all, U-end_all) and (C-one, U-end_one). The first combination completes all weakened axioms at once and updates at the end. The second completes the weakened axioms one at a time and updates the ontology after the weakened axiom set is handled for each wrong axiom. Regarding the choices for the combinations for the network, all choices for KB are implemented. The AS choices are supported using visual clues. The concepts in the axioms to be validated are labeled with the ontology source, such that the user can distinguish mappings from axioms within the ontologies. We also use different colors to represent the concepts which belong to different ontologies. At the appropriate times the system shows the different combinations that can be chosen and the user can select the desired choice.

For the basic operations the user interactions are adapted to the task at hand and different panels are used. For debugging the user requests the generation of the justifications. Then, the user can validate the axioms in the justifications or ask the system to compute Hitting sets and validate the axioms in those.

The axioms to be validated are shown in a list. During weakening, after choosing the preferred combination strategy, the user requests the system to generate the candidate weakened axioms for each axiom $\alpha \sqsubseteq \beta$ in the wrong asserted axioms set. The system computes the set of sub-concepts of α (sub) and the set of super-concepts of β (sup), thereby representing the possible choices for weaker axioms. These weaker axioms can be visualized in two ways: (i) as a list of axioms and (ii) by two panes representing the sub and sup sets with their subsumption relations (Fig. 5), respectively. In the latter case the user can choose weakened axioms by clicking on a concept in the sub set and a concept in the sup set and select the axiom as a weakened axiom. The advantage of this case is that these axioms are validated with the context in the domain of the ontology (showing the partial ontology through visualization). The set-up for completing is similar as for the weakening. For an axiom $\alpha \sqsubseteq \beta$ to be completed, the set of super-concepts of α and the set of sub-concepts of β are computed and visualized as lists or using two panes (Fig. 6). Validation is performed in a similar way as for weakening.

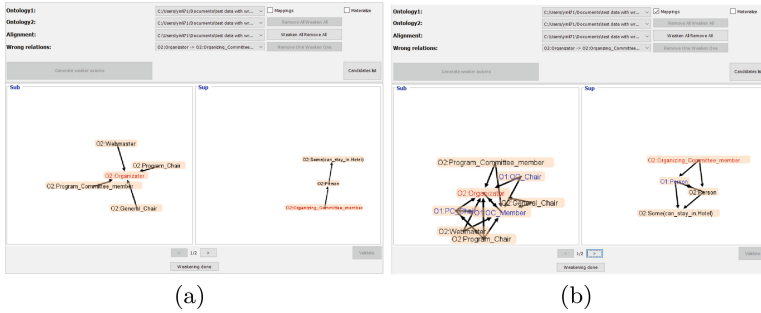


Fig. 5. Sub and sup when weakening wrong axiom $\text{Organizer} \sqsubseteq \text{Organizing_Committee_Member}$ in ontology network ekaw-sigkdd using (a) KB_O ; (b) KB_{ON} . Concepts in wrong axiom in red, concepts in ekaw in blue, concepts in sigkdd in black. (Color figure online)

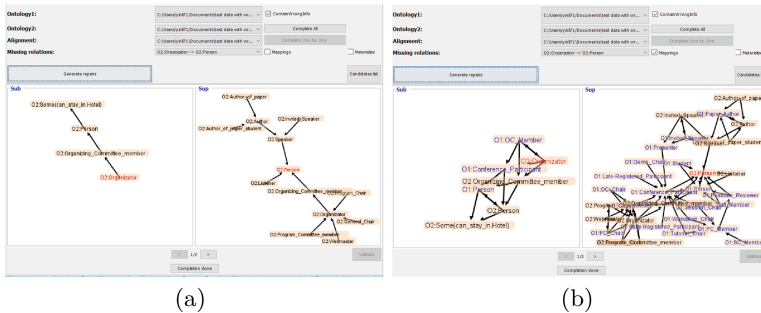


Fig. 6. Sub and sup when completing weakened axiom $\text{Organizer} \sqsubseteq \text{Person}$ in ontology network ekaw-sigkdd using (a) KB_O ; (b) KB_{ON} . Concepts in weakened axiom in red, concepts in ekaw in blue, concepts in sigkdd in black. (Color figure online)

7 Conclusion

In this paper we presented a framework for repairing ontology networks that alleviates the problem of removing correct knowledge when removing unwanted axioms from the network. It uses the basic operators of debugging, removing, weakening and completing for which there exist different approaches. We defined combination operators that represent choices to be made when combining the basic operations. We introduced different levels of autonomy for the ontologies and alignments in the networks and defined combination operators based on these levels and the use of background knowledge during the computation of repairs and the selection of the final solution. We have shown the influence of the combination operators on the quality of the repaired network. The framework gives flexibility to the ontology and alignment owners regarding their use and update policies. The framework also provides a blueprint to extend existing systems to more general systems for repairing ontologies and ontology networks.

Supplemental Material Statement: The supplemental material is available at <https://figshare.com/s/e652a8e100579316f876>. It includes (i) an extended version of this paper [25] with information about combinations of basic operations, derivation of the Hasse diagrams, extended information for the example in Sect. 4, and full experiment results, (ii) ontology versions used in the experiments and (iii) implemented system.

Acknowledgements. This work is financially supported by the Swedish e-Science Research Centre (SeRC) and the EU Horizon Europe project Onto-DESIDÉ under Grant Agreement 101058682.

References

1. Arif, M.F., Mencía, C., Ignatiev, A., Manthey, N., Peñaloza, R., Marques-Silva, J.: BEACON: an efficient SAT-based tool for debugging \mathcal{EL}^+ ontologies. In: Creignou, N., Le Berre, D. (eds.) SAT 2016. LNCS, vol. 9710, pp. 521–530. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40970-2_32
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
3. Baader, F., Kriegel, F., Nuradiansyah, A., Peñaloza, R.: Making repairs in description logics more gentle. In: KR, pp. 319–328 (2018)
4. Chen, J., et al.: Knowledge graphs for the life sciences: recent developments, challenges and opportunities. Trans. Graph Data Knowl. **1**(1), 5:1–33 (2023). <https://doi.org/10.4230/TGDK.1.1.5>
5. Confalonieri, R., Galliani, P., Kutz, O., Porello, D., Righetti, G., Troquard, N.: Towards even more irresistible axiom weakening. In: DL (2020)
6. Du, J., Wan, H., Ma, H.: Practical TBox abduction based on justification patterns. In: AAI, pp. 1100–1106 (2017)

7. Fleischhacker, D., Meilicke, C., Völker, J., Niepert, M.: Computing incoherence explanations for learned ontologies. In: Faber, W., Lembo, D. (eds.) RR 2013. LNCS, vol. 7994, pp. 80–94. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39666-3_7
8. Haifani, F., Koopmann, P., Tournet, S., Weidenbach, C.: Connection-minimal abduction in \mathcal{EL} via translation to fol. In: Blanchette, J., Kovács, L., Pattinson, D. (eds.) IJCAR 2022. LNCS, vol. 13385, pp. 188–207. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-10769-6_12
9. Hogan, A., et al.: Knowledge graphs. ACM Comput. Surv. **54**(4), 71:1–37 (2022). <https://doi.org/10.1145/3447772>
10. Ivanova, V., Laurila Bergman, J., Hammerling, U., Lambrix, P.: Debugging Taxonomies and their Alignments: the ToxOntology - MeSH Use Case. In: WoDOOM. LECP, vol. 79, pp. 25–36 (2012)
11. Ji, Q., Gao, Z., Huang, Z., Zhu, M.: An efficient approach to debugging ontologies based on patterns. In: Pan, J.Z., et al. (eds.) JIST 2011. LNCS, vol. 7185, pp. 425–433. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29923-0_33
12. Ji, Q., Haase, P., Qi, G., Hitzler, P., Stadtmüller, S.: RaDON — repair and diagnosis in ontology networks. In: Aroyo, L., et al. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 863–867. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02121-3_71
13. Jiménez-Ruiz, E., Cuenca Grau, B.: LogMap: logic-based and scalable ontology matching. In: Aroyo, L., et al. (eds.) ISWC 2011. LNCS, vol. 7031, pp. 273–288. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25073-6_18
14. Jiménez-Ruiz, E., Grau, B.C., Zhou, Y., Horrocks, I.: Large-scale interactive ontology matching: Algorithms and implementation. In: ECAI, pp. 444–449 (2012). <https://doi.org/10.3233/978-1-61499-098-7-444>
15. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 267–280. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_20
16. Kalyanpur, A., Parsia, B., Sirin, E., Cuenca-Grau, B.: Repairing unsatisfiable concepts in OWL ontologies. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 170–184. Springer, Heidelberg (2006). https://doi.org/10.1007/11762256_15
17. Kazakov, Y., Skočovský, P.: Enumerating justifications using resolution. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) IJCAR 2018. LNCS (LNAI), vol. 10900, pp. 609–626. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94205-6_40
18. Lambrix, P.: Completing and debugging ontologies: State of the art and challenges in repairing ontologies. ACM J. Data Inf. Qual. **15**(4), 41:1–38 (2023). <https://doi.org/10.1145/3597304>
19. Lambrix, P., Armiento, R., Li, H., Hartig, O., Abd Nikooie Pour, M., Li, Y.: The materials design ontology. Semantic Web **15**(2), 481–515 (2024). <https://doi.org/10.3233/SW-233340>
20. Lambrix, P., Dragisic, Z., Ivanova, V.: Get my pizza right: repairing missing is-a relations in \mathcal{ALC} ontologies. In: Takeda, H., Qu, Y., Mizoguchi, R., Kitamura, Y. (eds.) JIST 2012. LNCS, vol. 7774, pp. 17–32. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37996-3_2

21. Lambrix, P., Ivanova, V.: A unified approach for debugging is-a structure and mappings in networked taxonomies. *J. Biomed. Semant.* **4**, 10 (2013). <https://doi.org/10.1186/2041-1480-4-10>
22. Lambrix, P., Wei-Kleiner, F., Dragisic, Z.: Completing the is-a structure in light-weight ontologies. *J. Biomed. Semant.* **6**(12), 1–26 (2015). <https://doi.org/10.1186/s13326-015-0002-8>
23. Lehmann, J., Bühmann, L.: ORE - a tool for repairing and enriching knowledge bases. In: Patel-Schneider, P.F., et al. (eds.) *ISWC 2010. LNCS*, vol. 6497, pp. 177–193. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17749-1_12
24. Li, Y., Lambrix, P.: Repairing \mathcal{EL} ontologies using weakening and completing. In: Pesquita, C., et al. (eds.) *ESWC 2023. LNCS*, vol. 13870, pp. 298–315. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-33455-9_18
25. Li, Y., Lambrix, P.: Repairing networks of \mathcal{EL}_{\perp} ontologies using weakening and completing - extended version (2024). [arXiv:2407.18848](https://arxiv.org/abs/2407.18848)
26. Meilicke, C.: Alignment Incoherence in Ontology Matching. Ph.D. thesis, University of Mannheim (2011)
27. Meyer, T., Lee, K., Booth, R., Pan, J.: Finding maximally satisfiable terminologies for the description logic \mathcal{ALC} . In: *AAAI*, pp. 269–274 (2006)
28. Moodley, K., Meyer, T., Varzinczak, I.J.: Root justifications for ontology repair. In: Rudolph, S., Gutierrez, C. (eds.) *RR 2011. LNCS*, vol. 6902, pp. 275–280. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23580-1_24
29. Méndez, J., Alrabbaa, C., Koopmann, P., Langner, R., Baader, F., Dachsel, R.: Evonne: a visual tool for explaining reasoning with OWL ontologies and supporting interactive debugging. *Computer Graphics forum* **42**(6), e14730:1–15 (2023). <https://doi.org/10.1111/cgf.14730>
30. Nikitina, N., Rudolph, S., Glimm, B.: Interactive ontology revision. *J. Web Semant.* **12**, 118–130 (2012). <https://doi.org/10.1016/j.websem.2011.12.002>
31. Noy, N., Gao, Y., Jain, A., Narayanan, A., Patterson, A., Taylor, J.: Industry-scale knowledge graphs: lessons and challenges. *Commun. ACM* **62**(8), 36–43 (2019). <https://doi.org/10.1145/3331166>
32. Paulheim, H.: Knowledge graph refinement: a survey of approaches and evaluation methods. *Semantic Web J.* **8**(3), 489–508 (2017). <https://doi.org/10.3233/SW-160218>
33. Peñaloza, R.: Axiom pinpointing. In: Cota, G., Daquino, M., Pozzato, G.L. (eds.) *Applications and Practices in Ontology Design, Extraction, and Reasoning, Studies on the Semantic Web*, vol. 49, pp. 162–177. IOS Press (2020). <https://doi.org/10.3233/SSW200042>
34. Pesquita, C., Faria, D., Santos, E., Couto, F.M.: To repair or not to repair: reconciling correctness and coherence in ontology reference alignments. In: *OM*, pp. 13–24 (2013)
35. Rodler, P., Schmid, W.: On the impact and proper use of heuristics in test-driven ontology debugging. In: Benz Müller, C., Ricca, F., Parent, X., Roman, D. (eds.) *RuleML+RR 2018. LNCS*, vol. 11092, pp. 164–184. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99906-7_11
36. Santos, E., Faria, D., Pesquita, C., Couto, F.M.: Ontology alignment repair through modularization and confidence-based heuristics. *PLoS ONE* **10**(12), e0144807, 1–19 (2015). <https://doi.org/10.1371/journal.pone.0144807>

37. Schekotihin, K., Rodler, P., Schmid, W.: OntoDebug: interactive ontology debugging plug-in for Protégé. In: Ferrarotti, F., Woltran, S. (eds.) FoIKS 2018. LNCS, vol. 10833, pp. 340–359. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-90050-6_19
38. Schlobach, S.: Debugging and semantic clarification by pinpointing. In: Gómez-Pérez, A., Euzenat, J. (eds.) ESWC 2005. LNCS, vol. 3532, pp. 226–240. Springer, Heidelberg (2005). https://doi.org/10.1007/11431053_16
39. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: IJCAI, pp. 355–360 (2003)
40. Schlobach, S., Huang, Z., Cornet, R., van Harmelen, F.: Debugging Incoherent Terminologies. *J. Automated Reasoning* **39**(3), 317–349 (2007). <https://doi.org/10.1007/s10817-007-9076-z>
41. Shchekotykhin, K.M., Friedrich, G., Fleiss, P., Rodler, P.: Interactive ontology debugging: two query strategies for efficient fault localization. *J. Web Semantics* **12**, 88–103 (2012). <https://doi.org/10.1016/j.websem.2011.12.006>
42. Troquard, N., Confalonieri, R., Galliani, P., Peñaloza, R., Porello, D., Kutz, O.: Repairing ontologies via axiom weakening. In: AAI, pp. 1981–1988 (2018)
43. Wei-Kleiner, F., Dragisic, Z., Lambrix, P.: Abduction framework for repairing incomplete \mathcal{EL} ontologies: complexity results and algorithms. In: AAI, pp. 1120–1127 (2014)