Completing and Debugging Ontologies: State of the Art and Challenges in Repairing Ontologies

Patrick Lambrix
Department of Computer and Information Science, Linköping University, Sweden
E-mail: patrick.lambrix@liu.se

Abstract.
As semantically-enabled applications require high-quality ontologies, developing and maintaining ontologies that are as correct and complete as possible is an important although difficult task in ontology engineering. A key step is ontology debugging and completion. In general, there are two steps: detecting defects and repairing defects. In this paper we discuss the state of the art regarding the repairing step. We do this by formalizing the repairing step as an abduction problem and situating the state of the art with respect to this framework. We show that there are still many open research problems and show opportunities for further work and advancing the field.

Keywords: Ontology engineering, Ontology alignment, Ontology completion, Abductive reasoning

1. Introduction
Ontologies (e.g., [1]) aim to define the basic terms and relations of a domain of interest, as well as the rules for combining these terms and relations. They standardize terminology in a domain and are a basis for semantically enriching data, semantic search, integration of data from different data sources, and reasoning over the data. Using ontologies can alleviate the variety (data sources are heterogeneous regarding the type and nature of data they store), variability (data can be inconsistent) and veracity (not all data can be trusted) problems. Furthermore, they are proposed as an enabler making data FAIR, i.e., findable, accessible, interoperable, and reusable, with the purpose of enabling machines to automatically find and use the data, and individuals to easily reuse the data [2]. Ontologies are also a key technology for the semantic web.

In recent years many ontologies have been developed (see [3] for a survey on ontology libraries). Further, ontologies have been connected to each other into ontology networks and there are some portals that store these (e.g., BioPortal (http://bioportal.bioontology.org/), Unified Medical Language System (http://www.nlm.nih.gov/research/umls/about_umls.html)). However, developing ontologies and networks are not easy tasks and there may be issues related to the quality of the ontologies and networks [4]. Two such issues are incorrectness (does the ontology contain wrong information?) and completeness (is information lacking?). Ontologies containing wrong information or lacking information, although often useful, also lead to problems when used in semantically-enabled applications. Wrong conclusions may be derived or valid conclusions may be missed. As an example, in [5] it was shown that semantically-enabled querying of PubMed (http://www.ncbi.nlm.nih.gov/pubmed/) using MeSH (Medical Subject Headings, http://www.nlm.nih.gov/mesh/) with one piece of information missing (i.e., that scleritis is a scleral disease) would lead to missing 55% of the result that is obtained with this piece of information. Therefore, it is essential to complete and debug ontologies and their networks.

1570-0844/0-1900/$35.00 © 0 – IOS Press and the authors. All rights reserved
Defects in ontologies can take different forms (e.g., [6]). 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 and the semantic defects. Modeling defects relate to the domain that is being modeled and include such things as missing concepts and relations, or statements that are not correct according to the domain. For instance, in [7] it was shown that for the two ontologies used in the Anatomy track in the Ontology Alignment Evaluation Initiative (OAEI, yearly event for evaluation of ontology alignment systems), at least 121 and 83, respectively, isa relations (defined below) that are correct in the domain are missing in these ontologies. Further, BioPortal contains results for the OAEI Anatomy track with ca 20% incorrect statements and ca 20% of the connections between ontologies are missing. Semantic defects relate to logical defects such as defining concepts that are logically equivalent to the empty set (called unsatisfiable concepts, formally defined below) or ontologies that contain contradictions. As an example, in [6] it was shown that the TAMBIS ontology contained 144 unsatisfiable concepts.

In this paper we review approaches for raising the quality of ontologies by repairing them. In general, completing and debugging requires two steps. In the detection step possible defects are detected. In the repairing step the detected wrong information is removed and missing information is added. In this paper we focus on the repairing step which can be formalized as an abduction problem and for which there are still many open research problems.

The remainder of the paper is organized as follows. In Sect. 2 we discuss preliminaries introducing notions related to ontologies (Sect. 2.1), description logics which are used as a basis for the formalization of the repairing problem (Sect. 2.2), and a short review of methods for the detection step (which is not the focus of the paper) and its relation to the repairing step (Sect. 2.3). In Sect. 3 we relate this survey to earlier surveys. Then, in Sect. 4 we formalize ontology repair (completion and debugging) as an abductive reasoning problem and, as there may be different ways to repair ontologies, we introduce different preference relations between solutions that are relevant to this problem. In Sect. 5 and 6 we discuss the state of the art based on our formalization for ontologies and ontology networks, respectively regarding debugging, completion and the combination of these. Further, we give some open problems related to theory and algorithms as well as regarding user involvement in Sect. 7. The paper concludes in Sect. 8.

2. Preliminaries

2.1. Ontologies and ontology networks

From a knowledge representation point of view, ontologies may contain four components: (i) concepts that represent sets or classes of entities in a domain (e.g., fracture in the example in the appendix, representing all fractures), (ii) instances that represent the actual entities (e.g., an actual fracture), (iii) relations (e.g., hasAssociatedProcess in the example in the appendix), and (iv) axioms that represent facts that are always true in the topic area of the ontology (e.g., a fracture is a pathological phenomenon). Concepts and relations are often organized in hierarchies using the is-a (or subsumption) relation, denoted by ⊑. When \( P \subseteq Q \), then \( P \) is a sub-concept of \( Q \) and all entities belonging to \( P \) also belong to \( Q \). Axioms can represent such things as domain restrictions, cardinality restrictions, or disjointness restrictions. Many ontologies do not contain instances and represent knowledge on the concept (or ‘schema’) level. In this paper we do not deal with instances.

Ontologies can be represented in different ways, but one of the more popular ways nowadays is to use (variants of) the OWL language (e.g., https://www.w3.org/TR/2012/REC-owl2-primer-20121211/). OWL is based on description logics, which are presented in Sect. 2.2. In description logics concepts, roles, individuals and axioms are used, which relate to concepts, binary relations, instances and axioms in ontology terminology, respectively.

An ontology network is a collection of ontologies and pairwise alignments between these ontologies. An alignment is a set of mappings (also called correspondences) between entities from the different ontologies. The most common kinds of mappings are equivalence mappings as well as mappings using is-a and its inverse.

2.2. Description logics

In this paper we assume that ontologies are represented using a description logic TBox. We introduce notions in the field of description logics that are relevant for this paper.
Description logics [8] are knowledge representation languages. In description logics 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 description logics allow for different constructors for defining complex concepts and roles. In current work on completing and debugging different logics are used such as \( E\mathcal{L} \) (which uses the top concept \( \top \), and the concept constructors conjunction and existential restriction) and \( \mathcal{ALC} \) (top concept \( \top \), bottom concept \( \bot \) and concept constructors conjunction, disjunction, negation, existential and universal restrictions). We mention later also \( \mathcal{SHOIN} \) which in addition to the \( \mathcal{ALC} \) constructors allows constructors for transitive roles, role hierarchies, nominals, inverse roles and number restrictions.

In Table 1 we show the syntax of the \( \mathcal{ALC} \) constructors and refer to, e.g., [9] for information on other description logics.

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>top</td>
<td>( \top )</td>
<td>( \Delta^2 )</td>
</tr>
<tr>
<td>bottom</td>
<td>( \bot )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>negation</td>
<td>( \neg P )</td>
<td>( \Delta^2 \setminus P^2 )</td>
</tr>
<tr>
<td>conjunction</td>
<td>( P \land Q )</td>
<td>( P^2 \cap Q^2 )</td>
</tr>
<tr>
<td>disjunction</td>
<td>( P \lor Q )</td>
<td>( P^2 \cup Q^2 )</td>
</tr>
<tr>
<td>existential restriction</td>
<td>( \exists r.P )</td>
<td>( \exists y : (x,y) \in r^2 \land y \in P^2 )</td>
</tr>
<tr>
<td>universal restriction</td>
<td>( \forall r.P )</td>
<td>( \forall y : (x,y) \in r^2 \rightarrow y \in P^2 )</td>
</tr>
<tr>
<td>GCI</td>
<td>( P \subseteq Q )</td>
<td>( P^2 \subseteq Q^2 )</td>
</tr>
</tbody>
</table>

Table 1: \( \mathcal{ALC} \) syntax and semantics.

that the detection step is done, we briefly discuss detection and the connection to repairing.

2.3.1. Detection

There are many kinds of approaches to detect defects in ontologies and many of these are complementary.

A detection method that works for all kinds of defects is inspection of the ontologies. This requires ontology development environments that provide search, reasoning and explanation facilities to aid the domain expert in the inspection.

Most detection methods for semantic defects are logic-based and focus on wrong information in the ontologies. A common strategy is to detect unsatisfiable concepts or inconsistencies in ontologies or ontology networks using standard reasoning techniques. One problem that is reported in [10] is that, as an unsatisfiable concept that is used in the definition of other concepts may make these unsatisfiable as well, description logic reasoners could give large lists of unsatisfiable concepts. One way to alleviate this problem is as in [6] where these 'root' concepts are identified (although in [6] they did this during the repairing step).

There are many approaches to find missing information. There is much work on finding candidate relationships between terms in the ontology learning area [11]. In this setting, new ontology elements are derived from text using knowledge acquisition techniques. Another paradigm is based on machine learning and statistical methods, such as k-nearest neighbors approach [12], association rules [13], bottom-up hierarchical clustering techniques [14], supervised classification [15] and formal concept analysis [16].

A much used approach is to use patterns. The pioneering research conducted in this line is in [17], where...
the focus was on finding missing is-a relations. The work defines a set of lexico-syntactical patterns indicating is-a relationships between words in the text. However, depending on the chosen corpora, these patterns may occur rarely. Thus, although the approach has a reasonable precision, its recall is very low. Many variants have been proposed. Lexico-syntactic patterns as well as logic patterns have been used to find wrong as well as missing information in ontologies (e.g., [18–21]) and ontology networks (e.g., [22, 23]). The OOPS! system implements a variety of these [21].

There are also approaches that use knowledge that is intrinsic in an ontology network to detect defects. For instance, in [24, 25] a partial alignment between ontologies is used to detect missing is-a relations. These are found by looking at pairs of equivalence mappings. If there is an is-a relation between the terms in the mappings belonging to one ontology, but there is no is-a relation between the corresponding terms in the other ontology, then it is concluded that there is a candidate missing is-a relation in the second ontology. A similar approach is used in [26].

The detection of missing mappings is a research area on its own, i.e. ontology alignment [27], and we discuss this further in Sect. 6.

We note that for missing information these detection approaches usually do not detect all defects and they do not guarantee that the found defects are really missing. The found defects are really candidate defects which need to be validated by a domain expert.

2.3.2. Workflow

In much of the current work we find systems or methods that detect defects or repair defects, but usually not both.

A workflow for a system for completing and debugging ontologies and ontology networks contains two main steps: detection and repair. It is well-known that for high-quality results a domain expert needs to be involved in both steps. As the defects found by detection systems usually are candidate defects, a domain expert needs to validate the candidate defects as wrong input to the repairing step would lead to wrong repairs of the ontologies. Furthermore, in the repairing step domain experts are needed to validate repairs for modeling problems. Also for the semantic defects a domain expert is needed, as systems that are purely logic based may prefer logically correct solutions that are not correct according to the domain over solutions that are (e.g., [28]). The two steps do not need to be completely separated. For instance, when repairing the ontology new information is added or wrong information is deleted and this may be used to detect further defects.

As an example, the RepOSE system [24, 25] is a system for debugging and completing is-a structure and mappings in ontology networks, where only the named concepts and the is-a relations in the ontologies are considered. RepOSE has a detection step that uses knowledge intrinsic in the network to detect candidate missing is-a relations and mappings. These candidate defects are then validated by a domain expert. In the case a candidate missing is-a relation or mapping really is missing, then we need to complete, otherwise a wrong is-a relation or mapping is derivable from the network and thus we need to debug. The validated and classified defects (missing or wrong) are then used as input for the repairing step. Different algorithms are used for repairing different kinds of defects, but the sub-steps for each kind are generation of repairing actions (what to add or delete), the ranking of repairs (a proposed order in which to deal with the defects), the recommendation of repairing actions (using external knowledge) and finally, the execution of the repairs (using external knowledge) and finally, the execution of the repairing actions chosen by the domain expert (with computation of the consequences of the action). The consequences can include such things as other defects are also repaired, possible repairing actions for other defects change or new candidate defects are found. Furthermore, at any time during the process, the user can switch between different ontologies, start earlier phases, or switch between the repairing of different kinds of defects. The process ends when there are no more defects to deal with.

3. Related work

There are early surveys from 2007 [29, 30] where debugging approaches are reviewed. In this paper we introduce a framework with preference relations that in addition to debugging also includes completion, and that allows us to compare the different approaches in a uniform way. The early surveys discuss approaches where the ontologies can include instances, which we do not. Further, [29] introduces some criteria for debugging approaches. Regarding the criterion application the authors distinguish between different tasks. In this paper we focus on the repair task. The granularity of the repairs is on the axiom level in this paper. Regarding Tbox and Abox support, we focus on TBox support. Some algorithms focus on consistency
and some on incoherence. For some of the methods, an implementation is available. Regarding support of ontology networks, we have a dedicated section to this topic (Sect. 6). User involvement is discussed in Sect. 4.1.2 and Sect. 7.2. We mention the criteria preservation of structure, complexity and exploitation of context in relevant places in this paper.

4. Ontology Repair

In this section we focus on repairing ontologies represented in description logics, where we have already detected wrong and missing information. We only discuss ontologies at concept level, and thus do not deal with instances (or individuals in description logic terminology). We define this problem as an abductive reasoning problem, and as a repairing problem can have many solutions, we discuss preference relations between these. Further, we use an abstract example to exemplify the notions. However, in the appendix we give an example inspired by the Galen ontology.

4.1. Formalization

4.1.1. Repair

Definition 1. (Repair) Let T be a TBox and C be the set of all atomic concepts in T. Let M and W be finite sets of TBox axioms. Let Or be an oracle that given a TBox axiom returns true or false. A repair for Complete-Debug-Problem CDP(T, C, Or, M, W) is any pair of finite sets of TBox axioms (A, D) such that

(i) \( \forall \psi_d \in A: Or(\psi_d) = true \);
(ii) \( \forall \psi_d \in D: Or(\psi_d) = false \);
(iii) \( (T \cup A) \setminus D \) is consistent;
(iv) \( \forall \psi_m \in M: (T \cup A) \setminus D \models \psi_m \);
(v) \( \forall \psi_w \in W: (T \cup A) \setminus D \not\models \psi_w \).

Def. 1 formalizes the repair of an ontology for which missing and wrong information is given. An ontology is represented by a TBox T with its set of atomic concepts C. The identified missing and wrong information is represented by a set M of missing axioms, and a set W of wrong axioms. To repair the TBox, a set A of axioms that are correct according to the oracle should be added to the TBox and a set D of axioms that are not correct according to the oracle should be removed from the TBox such that the new TBox is consistent, the missing axioms are derivable from the new TBox and the wrong axioms are not derivable from the new TBox. As an example, consider the CDP in Fig. 1 and visualized in Fig. 2. Then \( R_1, R_2, R_3, R_4 \) and \( R_5 \) (visualized in Fig. 3) are all repairs of the CDP.

In general, the set of all axioms that are correct according to the domain and the set of all axioms that are not correct according to the domain are not known beforehand. Indeed, if these sets were given then we would only have to add the axioms of the first set to the TBox and remove the axioms in the second set from the TBox. The common case, however, is that we do not have these sets, but instead, we can rely on a domain expert who can decide whether an axiom is correct or not according to the domain. Therefore, in the formalization we introduce an oracle Or that represents the domain expert and that when given an axiom, returns true or false.

4.1.2. Influence of the quality of the oracle.

For Or we identified the following interesting cases.

The first case is the all-knowing oracle. In this case the oracle’s answer is always correct. This is the ideal case, but may not always be achievable. Most current work considers this kind of oracle. In the second case, the limited all-knowing oracle, if Or answers, then the answer is correct, but it may not know the answer to all questions. This case represents a domain expert who knows a part of the domain well. An approximation of this case is when there are several domain experts who may have different opinions and we use a skeptical approach. Only if all domain experts give the same answer regarding the correctness of an axiom, do we consider the answer. In the third case Or can make mistakes regarding the validation of axioms. Axioms that are not correct according to the domain may be validated as correct and vice versa. This is the most common case. Although most current work assumes an all-knowing oracle, recent work used, in addition to an all-knowing oracle, also oracles with specific error rates in the evaluation of ontology alignment systems [31]. A lesson learned was that oracles with error rates up to 30% were still beneficial for the systems. The fourth case represents situations where no domain expert is available and there is no validation of axioms, such as in fully automated systems.

As noted, most current work considers an all-knowing oracle. With an all-knowing oracle we can check that \( \forall \psi_m \in M: Or(\psi_m) = true \), and \( \forall \psi_w \in W: Or(\psi_w) = false \) and if this is not the case, we can remove the falsely identified defects. Therefore, we can, without loss of generality, assume that the axioms in M really are missing, and the axioms in W really are false. Furthermore, regarding repairs, when using an
given missing and wrong axioms really are missing in all-knowing oracle, we know that all added axioms in A are correct according to the domain and all removed axioms in D are false according to the domain. Furthermore, for an all-knowing oracle, we know that \( A \cap D = \emptyset \) and then \( (T \cup A) \setminus D = (T \setminus D) \cup A \). When using other oracles, we cannot be sure that the given missing and wrong axioms really are missing and wrong, respectively. Therefore, oracles that make mistakes or do not know the correctness of all axioms may start with wrong input. Also, wrong axioms may be added and correct axioms may be removed during the repairing. These issues may have a negative effect on the quality of the repaired ontology.

In practice, when using domain experts, it is not possible to know which kind of domain expert is used. When only one domain expert is available it is reasonable for the systems to assume that an all-knowing expert is used, although we should be aware that mistakes can occur. When more domain experts are available, a skeptical approach or a voting approach may be used for raising the quality of the ontology.

4.2. Preference relations

As there may exist many possible repairs for a given CDP, and not all are equally interesting, it is necessary to define preference relations between repairs.

4.2.1. Basic preferences

From the completion perspective of a complete-debug problem it is important to find repairs that add as much correct information as possible to the ontology, while from the correctness perspective wrong information should be removed as much as possible. Def. 2 and 3 formalize these intuitions respectively.

Def. 2 states that a repair \( R \) is more complete than another repair \( R' \) if all correct statements that can be
derived from the ontology repaired by $R'$ also can be derived from the ontology repaired by $R$ and there is a correct statement that can be derived from the ontology repaired by $R$, but not from the ontology repaired by $R'$. Therefore, if $S$ is more complete than another repair $R'$ then the ontology repaired by $R$ contains more correct statements than the ontology repaired by $R'$.

Further, when the same correct statements can be derived from the ontology repaired by $R$ and the ontology repaired by $R'$, then $R$ and $R'$ are equally complete.

**Definition 2.** (more complete) Let $R = (A, D)$ and $R' = (A', D')$ be two repairs for CDP($T, C, Or, M, W$). $R$ is more complete than $R'$ (or $R$ is preferred to $R'$ with respect to 'more complete') iff

\[
(\forall \psi : ((T \cup A') \setminus D' \models \psi \land Or(\psi) = true) \rightarrow (T \cup A) \setminus D \models \psi)
\]

\[
\land (\exists \psi : Or(\psi) = true \land (T \cup A) \setminus D \models \psi \land (T \cup A') \setminus D' \not\models \psi).
\]

$R$ and $R'$ are equally complete iff

\[
\forall \psi : Or(\psi) = true \rightarrow ((T \cup A') \setminus D' \models \psi \leftrightarrow (T \cup A) \setminus D \models \psi).
\]

Def. 3 states that a repair $R$ is less incorrect than another repair $R'$ if all wrong statements that can be derived from the ontology repaired by $R$ also can be derived from the ontology repaired by $R'$ and there is a wrong statement that can be derived from the ontology repaired by $R'$, but not from the ontology repaired by $R$. Therefore, if $R$ is less incorrect than another repair $R'$ then the ontology repaired by $R$ contains less wrong statements than the ontology repaired by $R'$.
Further, when the same wrong statements can be derived, from the ontology repaired by \( R \) and the ontology repaired by \( R' \), then \( R \) and \( R' \) are equally incorrect.

**Definition 3.** (less incorrect) Let \( R = (A, D) \) and \( R' = (A', D') \) be two repairs for CDP(\( T, C, Or, M, W \)). \( R \) is less incorrect than \( R' \) (or \( R \) is preferred to \( R' \) with respect to \( \text{less incorrect} \)) iff

\[
(\forall \psi : ((T \cup A) \setminus D \models \psi \land Or(\psi) = \text{false}) \rightarrow (T \cup A') \setminus D' \models \psi)) \\
\land (\exists \psi : Or(\psi) = \text{false} \land (T \cup A) \setminus D \not\models \psi) \\
\land (T \cup A') \setminus D' \models \psi).
\]

\( R \) and \( R' \) are equally incorrect iff

\[
\forall \psi : Or(\psi) = \text{false} \rightarrow ((T \cup A') \setminus D' \models \psi \leftrightarrow (T \cup A) \setminus D \models \psi))
\]

Def. 4 defines a classical preference relation for abduction problems related to removing redundancy using the subset relation. It compares the add and delete sets of two repairs.

**Definition 4.** (subset) Let \( R = (A, D) \) and \( R' = (A', D') \) be two repairs for CDP(\( T, C, Or, M, W \)). \( R \subseteq A \) \( R' \) iff \( A \subseteq A' \).

\( R \subseteq D \) \( R' \) iff \( D \subseteq D' \).

\( R \subseteq R' \) iff \( R \subseteq A \) \( D \subseteq D' \).

\( (R \subseteq R') \) we also say that \( R \) is preferred to \( R' \) with respect to \( \subseteq \).

As examples, for the CDP in Fig. 1 we have that \( R_3 \subseteq R_2 \subseteq R_1 \) and \( R_4 \subseteq R_5 \subseteq R_1 \). Further, \( R_1 \) is less incorrect than \( R_2, R_3, R_4 \) and \( R_5, R_2 \) and \( R_3 \) are equally incorrect, and \( R_1 \) and \( R_5 \) are equally incorrect. We also have that \( R_1, R_2, R_3 \) and \( R_4 \) are equally complete and they are more complete than \( R_5 \).

**4.2.2. Preferred repairs with respect to a basic preference**

Based on these preference relations we can define repairs that are preferred with respect to one particular preference relation (Def. 5 - 7).

**Definition 5.** (maximally complete) A repair \( R = (A, D) \) for CDP(\( T, C, Or, M, W \)) is said to be maximally complete (or preferred with respect to \( \text{more complete} \)) iff there is no repair \( R' \) which is more complete than \( R \).

**Definition 6.** (minimally incorrect) A repair \( R = (A, D) \) for CDP(\( T, C, Or, M, W \)) is said to be minimally incorrect (or preferred with respect to \( \text{less incorrect} \)) iff there is no repair \( R' \) which is less incorrect than \( R \).

**Definition 7.** (subset minimal) A repair \( R = (A, D) \) for CDP(\( T, C, Or, M, W \)) is said to be subset minimal (or preferred with respect to \( \subseteq \)) iff there is no repair \( R' \) such that \( R' \subseteq A \) \( R \) and \( R' \subseteq D \) \( R \).

As examples, for the CDP in Fig. 1 we have that \( R_3 \) and \( R_4 \) are subset minimal, \( R_1 \) is minimally incorrect, and \( R_1, R_2, R_3 \) and \( R_5 \) are maximally complete.

**4.2.3. Combining preferences**

The criteria regarding completeness and correctness are desirable as completeness leads to more correct information and correctness leads to less incorrect information. In most cases also the reduction of redundancy is desirable (but see below for cases where this is not the case). Therefore, we define different ways to combine these criteria. First, we need to define when a repair dominates another repair with respect to preference relations (Def. 8). A repair \( R \) dominates another repair \( R' \) if \( R \) is at least equally preferred to \( R' \) for each of a selected set of preference criteria and more preferred for at least one of those.

**Definition 8.** (dominate) Let \( R = (A, D) \) and \( R' = (A', D') \) be two repairs for CDP(\( T, C, Or, M, W \)). \( R \) dominates \( R' \) with respect to a set of preference relations \( P \subseteq \{ \text{more complete}, \text{less incorrect}, \subseteq \} \) if \( R \) is more or equally preferred to \( R' \) for all preference relations in \( P \). \( R \) is said to be \( X \)-optimal with respect to \( P \) if \( R \) is preferred with respect to \( X \) and there is no other repair that is preferred with respect to \( X \) and dominates \( R \) with respect to \( P \).

Using the definition of dominate, we can now define a preference relation that combines the basic preference relations, but which prioritizes one of those. We prefer repairs that are preferred with respect to a prioritized basic preference relation, and that are not dominated by other such repairs. Def. 9 formalizes this.

**Definition 9.** (combining with priority to one of the preference relations) Let \( X \subseteq \{ \text{more complete}, \text{less incorrect}, \subseteq \} \). Let \( P \subseteq \{ \text{more complete}, \text{less incorrect}, \subseteq \} \setminus \{ X \} \). A repair \( R \) for CDP(\( T, C, Or, M, W \)) is said to be \( X \)-optimal with respect to \( P \) if \( R \) is preferred with respect to \( X \) and there is no other repair that is preferred with respect to \( X \) and dominates \( R \) with respect to \( P \).

We can also define a preference relation that combines basic preference relations, but where the basic preference relations have equal priority. In this case we prefer repairs that are not dominated by other repairs according to the selected basic preferences. Def. 10 formalizes this.
Definition 10. (combining with equal priority) A repair \( R \) for CDP\((T, C, \text{Or}, M, W)\) is said to be skyline-optimal with respect to \( \mathcal{P} \) if there is no other repair that dominates \( R \) with respect to \( \mathcal{P} \).

We note that if a repair is X-optimal with respect to \( \mathcal{P} \), then it is skyline-optimal with respect to \( \mathcal{P} \cup \{X\} \).

As examples, for the CDP in Fig. 1 we have that \( R_4 \) is \( \subseteq \)-optimal with respect to \{less incorrect\} and \( R_3 \) is \( \subseteq \)-optimal with respect to \{more complete\}. Further, \( R_1 \) is less-incorrect-optimal with respect to \{more complete\} and more-complete-optimal with respect to \{less incorrect\}.

The advantage of maximally complete and more-complete-optimal repairs is that a maximal body of correct information is added to the ontology and for the latter without redundancy and/or with removing as much wrong information as possible. The advantage of minimally incorrect and less-incorrect-optimal repairs is that a maximal body of wrong information is removed from the ontology and for the latter without redundancy and/or with adding as much correct information as possible. Although these are the most attractive repairs, in practice it is not clear how to generate such repairs, apart from a usually infeasible brute-force procedure that checks the correctness of all axioms with the oracle. (Although a strategy can be devised to check all without asking the oracle for each axiom, the number of requests will still be large.) Repairs prioritizing subset minimality ensure that there is no redundancy. The advantage of removing redundant axioms is the reduction of computation time as well as the reduction of unnecessary user interaction. However, in some cases redundancy may be interesting. For instance, developers may want to have explicitly stated axioms in the ontologies even though they are redundant. This can happen, for instance, for efficiency reasons in applications or as domain experts have validated asserted axioms, these may be considered more trusted than derived axioms. Furthermore, focusing on redundancy may lead to less complete or more incorrect repairs. Skyline-optimal is a relaxed criterion. When, for instance, \( \mathcal{P} = \{\text{more complete, less incorrect}\} \), then a skyline-optimal repair with respect to \( \mathcal{P} \) is a preferred repair with respect to correctness for a certain level of completeness, or a preferred repair with respect to completeness for a certain level of correctness. In practice, as it is not clear how to generate more-complete-optimal and less-incorrect-optimal repairs, a skyline-optimal repair may be the next best thing, and in some cases (e.g., Sect. 5.2) it is easy to generate a skyline-optimal repair. However, in general, the difficulty lies in reaching as high levels of completeness and as low levels of incorrectness as possible.

5. State of the art - ontologies

Most of the current work has focused on the correctness or the completeness of ontologies, but very little work has dealt with both. However, a naive combination of a completion step and a debugging step does not necessarily lead to repairs for the combined problem. In this section we discuss current work.

5.1. Correctness

When only dealing with repairing the inconsistency or incoherence of Tboxes (semantic defects), only wrong information is dealt with. Therefore, in Def. 1, \( M = \emptyset \) and \( A = \emptyset \). In most current approaches the domain expert is not included. This means that choices are made solely based on the logic and that correct axioms may be removed from the ontologies. Therefore, not all solutions may actually be repairs as defined in Def. 1 as requirement (ii) may not be satisfied.

There is much work on repairing semantic defects. Most approaches are based on finding explanations or justifications for the defects using a glass-box or black-box approach [6]. A glass-box approach is based on the internals of the reasoning algorithm of a description logic reasoner. A black-box approach uses a description logic reasoner as an oracle to determine answers to standard description logic reasoning tasks such as checking concept satisfiability or subsumption with respect to an ontology.

A general approach for repairing incoherent ontologies is the following (adapted from [32]). (For inconsistent ontologies we can use a similar approach.) For a given set of unsatisfiable concepts for an ontology, compute the minimal explanations for the defects, i.e., the minimal reasons for the unsatisfiability of concepts. These minimal reasons for the unsatisfiability of a concept are sets of axioms and are called minimal unsatisfiability-preserving sub-TBoxes (MUPS) or justifications for the unsatisfiability. We need to compute these MUPS or justifications for all unsatisfiable concepts. From these we can compute the minimal incoherence-preserving sub-TBoxes (MIPS) which are the smallest sets of axioms in the original TBox that cause that TBox to be incoherent. To repair...
the incoherent TBox, we need to remove at least one axiom from each MIPS. We now define the notions in this general repairing approach formally.

The definition of MUPS is given in Def. 11. A MUPS in a consistent TBox can be seen as a justification (Def. 12) for an unsatisfiable concept. Indeed, if we instantiate $\psi$ in Def. 12 with $P \subseteq \bot$ we obtain the MUPS for $P$. The definition of MIPS is given in Def. 13.

**Definition 11.** (MUPS) [32] Let $T$ be a TBox and $P$ be an unsatisfiable concept in $T$. A set of axioms $T' \subseteq T$ is a minimal unsatisfiability-preserving sub-TBox (MUPS) if $P$ is unsatisfiable in $T'$ and $P$ is satisfiable in every sub-TBox $T'' \subseteq T'$.

**Definition 12.** (Justification) (similar to [33]) Let $T$ be a consistent TBox and $T \models \psi$. A set of axioms $T'$ \subseteq $T$ is a justification for $\psi$ in $T$ if $T' \models \psi$ and $\forall T'' \subseteq T' : T'' \not\models \psi$.

**Definition 13.** (MIPS) [32] Let $T$ be an incoherent TBox. A TBox $T' \subseteq T$ is a minimal incoherence-preserving sub-TBox (MIPS) if $T'$ is incoherent and every sub-TBox $T'' \subseteq T'$ is coherent.

As mentioned, to repair the incoherent TBox, we need to remove at least one axiom from each MIPS. Essentially, this means we should compute a hitting set (Def. 14) of the set of MIPS and remove the hitting set from the TBox. In [32] these hitting sets are called pinpoints. Complexity results regarding this problem are given in [34–36].

**Definition 14.** (hitting set) ([37]) Let $T$ be a collection of sets. A hitting set for $T$ is a set $H \subseteq \bigcup S \in T$ such that $\forall S \in T : H \cap S \neq \emptyset$.

As an example, consider the TBox in Fig. 1. This TBox is incoherent with unsatisfiable concepts $P_1$ and $P_3$. The set of MUPSs for $P_1$ is $\{(P_1 \nsubseteq P_2 (ax1), P_1 \nsubseteq \neg P_4 (ax3), P_2 \nsubseteq P_4 (ax4)), (P_1 \nsubseteq P_3 (ax2), P_3 \nsubseteq \neg P_5 (ax6), P_3 \nsubseteq P_6 (ax7), P_3 \nsubseteq \forall S.P_8 (ax9), P_3 \nsubseteq \exists S.\neg P_8 (ax10))\}$ while the set of MIPSs for $P_3$ is $\{(P_3 \nsubseteq P_5 (ax6), P_3 \nsubseteq P_6 (ax7), P_5 \nsubseteq \forall S.P_8 (ax9), P_6 \nsubseteq \exists S.\neg P_8 (ax10))\}$. The set of MIPSs is $\{(P_1 \nsubseteq P_2 (ax1), P_1 \nsubseteq \neg P_4 (ax3), P_2 \nsubseteq P_4 (ax4), P_3 \nsubseteq P_5 (ax6), P_3 \nsubseteq P_6 (ax7), P_3 \nsubseteq \forall S.P_8 (ax9), P_6 \nsubseteq \exists S.\neg P_8 (ax10))\}$. A possible hitting set is $\{P_1 \nsubseteq P_2 (ax1), P_3 \nsubseteq P_5 (ax6)\}$.

In general, there may be several hitting sets for the set of MIPS. Different approaches use different heuristics for ranking the possible repairs.
As there may be different ways to repair the ontologies and as computing justifications can be expensive, different heuristics and optimization approaches have been proposed (e.g., [43]). In [32] a heuristic is used stating that axioms appearing in more MUPSs are likely to be more erroneous. Therefore, axioms appearing in the most MUPSs are removed first (or, in other words, are first added to the hitting set). In [44] an arity-based heuristic is used which is similar to the heuristic in [32]. Further, [44] introduces heuristics based on the impact on the ontology when an axiom is removed and based on test cases applied by a user, e.g., by specifying desired and undesired entailments, which may be seen as an oracle that has validated certain entailments a priori. They also propose to use provenance information about the axioms as well as information about how often the elements in the axioms are used in the other axioms in the ontology to rank the axioms. Reiter’s hitting set algorithm is modified to take into account the axiom rankings. The approach is implemented in a prototype for a plug-in to SWOOP. In [6] root concepts are repaired first. A root concept is an unsatisfiable concept for which a contradiction in its definition does not depend on the unsatisfiability of another concept. The other unsatisfiable concepts are then derived concepts. Repairing root concepts may automatically repair derived concepts. The idea of roots is used in [45] where the debugging is not restricted to unsatisfiable concepts, but to axioms that are unwanted. A variant of justification, called root justification, for a set of axioms $U$ is defined as a set of axioms $RJ$ that is a justification of at least one of the axioms in $U$ and there is no justification of an axiom in $U$ that is a proper subset of $RJ$. The authors show via experiments that the number of root justifications is usually lower than the number of justifications. The idea of roots is also used in the ORE system [46] that implements a sound (all found dependencies are correct), but incomplete (not all dependencies are found) algorithm. In [47] a notion of relevance between axioms is defined and used to guide the computation of justifications. Patterns explaining unsatisfiability are used in [48] to optimize finding MUPS.

One of the few interactive approaches in debugging of ontologies is test-driven ontology debugging. In this approach queries are generated that are classified by an oracle as true (positive) or false (negative). Repairs that do not conform to the answers are discarded. Essentially, this is a strategy to guide a domain expert through the space of possible repairs to choose the repair that eventually is executed. An important issue in this approach is how to generate the queries to the oracle (e.g., [49]). As shown in [50], there are many strategies, but none performs best for all cases. The right choice of strategy, however, is important as in some experiments the overhead for the oracle effort for the worst strategy with respect to the best strategy was over 250%. A system that implements test-driven ontology debugging is OntoDebug [51] which implements a number of earlier described methods for computing repairs, and guides the user using queries to find a final repair.

An approach based on the idea of truth maintenance systems is proposed in [52] where a set of rules is defined that are used to compute consequences from the axioms in the ontology and explanations for unsatisfiable concepts and properties. The relatively lightweight description logic that is used should be sufficient for the representation of many learned (in contrast to manually developed) ontologies.

An approach that has not been proposed earlier, but that follows naturally from the definitions and preferences of Sect. 4, is to use the oracle for the axioms in the MUPSs. For every MUPS, remove the axioms $\phi$ such that $Or(\phi) = true$. If at least one of the MUPS becomes the empty set, then there is no repair unless we are willing to remove correct information. Assuming we have non-empty MUPSs after the removal of correct axioms, a hitting set would result in a repair. When redundancy is removed, we obtain a subset minimal repair. Another possibility, as we have checked the correctness using the oracle, is to use all remaining axioms in all MUPSs (as for these axioms $\phi$ we have that $Or(\phi) = false$). This repair is less incorrect than the repairs obtained using hitting sets.

There are also approaches that map the debugging problem into a revision problem (e.g., [53–55]). A revision state [54] is a tuple of ontologies $(O, O^w, O^u)$
where \( O^= \subseteq O \), \( O^\neq \subseteq O \), and \( O^= \cap O^\neq = \emptyset \). \( O^= \) represents the wanted consequences of \( O \), while \( O^\neq \) represents the unwanted consequences. In a complete revision state we also have that \( O = O^= \cup O^\neq \). For a CDP, \( O^\neq \) could be initialized with \( W \) (and when dealing with completion, \( O^= \) could be initialized with \( M \)).

The approach in [54] is an interactive method where questions are asked to an oracle to decide whether an axiom is correct or not, and then consequences are computed and revision states are updated iteratively. The decision on which questions to ask is based on the computation of an axiom impact measure. In [55] a MIPS approach is used in the definition of the revision operator. In [53] the authors show how ontology debugging relates to theoretical aspects in revision and show, for instance, that axiom pinpointing is related to the problem of finding kernels in revision.

5.2. Completeness

Most of the work on completing ontologies has dealt with completing the is-a structure of ontologies. An all-knowing oracle is often assumed. Therefore, in Def. 1, \( \forall \psi_m \in M : Or(\psi_m) = true \), \( W = \emptyset \) and \( D = \emptyset \).

There is not much work on the repairing of missing is-a structure. Most approaches just add the detected missing is-a relations. This conforms to the solution where \( A = M \). When \( T \cup M \) is consistent and \( \forall p \in M : Or(p) = true \), we are guaranteed that \( M \) is a solution. In the case all missing is-a relations were detected - this is the common case - there are different ways to repair the ontology which are not equally interesting and we can use the earlier defined preference relations.

As these approaches do not deal with correctness, Def. 3 and 6 are not used, and \( \subset_D \) should be removed in Def. 4. In Def. 8 and 10, \( P = \{ \text{more complete}, \subset \} \). In Def. 9, ‘less incorrect’ should be removed. In this case, the semantically maximal solutions in [56] are a special case of the maximally complete repairs where only subsumption axioms between atomic concepts are used. Further, the X-optimal and skyline-optimal repairs combine only completeness and subset minimality.

Interactive solutions to this completion problem have been proposed for taxonomies [7, 57, 58], for \( \mathcal{EL} \) TBoxes [56, 58] and for \( \mathcal{ALC} \) TBoxes [59]. All algorithms compute logically correct solutions which then need to be validated for correctness according to the domain by a domain expert. It is assumed that the axioms \( M \) and \( A \) represent subsumption between atomic concepts in the ontologies. The algorithms for taxonomies and (normalized) \( \mathcal{EL} \) TBoxes (unified notation in [58]) require that \( \forall m \in M : Or(m) = true \), and thus \( M \) is a repair. The algorithms start with a first step that computes skyline-optimal repairs with respect to \{ more complete, \subset \} for each missing is-a relation. This step is different for different representation languages of the TBox. For taxonomies the algorithm tries to find ways to repair a missing is-a relation \( P_1 \subseteq P_2 \) by adding axioms of the form \( P_1^1 \subseteq P_2^1 \) where \( P_1 \subseteq P_1^1 \) and \( P_2^2 \subseteq P_2 \). For \( \mathcal{EL} \), additional is-a relations of the form \( \exists r.P_1 \subseteq \exists r.P_2 \) are repaired by repairing \( P_1 \subseteq P_2 \). For \( \mathcal{EL}^{++} \) also role hierarchies and role inclusions need to be taken into account. Then the algorithms combine and modify these repairs into a single skyline-optimal repair for the whole set of missing is-a relations. Further, the algorithms repeat this process iteratively by solving new completion problems where the new \( M \) is set to the added axioms in \( A \) in the previous iteration. The union of the sets of added axioms of all iterations (with optionally removal of redundancy) is the final repair. It is shown that the skyline-optimal repairs (including the final repair if redundancy is removed) found during the iterations of the new completion problems are skyline-optimal repairs for the original completion problem that are equally or more complete than the repairs found in the first iteration. Complexity results for the existence problem (does a repair exist?), relevance problem (does a repair containing a given axiom exist?) and necessity problem (do all repairs contain a given axiom?) in general and with respect to different preferences are given for \( \mathcal{EL} \) and \( \mathcal{EL}^{++} \) in [56, 58]. In [59] an approach is proposed for \( \mathcal{ALC} \) TBoxes by modifying a tableau-based reasoner.

Reparis are found by closing leaf nodes in the completion graphs generated by trying to disprove missing is-a relations using the tableau reasoner. Open leaf nodes are closed by finding pairs of statements of the form \( x : \mathcal{P} \) and \( x : \neg \mathcal{N} \) and asserting then that \( \mathcal{P} \subseteq \mathcal{N} \). Additionally, the same technique as for taxonomies is applied.

A non-interactive solution, i.e., without validation of an oracle, that is independent of the constructors of the description logic (e.g., tested with ontologies with expressivity up to \( SHOQ^1N(D) \)) is proposed in [60]. In contrast to the previous approaches where the repairs only contain subsumption axioms between existing concepts, this approach introduces justification patterns that can be instantiated with existing concepts.
or ‘fresh’ concepts. Further, the notion of justification pattern-based repairs is introduced which are a kind of repairs that are subset-minimal. Methods for computing all justification patterns as well as justification-based repairs are given.

5.3. Completeness and correctness

There is very little work on dealing with both completeness and correctness. In [24, 25] two versions of the RepOSE system are presented that support debugging and completing the is-a structure of ontologies (and mappings between ontologies) in an iterative and interleaving way. Wrong information is removed by calculating justifications and allowing a user to mark wrong is-a relations. Missing information is added using the interactive techniques in Sect. 5.2. As the system always warns the user of influences of new additions or deletions on previous changes, the system can guarantee a repair if such exists, but it does not always guarantee a skyline-optimal solution.

6. State of the art - ontology networks

Completing and debugging of ontology networks has received more and more attention. Similar to single ontologies, also for networks the quality is dependent on the availability of domain experts, and completely automatic systems may reduce the quality [28].

Our definitions in Sect. 4 and 5 can be used for ontology networks by creating a TBox from the network (i.e., it includes all axioms of all TBoxes from the ontologies and treats all mappings in all alignments in the network as axioms) and using this TBox in the definitions. It also follows that the techniques for single ontologies can be used for ontology networks. However, in much of the current research the axioms in the ontologies in the network and the axioms in the alignments are distinguished and treated differently.

The field of ontology alignment [27] deals with completeness of alignments (and thus only completion of the alignments, not of the ontologies in the networks). Many ontology alignment systems have been developed and overviews can be found in, e.g., [27, 61–66] and at the ontology matching web site (http://www.ontologymatching.org). Usually ontology alignment systems take as input two source ontologies and output an alignment. Systems can contain a pre-processing component that, e.g., partitions the ontologies into mappable parts thereby reducing the search space for finding candidate mappings. Further, a matching component uses matchers that calculate similarities between the entities from the different source ontologies or mappable parts of the ontologies. They often implement strategies based on linguistic matching, structure-based strategies, constraint-based approaches, instance-based strategies, strategies that use auxiliary information or a combination of these. Each matcher utilizes knowledge from one or multiple sources. Candidate mappings are then determined by combining and filtering the results generated by one or more matchers. Common combination strategies are the weighted-sum and the maximum-based strategies. The most common filtering strategy is the threshold filtering. Many systems output the found candidate mappings as an alignment. This approach is mainly a detection approach and the actual repairing is to add the alignment into the network. However, it is well-known that to improve the quality user validation is necessary and several systems allow for user interaction in the different steps of the alignment including validation (see, e.g., overview in [66]). Some systems also allow the addition of partial results to influence the computation of new results and thus a repair can lead to a new detection phase. Some systems introduce other components such as recommendation for the settings for the components in the system. A system that integrates all of these is discussed in [67].

Regarding correctness, most approaches deal with mapping repair where mappings rendering the network incoherent or inconsistent are removed. Usually, the axioms in the ontologies are considered more trustworthy than the mappings and thus mappings are removed, rather than axioms in the ontologies. Although detection of defects can be different for different existing systems, justification-based techniques are often used for the repairing as in [68], and the Radon [69], ALCOMO [70], LogMap [71, 72] and AgreementMakerLight [73] systems. Other heuristics than the ones in Sect. 5 could be used. For instance, the conservativity principle [74] states that the integrated ontology should not induce any change in the concept hierarchies of the input ontologies. In [70, 73, 75] the confidence values of the mappings are taken into account and in [68] a semantic similarity measure between concepts in the mappings is used.

Similar to the case of ontologies, some approaches for ontology networks use a revision approach, e.g., [76–78]. Usually, the ontologies remain the same, but the set of mappings is revised.
An approach that distinguishes between axioms in the ontologies and in the alignments, but gives equal priority to them using approaches in Sect. 5 is discussed in [24, 25].

7. Opportunities

In this paper we have defined a framework for completing and debugging ontologies and shown the state of the art in the field. It is clear that many research opportunities still exist.

7.1. Theory and algorithms

There are still challenges regarding the development of algorithms. Many approaches have been proposed regarding correctness, but finding (preferred) repairs in an acceptable time is still an issue. The current heuristics and optimization are almost all related to logical properties. However, this does not fit non-semantic defects. Furthermore, for semantic defects solutions may be proposed that remove correct statements while there could exist repairs that only remove wrong statements. In this case involving a domain expert in the generation and validation of repairs seems to be a way forward. There is relatively little work on dealing with completeness. There is still a need for new approaches and interactive systems. The current system that allows for user interaction deals with light-weight ontologies, while the work that allows for higher expressivity is non-interactive. Even less work deals with completeness and correctness. We need work on algorithms guaranteeing different kinds of preferred repairs. The current system dealing with both completion and debugging does help a user to find a repair, but it cannot even guarantee to find skyline-optimal repairs.

From the theoretical point of view, there is quite some work on complexity results for debugging and some for completion. However, there is a need for results regarding completion and the combination of debugging and completion as well as results for all cases regarding preferred repairs. In addition to results for finding repairs, there are also the questions related to the checking of the existence of repairs, the relevance of an axiom (is there a repair containing a given axiom?) and the necessity of an axiom (do all repairs add/delete a given axiom?).

The current formalization of repair uses an oracle that replies true or false for an axiom. This means that we assume that an oracle always answers, although the answer may be correct or wrong. However, it is also possible that the oracle does not know an answer. In this case we may want to extend the formalization to allow the oracle to answer unknown. A prudent approach may not allow unknown axioms in the add set of a repair, but they could be allowed in the delete set.

A credulous approach may allow unknown axioms in the add set, but not in the delete set. Further, preferences may be defined related to the use of unknown axioms.

In some cases, for instance, when there is a consensus about some concepts and their relations to each other, we may want to state that certain parts of the ontology are correct and should not be changed. This would then restrict repairing of defects to not include these parts. This can be handled by extending the formalization of the complete-debug problem by using the notion of background knowledge which represents information about the relevance or importance of parts of the ontology ([30]). In [49] background knowledge is used to represent parts of the ontology that are asserted to be correct and therefore should not be changed in debugging. This is an example of the requirement called exploitation of context in [30].

In this survey a repair consists of set of axioms that are added and a set of axioms that are deleted from the ontology (Def. 1). However, by removing axioms sometimes correct inferences are lost. Therefore, instead of removing complete axioms we may want to weaken an axiom or rewrite an axiom such that only the parts that caused a defect are removed. Ways to formalize what a part of an axiom is in this sense, together with debugging algorithms, are presented in [79, 80]. A method that rewrites the axioms in the ontology into simpler axioms and debug these is shown in [44]. In [81] parts of axioms responsible for unsatisfiability of concepts are traced. Further, lost inferences are calculated for atomic concepts. Possible axiom changes can be tagged as harmful when they do not solve unsatisfiability or introduce new unsatisfiability. The authors also introduce the notion of helpful changes where part of an axiom is removed, but other axioms are added to make up for lost inferences.

An approach for dealing with defects in incoherent or inconsistent ontologies by changing axioms is axiom weakening where an axiom is replaced by another axiom that has fewer consequences subset-wise. Different ways to compute such weaker axioms are presented in [82, 83]. While the current work deals with debugging, there may be cases for completion as well where we essentially would like to modify existing axioms.
In principle, all these cases are already covered by the current framework as a change in an axiom can be represented by the removal of the original axiom together with the addition of the changed axiom. However, this would require solutions for the full problem in Def. 1 (which, as we have seen, are not available yet) and we may want to introduce new preference relations based on additions and deletions that reflect axiom changes.

In this survey we have focused on ontologies that do not contain instances. When instances are available these could be used in detection or repairing [46, 84–86]. For instance, using instances is one of the main ways to detect inconsistent ontologies.

7.2. User support

From a practical point of view it is clear that to obtain the best results of the completion and debugging, domain experts need to be involved. However, the support of the current tools regarding user involvement is still lacking. For instance, one of the outcomes in a study with users on ontology authoring is that debugging is difficult [87]. Although ontology development systems may have explanation facilities, debugging is still cumbersome and ontology developers use their own strategies, such as running a reasoner frequently when adding new axioms to the ontology to detect possible defects. The authors also state that, although good work has been done in ontology debugging, not that much has been integrated in ontology development tools. According to the study SWOOP had good debugging support, Protégé had explanation facilities, WebProtégé and the free edition of TopBraid did not have such support.

There are thus several opportunities for further work on user involvement in completion and debugging systems. One way forward is to consult the guidelines for ontology alignment systems that are also valid for the more general completion and debugging systems. Recommendations for user support for ontology alignment systems regarding user interfaces (partly based on [88]) as well as infrastructure and algorithms are given in [89]. The former include support for manipulation, inspection and explanation of mappings, while the latter include, among others, support for sessions, reduction of user interventions, collaboration, recommendations by the system, system configuration, debugging, trial execution and temporary decisions. In [90] the authors focus more specifically on user validation in ontology alignment and discuss issues related to the profile of the user, the services of the alignment system, and its user interface. Recommendations for tool support for each aspect are given and the support in current systems is presented. For the user profile the authors discuss user expertise in terms of domain, technology and alignment systems. For services they discuss stage of involvement, feedback demand and feedback propagation. Finally, for the user interface issues regarding visualization and interaction are discussed.

The paper reports that while there have been significant advances on the part of alignment systems in these areas, there are still key challenges to overcome such as reducing user workload, balancing informativeness with cognitive load and balancing user workload with user errors. More generally, the field of visualization and interaction for ontologies and Linked Data has received more and more attention during the recent years and in [91] the issues of cognitive support, user profiles and visual exploratory analysis are briefly discussed.

There are also some specific problems that have been noted in the debugging and completion area.

An important issue is that domain experts make mistakes and thus the oracle makes mistakes (see third case in Sect. 4.1.2). This has been reported often, and for instance, the study in [92] states that questions to the oracle about statements that are true receive more reliable answers, that domain experts are sometimes overconfident, and that they consider themselves as imperfect. Some research has discussed approaches to deal with this issue. For instance, in [93] an approach (for ontologies with instances) is presented where the domain expert can request the history of the given answers, correct wrongly given answers and continue, while in [92] a prediction model is developed for predicting oracle errors. The impact of oracle errors on the effectiveness and efficiency of ontology alignment systems are shown in [90] as assessed in the Interactive Anatomy track of the Ontology Alignment Evaluation Initiative 2015 - 2018.

Another issue that has been mentioned is that some algorithms work on normalized versions of a TBox and therefore the results may not be that intuitive in terms of the original ontology. This requirement is called preservation of structure in [30].

Further, in [93] it was reported that domain experts would want to be able to sometimes postpone their answers as an oracle, e.g., to have the time to check up some information or reflect more deeply.

More generally, completion and debugging should be integrated in every ontology development methodology, such that developers can detect and repair defects as soon as possible. One of the few that has differ-
ent steps regarding completion and debugging within the general framework is an extension of the eXtreme Design Methodology [94].

8. Conclusion

As semantically-enabled applications require high-quality ontologies, developing and maintaining as correct and complete as possible ontologies is an important, although difficult task in ontology engineering. A key step for guaranteeing a certain level of correctness and completeness is ontology debugging and completion.

In this survey paper we have reviewed the state of the art in ontology debugging and completion where we have focused on the repairing step. We have done this by introducing a formalization for the completion and debugging problem which allowed us to review and discuss the state-of-the-art in this field in a uniform way. Using this formalization we show that, traditionally, debugging and completion have been tackled separately, we compared different approaches, and gained insights in the field and we point to new opportunities for further research to advance the field.

Acknowledgements

We thank Mina Abd Nikooie Pour, Ying Li and Chunyan Wang for proofreading and checking the examples. This work has been financially supported by the Swedish e-Science Research Centre (SeRC), the Swedish Research Council (Vetenskapsrådet, dnr 2018-04147) and the Horizon 2020 project SPIRIT (grant agreement No 786993).

References


Appendix

In Fig. 4 (and visually in Fig. 5) we show an example of a complete-debug-problem inspired by the Galen ontology (http://www.openclinical.org/prj_galen.html) in the $\mathcal{EL}$ language. For the ontology represented by the TBox we assume that we have detected two missing axioms Endocarditis $\sqsubseteq$ PathologicalPhenomenon and GranulomaProcess $\sqsubseteq$ NonNormalProcess. Further, we have detected that we can infer the wrong axiom PathologicalProcess $\sqsubseteq$ GranulomaProcess from the ontology.

Repairs

There are different possible repairs of which we show some. Fig. 6 shows the ontology after repair R1 is executed, where $R1 = (A_1, D_1)$ with $A_1 = \{ \text{Endocarditis} \sqsubseteq \text{PathologicalPhenomenon}, \text{GranulomaProcess} \sqsubseteq \text{NonNormalProcess} \}$ and $D_1 = \{ \text{PathologicalProcess} \sqsubseteq \text{InflammationProcess} \}$. We show that R1 is a repair by discussing the different criteria in the definition of repair. (i) Regarding the statements in $A_1$, we know that $Or(\text{Endocarditis} \sqsubseteq \text{PathologicalPhenomenon}) = true$, and $Or(\text{GranulomaProcess} \sqsubseteq \text{NonNormalProcess}) = true$. (ii) For the statement in $D_1$, we know that $Or(\text{PathologicalProcess} \sqsubseteq \text{InflammationProcess}) = false$. (iii) $(T \cup A_1) \setminus D_1$ is consistent as $\mathcal{EL}$ Tboxes are consistent. (iv) $(T \cup A_1) \setminus D_1 \models \text{Endocarditis} \sqsubseteq \text{PathologicalPhenomenon}$, as $\text{Endocarditis} \sqsubseteq \text{PathologicalPhenomenon}$ has been explicitly added. $(T \cup A_1) \setminus D_1 \models \text{GranulomaProcess} \sqsubseteq \text{NonNormalProcess}$, as $\text{GranulomaProcess} \sqsubseteq \text{NonNormalProcess}$ has been explicitly added. (v) $(T \cup A_1) \setminus D_1 \not\models \text{PathologicalProcess} \sqsubseteq \text{GranulomaProcess}$. The only justification of $\text{PathologicalProcess} \sqsubseteq \text{GranulomaProcess}$ in the original ontology was $\{ \text{PathologicalProcess} \sqsubseteq \text{InflammationProcess}, \text{InflammationProcess} \sqsubseteq \text{GranulomaProcess} \}$ and this is not available anymore after the repair as $\text{PathologicalProcess} \sqsubseteq \text{InflammationProcess}$ is removed. The newly added axioms do

References


M = { Endocarditis ⊑ PathologicalPhenomenon, GranulomaProcess ⊑ NonNormalProcess } 

W = { PathologicalProcess ⊑ GranulomaProcess } 

The following axioms are correct according to the domain, i.e., $\text{Or}$ returns true for:

- GranulomaProcess ⊑ InflammationProcess,
- GranulomaProcess ⊑ PathologicalProcess,
- GranulomaProcess ⊑ NonNormalProcess,
- InflammationProcess ⊑ PathologicalProcess,
- InflammationProcess ⊑ NonNormalProcess,
- CardioVascularDisease ⊑ PathologicalPhenomenon,
- Fracture ⊑ PathologicalPhenomenon,
- Endocarditis ⊑ PathologicalPhenomenon,
- Endocarditis ⊑ Carditis,
- Endocarditis ⊑ CardioVascularDisease,
- Carditis ⊑ PathologicalPhenomenon,
- Carditis ⊑ CardioVascularDisease,
- Endocarditis ⊑ PathologicalPhenomenon,
- Endocarditis ⊑ PathologicalPhenomenon, 
- Endocarditis ⊑ PathologicalPhenomenon,
- Endocarditis ⊑ PathologicalPhenomenon.

Note that for an oracle that does not make mistakes, $\text{Or}(P ⊑ Q) = true$ then also $\text{Or}(∃ r.P ⊑ ∃ r.Q)=true.$

For other axioms $P ⊑ Q$ with $P, Q ∈ C$, $\text{Or}(P ⊑ Q) = false.$

Fig. 4. $\mathcal{EL}$ example. ($C$ is the set of atomic concepts in the ontology. $T$ is a TBox representing the ontology. $M$ is a set of missing axioms. $W$ is the set of wrong axioms. $\text{Or}$ is the oracle representing the domain expert.)

not give rise to new justifications for PathologicalProcess ⊑ GranulomaProcess.

Fig. 7 shows the ontology after repair R2 is executed, where $R2 = (A_2, D_2)$ with $A_2 = A_1 = \{ \text{Endocarditis} ⊑ \text{PathologicalPhenomenon}, \text{GranulomaProcess} ⊑ \text{NonNormalProcess} \}$ and $D_2 = \{ \text{InflammationProcess} ⊑ \text{GranulomaProcess} \}$. We show that $R2$ is a repair. (i) Same as R1. (ii) For the statement in $D_2$, we know that $\text{Or}(\text{InflammationProcess} ⊑ \text{GranulomaProcess}) = false$. (iii) Same as R1. (iv) Same as R1. (v) $\text{Or}(T ∪ A_2) \setminus D_2 = \text{PathologicalProcess} ⊑ \text{GranulomaProcess}$. The only justification of PathologicalProcess ⊑ GranulomaProcess in the original ontology was [ PathologicalProcess ⊑ InflammationProcess, InflammationProcess ⊑ GranulomaProcess ] and this is not available anymore after the repair as InflammationProcess ⊑ GranulomaProcess is removed. The newly
Fig. 5. Visualization of the example complete-debug problem in Fig. 4. The axioms in the Tbox are represented with black arrows. The detected missing axioms are represented in blue. The detected wrong axiom is represented in red. The oracle’s knowledge about the axioms in the ontology is marked with T (true) or F (false) at the arrows.

Fig. 6. Result after repairing the ontology in the complete-debug problem in Fig. 4 with repair R1. R1 = (A₁,D₁) with A₁ = { Endocarditis ⊑ PathologicalPhenomenon, GranulomaProcess ⊑ NonNormalProcess }, D₁ = { PathologicalProcess ⊑ InflammationProcess }.

added axioms do not give rise to new justifications for PathologicalProcess ⊑ GranulomaProcess.

Fig. 8 shows the ontology after repair R3 is executed, where R3 = (A₃,D₃) with A₃ = A₁ = A₂ = { Endocarditis ⊑ PathologicalPhenomenon, GranulomaProcess ⊑ NonNormalProcess } and D₃ = D₁ ∪ D₂ = { PathologicalProcess ⊑ InflammationProcess, InflammationProcess ⊑ GranulomaProcess }. We show that R3 is a repair. (i) Same as R1. (ii) For the statements in D₃, we know that Or( PathologicalProcess ⊑ InflammationProcess ) = false and Or( InflammationProcess ⊑ GranulomaProcess ) = false. (See R1 and R2.) (iii) Same as R1. (iv) Same as R1. (v) (T ∪ A₃) \ D₃ \̸= PathologicalProcess ⊑ GranulomaProcess. The only justification of PathologicalProcess ⊑ GranulomaProcess in the original ontology was { PathologicalProcess ⊑ InflammationProcess, InflammationProcess ⊑ GranulomaProcess } and this is not available anymore after the repair as the axioms in the justification have been removed. The newly added axioms do not give rise to new justifications for PathologicalProcess ⊑ GranulomaProcess.

Fig. 9 shows the ontology after repair R4 is executed, where R4 = (A₄,D₄) with A₄ = { Endocardi-
Fig. 7. Result after repairing the ontology in the complete-debug problem in Fig. 4 with repair R2. R2 = (A₂, D₂) with
A₂ = { Endocarditis ⊑ PathologicalPhenomenon, GranulomaProcess ⊑ NonNormalProcess },
D₂ = { InflammationProcess ⊑ GranulomaProcess }.

Fig. 8. Result after repairing the ontology in the complete-debug problem in Fig. 4 with repair R3. R3 = (A₃, D₃) with
A₃ = { Carditis ⊑ PathologicalPhenomenon, GranulomaProcess ⊑ PathologicalProcess } and D₃ = { PathologicalProcess ⊑ InflammationProcess, InflammationProcess ⊑ GranulomaProcess }. We show that R4 is a repair. (i) Regarding the statements in A₄, we know that
Or(Endocarditis ⊑ PathologicalPhenomenon) = true, and
Or(GranulomaProcess ⊑ PathologicalProcess) = true. (ii) Same as R3. (iii) Same as R1. (iv) (T ∪ A₄) \ D₄ ⊨ Endocarditis ⊑ PathologicalPhenomenon, as Endocarditis ⊑ PathologicalPhenomenon has been explicitly added. (T ∪ A₄) \ D₄ ⊨ GranulomaProcess ⊑ NonNormalProcess, as this can be derived from the added GranulomaProcess ⊑ PathologicalProcess and the axiom PathologicalProcess ⊑ NonNormalProcess that was already in the TBox. (v) Same as R3.

Fig. 10 shows the ontology after repair R5 is executed, where R5 = (A₅, D₅) with A₅ = { Carditis ⊑ PathologicalPhenomenon, GranulomaProcess ⊑ PathologicalProcess } and D₅ = D₄ = { PathologicalProcess ⊑ InflammationProcess, InflammationProcess ⊑ GranulomaProcess }. We show that R5 is a repair. (i) Regarding the statements in A₅, we know that
Or(Carditis ⊑ PathologicalPhenomenon) = true, and
Or(GranulomaProcess ⊑ PathologicalProcess) = true.
Fig. 9. Result after repairing the ontology in the complete-debug problem in Fig. 4 with repair $R_4$. $R_4 = (A_4, D_4)$ with $A_4 = \{ \text{Endocarditis} \sqsubseteq \text{PathologicalPhenomenon}, \text{GranulomaProcess} \sqsubseteq \text{PathologicalProcess} \}$, $D_4 = \{ \text{PathologicalProcess} \sqsubseteq \text{InflammationProcess}, \text{InflammationProcess} \sqsubseteq \text{GranulomaProcess} \}$.

Fig. 10. Result after repairing the ontology in the complete-debug problem in Fig. 4 with repair $R_5$. $R_5 = (A_5, D_5)$ with $A_5 = \{ \text{Carditis} \sqsubseteq \text{PathologicalPhenomenon}, \text{GranulomaProcess} \sqsubseteq \text{PathologicalProcess} \}$, $D_5 = \{ \text{PathologicalProcess} \sqsubseteq \text{InflammationProcess}, \text{InflammationProcess} \sqsubseteq \text{GranulomaProcess} \}$.

(ii) Same as $R3$. (iii) Same as $R1$. (iv) $(T \cup A_6) \succeq D_6$ $\models \text{Endocarditis} \sqsubseteq \text{PathologicalPhenomenon}$, as this can be derived from the axiom $\text{Endocarditis} \sqsubseteq \text{Carditis}$ that was already in the TBox and the added axiom $\text{Carditis} \sqsubseteq \text{PathologicalPhenomenon}$. $(T \cup A_3) \succeq D_3$ $\models \text{GranulomaProcess} \sqsubseteq \text{NonNormalProcess}$, for the same reason as in $R4$. (v) Same as $R3$.

Fig. 11 shows the ontology after repair $R6$ is executed, where $R6 = (A_6, D_6)$ with $A_6 = \{ \text{Carditis} \sqsubseteq \text{CardioVascularDisease}, \text{GranulomaProcess} \sqsubseteq \text{PathologicalProcess} \}$ and $D_6 = D_3 = D_4 = D_5 = \{ \text{PathologicalProcess} \sqsubseteq \text{InflammationProcess}, \text{InflammationProcess} \sqsubseteq \text{GranulomaProcess} \}$. We show that $R6$ is a repair. (i) Regarding the statements in $A_6$, we know that $Or(\text{Carditis} \sqsubseteq \text{CardioVascularDisease}) = true$, and $Or(\text{GranulomaProcess} \sqsubseteq \text{PathologicalProcess}) = true$. (ii) Same as $R3$. (iii) Same as $R1$. (iv) $(T \cup A_6) \succeq D_6$ $\models \text{Endocarditis} \sqsubseteq \text{PathologicalPhenomenon}$, as this can be derived from the existing axiom $\text{Endocarditis} \sqsubseteq \text{Carditis}$, the added axiom $\text{Carditis} \sqsubseteq \text{CardioVascularDisease}$ and the existing axiom $\text{CardioVascularDisease} \sqsubseteq \text{PathologicalPhenomenon}$. $(T \cup A_6) \succeq D_6$ $\models \text{GranulomaProcess} \sqsubseteq \text{NonNormalProcess}$, for the same reason as in $R4$. (v) Same as $R3$. 

P. Lamrix / Repairing Ontologies
Fig. 11. Result after repairing the ontology in the complete-debug problem in Fig. 4 with repair R6. $R6 = (A_6, D_6)$ with $A_6 = \{ \text{Carditis} \sqsubseteq \text{CardioVascularDisease}, \text{GranulomaProcess} \sqsubseteq \text{PathologicalProcess} \}$, $D_6 = \{ \text{PathologicalProcess} \sqsubseteq \text{InflammationProcess}, \text{InflammationProcess} \sqsubseteq \text{GranulomaProcess} \}$.

Fig. 12. Result after repairing the ontology in the complete-debug problem in Fig. 4 with repair R7. $R7 = (A_7, D_7)$ with $A_7 = \{ \text{GranulomaProcess} \sqsubseteq \text{InflammationProcess}, \text{InflammationProcess} \sqsubseteq \text{PathologicalProcess} \}$, $D_7 = \{ \text{PathologicalProcess} \sqsubseteq \text{InflammationProcess}, \text{InflammationProcess} \sqsubseteq \text{GranulomaProcess} \}$.

Fig. 12 shows the ontology after repair R7 is executed, where $R7 = (A_7, D_7)$ with $A_7 = \{ \text{GranulomaProcess} \sqsubseteq \text{InflammationProcess}, \text{InflammationProcess} \sqsubseteq \text{PathologicalProcess} \}$ and $D_7 = D_3 = D_4 = D_5 = D_6 = \{ \text{PathologicalProcess} \sqsubseteq \text{InflammationProcess}, \text{InflammationProcess} \sqsubseteq \text{GranulomaProcess} \}$. We show that R7 is a repair. (i) Regarding the statements in $A_7$, we know that $Or(\text{GranulomaProcess} \sqsubseteq \text{InflammationProcess}) = true$, and $Or(\text{InflammationProcess} \sqsubseteq \text{PathologicalProcess}) = true$. (ii) Same as R3. (iii) Same as R1. (iv) $(T \cup A_7) \setminus D_7 \models \text{Endocarditis} \sqsubseteq \text{PathologicalPhenomenon}$, as this can be derived from the existing axiom $\exists \text{hasAssociatedProcess.InflammationProcess}$, the newly derivable axiom $\exists \text{hasAssociatedProcess.InflammationProcess}$, and the existing axiom $\exists \text{hasAssociatedProcess.PathologicalProcess}$. PathologicalProcess can be derived from InflammationProcess $\sqsubseteq$ PathologicalProcess which is newly added. $(T \cup A_7) \setminus D_7 \models \text{GranulomaProcess} \sqsubseteq \text{NonNormalProcess}$, as it can be derived from the two newly added axioms Granu-

Fig. 13 shows the ontology after repair R8 is executed, where R8 = (A8,D8) with A8 = { Carditis ⊑ CardioVascularDisease, GranulomaProcess ⊑ InflammationProcess, InflammationProcess ⊑ PathologicalProcess } and D8 = D1 = D2 = D3 = D7 = { PathologicalProcess ⊑ InflammationProcess, InflammationProcess ⊑ GranulomaProcess }. We show that R8 is a repair. (i) Regarding the statements in A8, we know that Or(Carditis ⊑ CardioVascularDisease) = true, Or(GranulomaProcess ⊑ InflammationProcess) = true, and Or(InflammationProcess ⊑ PathologicalProcess) = true. (ii) Same as R3. (iii) Same as R1. (iv) (T ∪ A8) \ D8 |= Endocarditis ⊑ PathologicalPhenomenon, as this can be derived from the existing axiom Endocarditis ⊑ PathologicalPhenomenon. The axiom Endocarditis ⊑ PathologicalPhenomenon can be derived from InflammationProcess disputed by the repair. The axiom PathologicalProcess ⊑ PathologicalProcess can be derived from InflammationProcess disputed by the repair. The axiom PathologicalProcess ⊑ PathologicalProcess which is newly added. (The missing axiom can also be derived from the existing axiom Endocarditis ⊑ Carditis, the added axiom Carditis ⊑ CardioVascularDisease and the existing axiom CardioVascularDisease ⊑ PathologicalPhenomenon.) (T ∪ A8) \ D8 |= GranulomaProcess ⊑ NonNormalProcess, as it can be derived from the two newly added axioms GranulomaProcess ⊑ InflammationProcess and InflammationProcess ⊑ PathologicalProcess, together with the existing axiom PathologicalProcess ⊑ NonNormalProcess. (v) Same as R3.

Less incorrect

R3, R4, R5, R6, R7 and R8 are less incorrect than R1 and R2, as R3, R4, R5, R6, R7 and R8 do not contain any wrong axioms but R1 contains the wrong axiom InflammationProcess ⊑ GranulomaProcess, and R2 contains the wrong axiom PathologicalProcess ⊑ InflammationProcess. R3, R4, R5, R6, R7 and R8 are all equally incorrect.

More complete

Let R_{\text{ext}} be the set of all correct axioms that can be derived from all repaired ontologies by repairs R1-R8 respectively. (If P ⊑ Q can be derived, then also ∃r.P ⊑ ∃r.Q. If P ⊑ Q, then also P ∩ Q ⊑ Q. We do not add these statements nor tautologies.) As none of these repairs has taken away a correct axiom from the ontology, all original correct axioms are in R_{\text{ext}}. Furthermore, all repairs need to make sure the axioms in M are derivable and thus also these are in R_{\text{ext}}. Then, R_{\text{ext}} = { CardioVascularDisease ⊑ PathologicalPhenomenon, Fracture ⊑ PathologicalPhenomenon, ∃hasAssociatedProcess.PathologicalPhenomenon, PathologicalProcess ⊑ PathologicalPhenomenon, Endocarditis ⊑ CardioVascularDisease, Endocarditis ⊑ ∃hasAssociatedProcess.PathologicalPhenomenon, PathologicalProcess, PathologicalProcess ⊑ NonNormalProcess, GranulomaProcess ⊑ NonNormalProcess, Endocarditis ⊑ PathologicalPhenomenon }.

The set of axioms derivable from the ontology repaired by R1 is R_{\text{ext}} = R_{\text{ext}} ∪ { InflammationProcess ⊑ NonNormalProcess }. This additional axiom for the ontology repaired by R1 can be derived from (the wrong axiom) InflammationProcess ⊑ GranulomaProcess and the added axiom GranulomaProcess ⊑ NonNormalProcess.

Fig. 13. Result after repairing the ontology in the complete-debug problem in Fig. 4 with repair R8. $R_8 = (A_8, D_8)$ with $A_8 = \{ \text{Carditis} \sqsubseteq \text{CardioVascularDisease}, \text{GranulomaProcess} \sqsubseteq \text{InflammationProcess}, \text{InflammationProcess} \sqsubseteq \text{PathologicalProcess} \}$. $D_8 = \{ \text{PathologicalProcess} \sqsubseteq \text{InflammationProcess}, \text{InflammationProcess} \sqsubseteq \text{GranulomaProcess} \}$. VascularDisease, Endocarditis $\sqsubseteq$ CardioVascularDisease, $\exists$\text{hasAssociatedProcess.InflammationProcess} $\sqsubseteq$ PathologicalPhenomenon, Endocarditis $\sqsubseteq$ $\exists$\text{hasAssociatedProcess.PathologicalProcess}. PathologicalPhenomenon $\sqsubseteq$ NonNormalProcess.

Thus, $R_{ax} = R_{2ax} \sqsubset R_{3ax} \sqsubseteq R_{1ax} \sqsubseteq R_{7ax} \sqsubseteq R_{8ax}$. From this we conclude that $R_8$ is more complete than $R_7$, which is more complete than $R_1$ which is more complete than $R_2$ and $R_3$. Further, $R_8$ is more complete than $R_6$ which is more complete than $R_5$ which is more complete than $R_4$ which is more complete than $R_2$ and $R_3$.

Subset

For the add and delete sets for the repairs $R_1$-$R_8$ we obtain the following:

$A_1 = A_2 = A_3, A_7 \subseteq A_8, D_1 \sqsubset D_3 = D_4 = D_5 = D_6 = D_7 = D_8$, and $D_2 \sqsubset D_3 = D_4 = D_5 = D_6 = D_7 = D_8$. Therefore, $R_1 \sqsubset R_3, R_2 \sqsubset R_3$ and $R_7 \sqsubset R_8$. $R_3$ deletes more wrong information than $R_1$ and $R_2$, respectively, but for repairing this is redundant (although $R_3$ is less incorrect than $R_1$ and $R_2$). $R_8$ adds additional correct axioms compared to $R_7$, but this is redundant for the repairing (although $R_8$ is more complete than $R_7$).

Preferred repairs

$R_8$ is maximally complete as the ontology repaired by $R_8$ contains all correct information.

R3, R4, R5, R6, R7 and R8 are minimally incorrect as the ontologies repaired by any of these repairs do not contain incorrect axioms. $R_1$ and $R_2$ are subset minimal as if we remove an axiom from the add or delete set we would not have a repair anymore. None of the other repairs is subset minimal as there are always variants where we can remove one of the axioms in the delete sets and still have repairs.

Combined preferences

According to the definition, only preferred repairs with respect to a preference $X$ can be $X$-optimal. Regarding more complete-optimal the only candidate among our example repairs is $R_8$. $R_8$ is more complete-optimal with respect to $\{ \text{less incorrect} \}$ as the ontology repaired by $R_8$ does not contain wrong information. It is not more complete-optimal with respect to $\{ \subset \}$ as there are repairs that are also preferred with respect to more complete, but that remove one fewer wrong axiom. However, $R_8$ is more complete-optimal with respect to $\{ \text{less incorrect}, \subset \}$. If there would be a preferred repair with respect to more complete that dominates $R_8$ with respect to $\{ \text{less incorrect}, \subset \}$, then it would have to be more preferred to $R_8$ with respect to $\subset$ as it cannot be more preferred with respect to less incorrect. However, removing an added axiom from $R_8$ would make the repair not preferred with respect to more complete and removing fewer deleted axioms would make the repair less incorrect than $R_8$. 

Thus, $R_{ax} = R_{2ax} \sqsubset R_{3ax} \sqsubseteq R_{1ax} \sqsubseteq R_{7ax} \sqsubseteq R_{8ax}$. From this we conclude that $R_8$ is more complete than $R_7$ which is more complete than $R_1$ which is more complete than $R_2$ and $R_3$. Further, $R_8$ is more complete than $R_6$ which is more complete than $R_5$ which is more complete than $R_4$ which is more complete then $R_2$ and $R_3$.

Subset

For the add and delete sets for the repairs $R_1$-$R_8$ we obtain the following:

$A_1 = A_2 = A_3, A_7 \subseteq A_8, D_1 \sqsubset D_3 = D_4 = D_5 = D_6 = D_7 = D_8$, and $D_2 \sqsubset D_3 = D_4 = D_5 = D_6 = D_7 = D_8$. Therefore, $R_1 \sqsubset R_3, R_2 \sqsubset R_3$ and $R_7 \sqsubset R_8$. $R_3$ deletes more wrong information than $R_1$ and $R_2$, respectively, but for repairing this is redundant (although $R_3$ is less incorrect than $R_1$ and $R_2$). $R_8$ adds additional correct axioms compared to $R_7$, but this is redundant for the repairing (although $R_8$ is more complete than $R_7$).

Preferred repairs

$R_8$ is maximally complete as the ontology repaired by $R_8$ contains all correct information.
Regarding less incorrect-optimal the candidates among our repairs are R3, R4, R5, R6, R7 and R8. R8 is less incorrect-optimal with respect to \{ more complete \}. Similar reasoning as above leads to the fact that R8 is less incorrect-optimal with respect to \{ more complete, \subset \}. As R8 dominates R3, R4, R5, R6 and R7 with respect to more complete, R3, R4, R5, R6 and R7 cannot be less incorrect-optimal with respect to \{ more complete \}. R7 dominates R8 with respect to \{ \subset \}, so R8 cannot be less incorrect-optimal with respect to \{ \subset \}. A repair that would be preferred with respect to less incorrect and dominate R3, R4, R5, R6 or R7 with respect to \{ \subset \} would need to remove the two wrong axioms (to be preferred with respect to less incorrect) and would therefore need to add fewer (subset-wise) axioms. However, removing one of the added axioms in R3, R4, R5, R6 and R7 would lead to sets of axioms that are not a repair. Therefore, R3, R4, R5, R6 and R7 are less incorrect-optimal with respect to \{ \subset \}.

Regarding \subset \text{-optimal the candidates are R1 and R2. R1 is more complete than R2, so R2 cannot be \subset \text{-optimal with respect to \{ more complete \}. Also R1 is not \subset \text{-optimal with respect to \{ more complete \}} as there is another subset minimal solution that dominates R1 with respect to \{ more complete \} (e.g. same delete set as R1, but Carditis \subset CardioVascularDisease in the add set instead of Endocarditis \subset PathologicalPhenomenon). For similar reasons R1 and R2 are not \subset \text{-optimal with respect to \{ more complete, less incorrect \}. They are, however, \subset \text{-optimal with respect to \{ less incorrect \}. A less incorrect repair than R1 or R2 would need to remove both wrong axioms, but would then not be subset minimal.

We note that all preferred repairs are skyline optimal. Further, if a repair is X-optimal with respect to \mathcal{P}, then it is skyline-optimal with respect to \mathcal{P} \cup X. Thus, R8 is skyline-optimal with respect to \{ more complete, less incorrect \} and \{ more complete, less incorrect, \subset \}. R1, R2, R3, R4, R5, R6 and R7 are skyline-optimal with respect to \{ less incorrect, \subset \}.

In addition there are also skyline-optimal repairs that are not X-optimal. For instance, R1 is not more-complete-optimal with respect to \{ \subset \} nor \subset \text{-optimal with respect to \{ more complete \}. However, R1 is skyline-optimal with respect to \{ more complete, \subset \}. If there would be a repair that dominates R1 with respect to \{ more complete, \subset \}, then there are two possibilities. The first possibility is that the other repair is more preferred with respect to \subset, which would mean taking away an axiom in the add set or in the delete set, but then we do not have a repair. The second possibility is that the other repair is more preferred with respect to more complete and equally preferred with respect to \subset. The second condition would only be satisfied if the add and delete sets are the same, but then we have the same repair.