Completing and Debugging Ontologies: State-of-the-art and Challenges in Repairing Ontologies

PATRICK LAMBRIX, Linköping University and University of Gävle, Sweden

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 task is ontology debugging and completion. In general, there are two steps: detecting defects and repairing defects. In this article, we discuss the state-of-the-art regarding the repairing step. We do this by formalizing the repairing step as an abductive reasoning 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.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**; **Knowledge representation and reasoning**; **Description logics;**

Additional Key Words and Phrases: Ontology engineering, ontology debugging, ontology completion, ontology alignment

ACM Reference format:
Patrick Lambrix. 2023. Completing and Debugging Ontologies: State-of-the-art and Challenges in Repairing Ontologies. *ACM J. Data Inform. Quality* 15, 4, Article 41 (October 2023), 38 pages.
https://doi.org/10.1145/3597304

1 INTRODUCTION

Ontologies (e.g., Reference [103]) 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 [109]. Ontologies are also a key technology for the semantic web.

In recent years many ontologies have been developed (see Reference [18] for a survey on ontology libraries). Further, ontologies have been connected to each other using mappings resulting into ontology networks, and there are some portals that store these ontology networks (e.g.,

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).
Author's address: P. Lambrix, Department of Computer and Information Science, Linköping University, SE-58183 Linköping, Sweden; email: patrick.lambrix@liu.se.

1936-1955/2023/10-ART41 $15.00
https://doi.org/10.1145/3597304

BioPortal[1] and Unified Medical Language System[2]). However, developing ontologies and networks are not easy tasks, and there may be issues related to the quality of the ontologies and networks [100]. Two such issues are incorrectness (does the ontology contain wrong information?) and completeness (is information lacking?). Ontologies containing wrong information or lacking information, nevertheless 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 Reference [64] it was shown that semantically enabled querying of PubMed[3] using **MeSH (Medical Subject Headings)**[4] with one piece of information missing (i.e., that scleritis is a scleral disease) would lead to missing 55% of the documents that were obtained as a query result with this piece of information. Therefore, it is essential to complete and debug ontologies and their networks.

Defects in ontologies can take the forms of syntactic, semantic, and modeling defects (e.g., Reference [50]). 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 in the domain. For instance, the **Ontology Alignment Evaluation Initiative (OAEI)**[5] is a yearly event for the evaluation of ontology alignment systems. In the Anatomy track of OAEI the Adult Mouse Anatomy (2,744 concepts) and the NCI Thesaurus (3,304 concepts) need to be aligned. In Reference [58] it was shown that at least 121 and 83, respectively, is-a relations (defined below) that are correct in the domain are missing in these ontologies. 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 Reference [50] it was shown that the TAMBIS ontology contained 144 unsatisfiable concepts. An example of the occurrence of defects in ontology networks is, for instance, that BioPortal contains mapping results for the OAEI Anatomy track produced by ontology alignment systems that contain ca. 20% incorrect mappings and ca. 20% of the mappings between the ontologies are missing.

It is beneficial to deal with defects already during the development of the ontologies. In Reference [22] an extension of the eXtreme Design Methodology was proposed that introduces explicit steps to deal with defects during the *test and revise*, *integrate,* and *evaluate and revise* activities. It was shown that for a use case with an integrated ontology based on 51 small ontologies, due to completion, out of 197 named concepts in the integrated ontology, 56 received new subconcepts and/or new superconcepts. Further, 3 subsumption relations were removed that affected the sub- and superconcepts of 16 concepts as well as the concepts for which these concepts where in the domain or range of some relation. The use case contained 88 instances, of which 57 were affected by the changes due to debugging and completion. The changes to the ontology affected also 5,402 out of 10,742 queries in the log files of the use case's application.

In this article, we review approaches for improving the quality of ontologies by repairing them. In general, completing and debugging requires two steps.[6] In the detection step, possible defects are detected and localized. In the repairing step, the detected wrong information is removed and

[1] http://bioportal.bioontology.org/.

[2] http://www.nlm.nih.gov/research/umls/about_umls.html.

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

[4] http://www.nlm.nih.gov/mesh/.

[5] http://oaei.ontologymatching.org/.

[6] We note that there is no standardized terminology for these steps, and other terminology may be used in different research areas. For instance, in model-based diagnosis research, three steps are introduced: fault detection (checking whether there

missing information is added. In this article, we focus on the repairing step for which there are still many open research problems. We provide a framework that formally defines the problem and reasonable preference relations between repairs and that can be used for comparison of repairing approaches.

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

2 PRELIMINARIES

2.1 Ontologies and Ontology Networks

Intuitively, ontologies can be seen as defining the basic terms and relations of a domain of interest, as well as the rules for combining these terms and relations. Ontologies are used for communication between people and organizations by providing a common terminology over a domain. They provide the basis for interoperability between systems and can be used as an index to a repository of information as well as a query model and a navigation model for data sources. They are often used as a basis for integration of data sources, thereby alleviating the variety and variability problems. The benefits of using ontologies include reuse, sharing, and portability of knowledge across platforms and improved maintainability, documentation, maintenance, and reliability. Overall, ontologies lead to a better understanding of a field and to more effective and efficient handling of information in that field (e.g., Reference [104]).

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 $\sqsubseteq$. When $P \sqsubseteq Q$, 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 level.[7] In the formalization and survey in this article, 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.[8] OWL is based on description logics, which are presented in Section 2.2. In description logics, concepts, roles, individuals, and axioms are

is a defect), fault localization (finding where the defect is), and fault repair (repairing). In this article the first step, detection, covers fault detection and fault localization. The second step, repair, which is the focus of the article, covers fault repair.

[7] Similar to the schema level in databases.

[8] E.g., https://www.w3.org/TR/2012/REC-owl2-primer-20121211/.

used, which relate to concepts, binary relations, instances, and axioms in ontology terminology, respectively.[9]

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 (e.g., Reference [99]).

2.2 Description Logics

In this article, we assume that ontologies are represented using a description logic TBox. Description logics [2] are knowledge representation languages. We briefly introduce notions in the field of description logics that are relevant to this article.

Syntax. 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. As an example, Table 1 shows the syntax and semantics of the $\mathcal{ALC}$ description logic's constructors and we refer to, e.g., Reference [4] for information on other description logics.

In current work on completing and debugging ontologies, different logics are used such as $\mathcal{EL}$ (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, and existential and universal restrictions). We mention later also $\mathcal{EL}$++, which is an extension of $\mathcal{EL}$ that uses additionally the bottom concept $\bot$ and nominals. Further, we mention $\mathcal{SHOIN}$, which in addition to the $\mathcal{ALC}$ constructors allows constructors for transitive roles, role hierarchies, nominals, inverse roles, and number restrictions.

Semantics. An interpretation $\mathcal{I}$ consists of a non-empty set $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that assigns to each atomic concept $P_a \in N_C$ a subset $P_a^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to each atomic role $r_a \in N_R$ a relation $r_a^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and to each individual name[10] $i \in N_I$ an element $i^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The interpretation function is straightforwardly extended to complex concepts (see Table 1). A TBox is a finite set of axioms. Axioms can be ***general concept inclusions* (GCIs)** (cf. Table 1) and *role inclusions* (RIs).[11] An interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ if for each GCI and RI 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 $\mathcal{T}$: $P^{\mathcal{I}} = \emptyset$. We say that a TBox is *incoherent* if it contains an *unsatisfiable* concept.

One of the main reasoning tasks for description logics is subsumption checking 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 of TBox $\mathcal{T}$. We say then also that Q *subsumes* P or that P *is-a* Q (or pronounced "every P *is-a* Q").

2.3 Completing and Debugging Workflow

In this article, we review approaches for improving the quality of ontologies by repairing them. In general, completing and debugging requires two steps. In the detection step, defects are found

[9] We note that terminology may differ when discussing ontologies, description logics, and OWL. Concepts in ontologies are called concepts in description logic terminology and classes in OWL terminology. Relations in ontologies are called roles (binary) in description logic terminology and properties in OWL terminology. Axioms in ontologies are also called axioms in description logic terminology and in OWL terminology. Instances in ontologies are called individuals in description logic terminology and in OWL terminology.

[10] As we do not deal with instances in this article, we do not use individuals in the later sections.

[11] Note that not all description logics allow role inclusions, e.g., $\mathcal{ALC}$ in Table 1.

Table 1. $\mathcal{ALC}$ Syntax and Semantics

Name	Syntax	Semantics
top	$\top$	$\Delta^{\mathcal{I}}$
bottom	$\bot$	$\varnothing$
negation	$\neg P$	$\Delta^{\mathcal{I}} \setminus P^{\mathcal{I}}$
conjunction	$P \sqcap Q$	$P^{\mathcal{I}} \cap Q^{\mathcal{I}}$
disjunction	$P \sqcup Q$	$P^{\mathcal{I}} \cup Q^{\mathcal{I}}$
existential restriction	$\exists r.P$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} (x,y) \in r^{\mathcal{I}} \wedge y \in P^{\mathcal{I}}\}$
universal restriction	$\forall r.P$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}} : (x,y) \in r^{\mathcal{I}} \rightarrow y \in P^{\mathcal{I}}\}$
GCI	$P \sqsubseteq Q$	$P^{\mathcal{I}} \subseteq Q^{\mathcal{I}}$

using different approaches. In the repairing step, the detected wrong information is removed and missing information is added. Although we focus on the repairing step and most work on repairing assumes 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 to each other, i.e., they detect different kinds of defects.

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. Further, as most ontology development methods require the use of competency questions to scope and delimit the ontology, these can be used to evaluate whether the ontology covers the intended domain.

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 Reference [90] 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 to identify these "root" concepts (e.g., as in Reference [50], where they do 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 [14]. 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 a k-nearest neighbors approach [65], association rules [66], bottom-up hierarchical clustering techniques [110], supervised classification [102], and formal concept analysis [15].

A much-employed approach is to use patterns. The pioneering pattern-based detection research was proposed by Reference [34]. The focus in that work 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. Lexico-syntactic patterns as well as logic patterns have been used to find wrong as well as missing information in ontologies (e.g., References [17, 52, 79, 86]) and ontology networks (e.g., References [11, 107]). The OOPS! system implements a variety of these [79].

There are also approaches that use knowledge that is intrinsic in an ontology network to detect defects. For instance, in References [37, 55] 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 Reference [10].

The detection of missing mappings is a research area on its own, i.e., ontology alignment [28], and we discuss this further in Section 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 actually *candidate* defects that 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. For high-quality results a domain expert needs to be involved in both steps (as in ontology alignment, e.g., References [78, 99]). 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 in the domain over other logically correct solutions that are correct in the domain (e.g., Reference [78]). 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 [37, 55] 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 is validated as missing, then we need to complete the ontology network. Otherwise, if this candidate missing is-a relation or mapping is validated as wrong, then it means that a wrong is-relation or mapping is derivable from the network and thus we need to debug the network. 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 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 repairs for other defects change, or new candidate defects are found. Furthermore, at any time during the process, the user can switch between different ontologies, restart earlier steps in the process (e.g., detecting again after repairing some defects), 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 [12, 33] where debugging approaches are reviewed. In this article, 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, Reference [33] introduces some criteria for debugging approaches. Regarding the criterion *application* the authors distinguish between different tasks such as repair, merging, and evolution. In this article, we focus on the repair task. The *granularity* of the repairs can be on the axiom level or on parts of axioms. We focus on the axiom level in this article, although we do mention recent approaches for axiom weakening that can deal with parts of axioms. Regarding *TBox and ABox support*, we focus on TBox support. Some repairing algorithms discussed in Section 5.1 focus on *inconsistency* 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 (Section 6). *User involvement* is discussed in Section 4.1.2 and Section 7.2. We mention the criteria *preservation of structure* (i.e., whether the original ontologies or normalized versions are repaired), *complexity* and *exploitation of background or context knowledge* in relevant places in this article.

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 the concept level, and thus do not deal with instances (or individuals in description logic terminology). Repairing essentially consists of two dual tasks that are both abductive in nature. Debugging deals with removing wrong information and completing with adding correct information. Both tasks need domain experts to obtain high-quality solutions. These dual tasks do influence each other and very little work has studied these in combination. In this article, we define the repairing problem as an abductive reasoning problem dealing with both removing and adding information. Further, as a repairing problem can have many solutions, we discuss preference relations between these. 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 formalizes the repair of an ontology for which missing and wrong information is given. An ontology is represented by a TBox $\mathcal{T}$. The identified missing and wrong information is represented by a set M of missing axioms and a set W of wrong[12] axioms.[13] To repair the TBox, a set A of axioms that are correct in the domain should be added to the TBox and a set D[14] of axioms that are not correct in the domain 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. In general, the set of all axioms that are correct in the domain and the set of all axioms that are not correct in 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 domain experts that can decide whether an axiom is correct or not in 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.

[12]We note that the wrong axioms can be explicitly asserted axioms or axioms derivable from the ontology.

[13]This means that both semantic defects and modelling defects can be dealt with. As an example of a semantic defect, for an incoherent ontology, the fact that concept C is unsatisfiable can be represented by axiom $C \sqsubseteq \bot$ in W. As an example of a modelling defect, for an axiom that is wrong in the domain (or missing), this axiom can be an axiom in W (or M).

[14]The axioms in D are axioms that are asserted in the ontology.

$\mathcal{T}$: {ax1: $P_1 \sqsubseteq P_2$, ax2: $P_1 \sqsubseteq P_3$, ax3: $P_1 \sqsubseteq \neg P_4$, ax4: $P_2 \sqsubseteq P_4$, ax5: $P_2 \sqsubseteq P_5$,
ax6: $P_3 \sqsubseteq P_5$, ax7: $P_3 \sqsubseteq P_6$, ax8: $P_4 \sqsubseteq P_7$, ax9: $P_5 \sqsubseteq \forall s.P_8$,
ax10: $P_6 \sqsubseteq \exists s.\neg P_8$}
Atomic concepts in $\mathcal{T}$: {P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8}
Or(X) = true for X = $P_1 \sqsubseteq P_3$ (ax2), $P_1 \sqsubseteq \neg P_4$ (ax3), $P_1 \sqsubseteq P_6$, $P_2 \sqsubseteq P_3$,
$P_2 \sqsubseteq P_4$ (ax4), $P_2 \sqsubseteq P_5$ (ax5), $P_2 \sqsubseteq P_6$, $P_2 \sqsubseteq P_7$, $P_2 \sqsubseteq \forall s.P_8$,
$P_3 \sqsubseteq P_6$ (ax7), $P_4 \sqsubseteq P_3$, $P_4 \sqsubseteq P_5$, $P_4 \sqsubseteq P_6$, $P_4 \sqsubseteq P_7$ (ax8),
$P_4 \sqsubseteq \forall s.P_8$, $P_5 \sqsubseteq \forall s.P_8$ (ax9), $P_7 \sqsubseteq P_3$, $P_7 \sqsubseteq P_6$,
and axioms derivable from this list (e.g., for concepts P, Q and O, if $P \sqsubseteq Q$, then also $P \sqcap O \sqsubseteq Q$.)
Or(X) = false if X is not in the list above (for true) or cannot be derived from the axioms in the list above.
M = {$P_4 \sqsubseteq P_5$}
W = {$P_1 \sqsubseteq \bot$, $P_3 \sqsubseteq \bot$}

R_1=({$P_4 \sqsubseteq P_5$, $P_7 \sqsubseteq P_3$}, {$P_1 \sqsubseteq P_2$ (ax1), $P_3 \sqsubseteq P_5$ (ax6), $P_6 \sqsubseteq \exists s.\neg P_8$ (ax10)})
R_2=({$P_4 \sqsubseteq P_5$, $P_7 \sqsubseteq P_3$}, {$P_1 \sqsubseteq P_2$ (ax1), $P_6 \sqsubseteq \exists s.\neg P_8$ (ax10)})
R_3=({$P_7 \sqsubseteq P_3$}, {$P_1 \sqsubseteq P_2$ (ax1), $P_6 \sqsubseteq \exists s.\neg P_8$ (ax10)})
R_4=({$P_4 \sqsubseteq P_5$}, {$P_1 \sqsubseteq P_2$ (ax1), $P_3 \sqsubseteq P_5$ (ax6)})
R_5=({$P_4 \sqsubseteq P_5$, $P_7 \sqsubseteq P_3$}, {$P_1 \sqsubseteq P_2$ (ax1), $P_3 \sqsubseteq P_5$ (ax6)})

Fig. 1. Example complete-debug-problem.

Definition 1 (Repair).[15] Let $\mathcal{T}$ be a TBox. 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 $\text{CDP}(\mathcal{T}, Or, M, W)$ is any pair of finite sets of TBox axioms (A, D) such that
(i) $\forall\ \psi_a \in A$: $Or(\psi_a)$ = true;
(ii) $\forall\ \psi_d \in D$: $Or(\psi_d)$ = false;
(iii) $(\mathcal{T} \cup A) \setminus D$ is consistent;
(iv) $\forall\ \psi_m \in M$: $(\mathcal{T} \cup A) \setminus D \models \psi_m$;
(v) $\forall\ \psi_w \in W$: $(\mathcal{T} \cup A) \setminus D \not\models \psi_w$.

As an example, consider the CDP in Figure 1 and visualized in Figure 2. Then, R_1, R_2, R_3, R_4, and R_5 (visualized in Figure 3) are all repairs of the CDP.

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 assumes 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

[15] In the field of model-based diagnosis, definitions for repairing ontologies are given that have similarities to our definition. For instance, in Reference [91], for an incoherent ontology $\mathcal{O}$, a *diagnosis* is defined as a minimal set of axioms $\mathcal{D} \subseteq \mathcal{O}$ such that $\mathcal{O} \setminus \mathcal{D}$ is coherent. A more elaborate definition is given in Reference [96]. Given an ontology $\mathcal{O}$, a background theory $\mathcal{B}$, which is a set of axioms that should not be changed, sets of axioms $\mathcal{P}$ and $\mathcal{N}$, which contain logical sentences that must be entailed and not entailed by the target ontology, respectively, a *diagnosis* $\mathcal{D} \subseteq \mathcal{O}$ is a set of axioms such that the set of axioms $\mathcal{O} \setminus \mathcal{D}$ can be extended by a logical description $\mathcal{E}X$ such that: (i) $(\mathcal{O} \setminus \mathcal{D}) \cup \mathcal{B} \cup \mathcal{E}X$ is consistent (and if required coherent), (ii) $\forall\ p \in \mathcal{P}$: $(\mathcal{O} \setminus \mathcal{D}) \cup \mathcal{B} \cup \mathcal{E}X \models p$, and (iii) $\forall\ n \in \mathcal{N}$: $(\mathcal{O} \setminus \mathcal{D}) \cup \mathcal{B} \cup \mathcal{E}X \not\models n$.

The sets $\mathcal{P}$ and $\mathcal{N}$ could be seen as similar to our sets M and W. The diagnosis $\mathcal{D}$ can be seen as our set D of axioms to remove. The set $\mathcal{E}X$ could be seen as a set of axioms to add. However, as the definition focuses on debugging and not completion, the definition only requires the existence of such a set and does not actually consider the addition of the axioms. The actual updating with $\mathcal{E}X$ is performed by the user after the diagnosis. Further, as most works on debugging, there is no mention of an oracle or domain expert in the definition. We note, however, that the implemented approach in Reference [96] does use a domain expert during the computation of the diagnosis. It does so by generating queries about entailments, asking a domain expert to answer these queries, and using these answers to guide the identification of the target diagnosis.

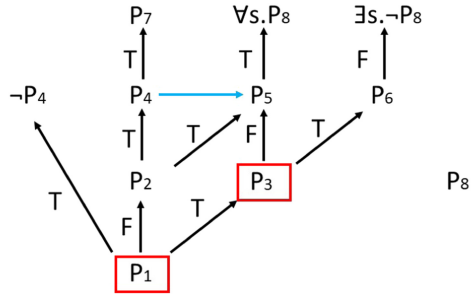

Fig. 2. Visualization of the example complete-debug-problem in Figure 1. The subsumption axioms in the TBox are represented with black arrows. The missing axiom is represented in blue. The unsatisfiable concepts are marked with a red box. The oracle's knowledge about the axioms in the ontology is marked with T (true) or F (false) at the arrows.

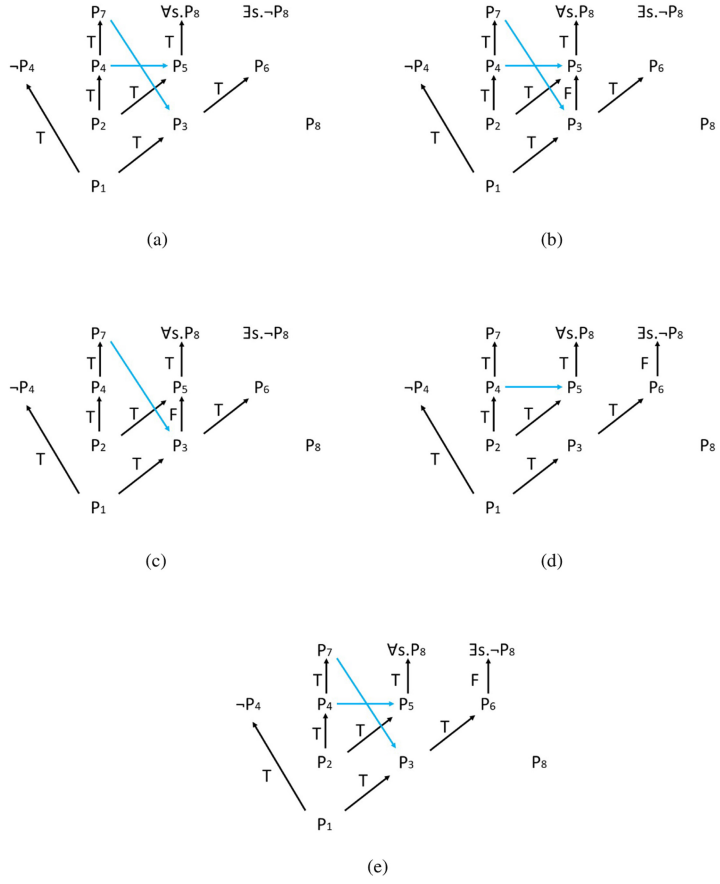

Fig. 3. Repairs (a) R1, (b) R2, (c) R3, (d) R4, (e) R5 for the example in Figure 1.

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 in 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 [20]. 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, then 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 all-knowing oracle, we know that all added axioms in A are correct in the domain and all removed axioms in D are false in the domain. Furthermore, for an all-knowing oracle, we know that $A \cap D = \varnothing$ (and then $(\mathcal{T} \cup A) \setminus D = (\mathcal{T} \setminus D) \cup A$). 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, a reasonable remedy is to define preference relations between repairs.

4.2.1 Basic Preferences. From the correctness perspective of a CDP it is important to find repairs that remove wrong information as much as possible, while from the completeness perspective as much correct information as possible should be added to the ontology. Therefore, we introduce the preference relations *less incorrect* and *more complete* between ontologies (Definitions 2 and 4) and repairs (Definitions 3 and 5) that formalize these intuitions.

Definition 2 states that one ontology is less incorrect than another if all wrong statements that can be derived from the first ontology also can be derived from the second ontology and there is a wrong statement that can be derived from the second ontology, but not from the first ontology. Therefore, if an ontology is less incorrect than another ontology, then the first ontology contains fewer wrong statements than the second ontology. Further, when the same wrong statements can be derived from two ontologies, then they are equally incorrect.

Definition 2 (Less Incorrect - Ontologies). Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be two ontologies represented by TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. Then, we say that $\mathcal{O}_1$ is *less incorrect* than $\mathcal{O}_2$ ($\mathcal{O}_2$ is *more incorrect* than $\mathcal{O}_1$) iff $(\forall \psi : (\mathcal{T}_1 \models \psi \wedge Or(\psi) = false) \rightarrow \mathcal{T}_2 \models \psi)) \wedge (\exists \psi : Or(\psi) = false \wedge \mathcal{T}_1 \not\models \psi \wedge \mathcal{T}_2 \models \psi)$. $\mathcal{O}_1$ and $\mathcal{O}_2$ are *equally incorrect* iff $\forall \psi : Or(\psi) = false \rightarrow (\mathcal{T}_1 \models \psi \leftrightarrow \mathcal{T}_2 \models \psi)$.

According to Definition 3, a repair R for an ontology is less incorrect than another repair R' for that ontology if the ontology repaired by R is less incorrect than the ontology repaired by R'.

Definition 3 (Less Incorrect - Repairs). Let $\mathcal{O}$ be an ontology represented by TBox $\mathcal{T}$ and let (A_1, D_1) and (A_2, D_2) be two repairs for CDP$(\mathcal{T}, Or, M, W)$. Let $\mathcal{O}_1$ be the ontology represented by $((\mathcal{T} \cup A_1) \setminus D_1)$ and $\mathcal{O}_2$ the ontology represented by $((\mathcal{T} \cup A_2) \setminus D_2)$. Then, (A_1, D_1) is *less incorrect*

than (A_2, D_2) (or (A_1, D_1) is preferred to (A_2, D_2) w.r.t. "less incorrect") iff $\mathcal{O}_1$ is *less incorrect* than $\mathcal{O}_2$.

Definition 4 states that an ontology is more complete than another ontology if all correct statements that can be derived from the second ontology also can be derived from the first ontology and there is a correct statement that can be derived from the first ontology, but not from the second ontology. Therefore, if an ontology is more complete than another ontology, then the first ontology contains more correct statements than the second ontology. Further, when the same correct statements can be derived from two ontologies, then they are equally complete.

Definition 4 (More Complete - Ontologies). Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be two ontologies represented by TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. Then, we say that $\mathcal{O}_1$ is *more complete* than $\mathcal{O}_2$ (or $\mathcal{O}_2$ is *less complete* than $\mathcal{O}_1$) iff $(\forall \psi : (\mathcal{T}_2 \models \psi \wedge Or(\psi) = true) \rightarrow \mathcal{T}_1 \models \psi)) \wedge (\exists \psi : Or(\psi) = true \wedge \mathcal{T}_1 \models \psi \wedge \mathcal{T}_2 \not\models \psi)$. $\mathcal{O}_1$ and $\mathcal{O}_2$ are *equally complete* iff $\forall \psi : Or(\psi) = true \rightarrow (\mathcal{T}_1 \models \psi \leftrightarrow \mathcal{T}_2 \models \psi)$.

According to Definition 5, a repair R for an ontology is more complete than another repair R' for that ontology if the ontology repaired by R is more complete than the ontology repaired by R'.

Definition 5 (More Complete - Repairs). Let $\mathcal{O}$ be an ontology represented by TBox $\mathcal{T}$ and let (A_1, D_1) and (A_2, D_2) be two repairs for CDP$(\mathcal{T}, Or, M, W)$. Let $\mathcal{O}_1$ be the ontology represented by $((\mathcal{T} \cup A_1) \setminus D_1)$ and $\mathcal{O}_2$ the ontology represented by $((\mathcal{T} \cup A_2) \setminus D_2)$. Then, repair (A_1, D_1) is *more complete* than repair (A_2, D_2) (or (A_1, D_1) is preferred to (A_2, D_2) w.r.t. "more complete") iff $\mathcal{O}_1$ is *more complete* than $\mathcal{O}_2$.

Definition 6 defines a preference relation for abduction problems related to removing redundancy using the subset relation. It compares the add and delete sets of two repairs and prefers to add and delete subset-wise fewer axioms.

Definition 6 (Subset). Let $R = (A, D)$ and $R' = (A', D')$ be two repairs for CDP$(\mathcal{T}, Or, M, W)$.
$R \subset_A R'$ iff $A \subset A'$.
$R \subset_D R'$ iff $D \subset D'$.
$R \subset R'$ iff $R \subset_A R' \wedge R \subset_D R'$.
(If $R \subset R'$, then we also say that R is preferred to R' with respect to $\subset$.)

As examples, for the CDP in Figure 1, 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_4 and R_5 are equally incorrect. R_1, R_2, R_3, and R_5 are equally complete and they are more complete than R_4. Further, $R_3 \subset R_2 \subset R_1$ and $R_4 \subset R_5 \subset R_1$.

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 (Definitions 7–9).

Definition 7 (Minimally Incorrect). A repair $R = (A, D)$ for CDP$(\mathcal{T}, Or, M, W)$ is said to be minimally incorrect (or preferred with respect to "less incorrect") iff there is no repair R' that is less incorrect than R.

Definition 8 (Maximally Complete). A repair $R = (A, D)$ for CDP$(\mathcal{T}, Or, M, W)$ is said to be maximally complete (or preferred with respect to "more complete") iff there is no repair R' that is more complete than R.

Definition 9 (Subset Minimal). A repair $R = (A, D)$ for CDP$(\mathcal{T}, Or, M, W)$ is said to be subset minimal (or preferred with respect to $\subset$) iff there is no repair R' such that $R' \subset_A R$ and $R' \subset_D R$.

As examples, for the CDP in Figure 1, R_1 is minimally incorrect, R_1, R_2, R_3, and R_5 are maximally complete, and R_3 and R_4 are subset minimal.

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 (Definition 10). 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 10 (Dominate). Let $R = (A, D)$ and $R' = (A', D')$ be two repairs for CDP$(\mathcal{T}, Or, M, W)$. *R dominates R'* with respect to a set of preference relations $\mathcal{P} \subseteq$ {more complete, less incorrect, $\subset$} if R is more or equally preferred to R' for all preference relations in $\mathcal{P}$, and R is more preferred to R' for at least one of the preference relations in $\mathcal{P}$.

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. Definition 11 formalizes this.

Definition 11 (Combining With Priority to One of The Preference Relations). Let $X \in$ {less incorrect, more complete, $\subset$}. Let $\mathcal{P} \subseteq$ {less incorrect, more complete, $\subset$} $\setminus$ {X}. A repair R for CDP$(\mathcal{T}, Or, M, W)$ is said to be *X-optimal* with respect to $\mathcal{P}$ iff 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 $\mathcal{P}$.

Essentially, we are looking for repairs that are preferred with respect to X. However, there could be several such. Among these, we use $\mathcal{P}$ to filter and retain the ones that are not dominated with respect to $\mathcal{P}$. For instance, if X is "more complete" and $\mathcal{P}$ = {less incorrect}, then we would retain the repairs that are the most complete and among these the ones that are least incorrect.

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. Definition 12 formalizes this.[16]

Definition 12 (Combining With Equal Priority). Let $\mathcal{P} \subseteq$ {less incorrect, more complete, $\subset$}. A repair R for CDP$(\mathcal{T}, Or, M, W)$ is said to be *skyline-optimal* with respect to $\mathcal{P}$ iff there is no other repair that dominates R with respect to $\mathcal{P}$.

As an example, for $\mathcal{P}$ = {less incorrect, more complete}, we retain the repairs for which there are no other repairs that are both less incorrect and more complete.

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 Figure 1, R_4 is $\subset$-optimal with respect to {less incorrect} and R_3 is $\subset$-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. The advantage of minimally incorrect and

[16]The notion of a skyline was introduced in the database field for retrieving multidimensional data points that are not worse than any other point in all dimensions [13]. An example given in Reference [13] refers to going on holiday and looking for a hotel that is cheap and close to the beach. As hotels close to the beach usually are more expensive, it is unlikely to find a hotel that is closest and cheapest. Interesting alternatives could be hotels that are not more expensive as well as farther away from the beach than other hotels. These hotels make up the "skyline."

less-incorrect-optimal repairs is that a maximal body of wrong information is removed from the ontology. 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, by taking into account subsumption relations between the concepts in the axioms,[17] the number of requests will still be large.) Repairs prioritizing subset minimality ensure that there is no redundancy. The advantage of removing redundant axioms from a repair is the reduction of computation time as well as the reduction of unnecessary user interaction. However, in some cases redundancy may be desired. 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. Redundancy can also be useful in a collaborative environment when multiple parties edit the ontology. Furthermore, prioritizing redundancy may lead to less complete or more incorrect repairs. Skyline-optimal is a relaxed criterion. When, for instance, $\mathcal{P}$ = {less incorrect, more complete}, 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., see Section 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 Definition 1, $M = \varnothing$ and $A = \varnothing$. 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 Definition 1 as requirement (ii) may not be satisfied. We also note that, as axioms are removed from the ontology and some wrong axioms are not derivable anymore, the resulting ontology will be less incorrect than the original ontology. However, correct axioms may be removed and thus the resulting ontology may be less complete than the original ontology.

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 [50]. 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. Therefore, such reasoners that can be highly optimized can be used unchanged, but they may need to be called an exponential number of times [7]. A glass-box approach, however, is based on the internals of the reasoning algorithm of a description logic reasoner and tries to find solutions in a single run. Many glass-box approaches extended the reasoner for specific description

[17] As an example, if $P \sqsubseteq Q$ is correct, then also $P_1 \sqsubseteq Q_1$ is correct for P_1 a sub-concept of P and Q_1 a superconcept of Q.

logics. To overcome having to design new extensions for every description logic reasoner, general approaches for extending reasoners have been proposed in References [7, 8].

5.1.1 A General Approach. A general approach for repairing incoherent ontologies is the following (adapted from Reference [89]). (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.[18] These minimal reasons for the unsatisfiability of a concept are sets of axioms and such a set of axioms is called a **minimal unsatisfiability-preserving sub-TBox (MUPS)** or justifications for the unsatisfiability. We need to compute these MUPS or justifications for all unsatisfiable concepts. From these, we can compute the smallest sets of axioms in the original TBox that cause that TBox to be incoherent. Such a set is called a **minimal incoherence-preserving sub-TBox (MIPS)**. 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 Definition 13. A MUPS in a consistent TBox can be seen as a justification (Definition 14) for an unsatisfiable concept. Indeed, if we instantiate ψ in Definition 14 with $P \sqsubseteq \bot$, then we obtain the MUPS for P. Justifications are sometimes called **MinAs (Minimal Axiom sets)**. The task of finding justifications is also called axiom pinpointing. Complexity results regarding axiom pinpointing in lightweight description logics are given in References [75–77].[19] The definition of MIPS is given in Definition 15.

Definition 13 ((MUPS) [89]). Let $\mathcal{T}$ be a TBox and P be an unsatisfiable concept in $\mathcal{T}$. A set of axioms $\mathcal{T}' \subseteq \mathcal{T}$ is a **minimal unsatisfiability-preserving sub-TBox (MUPS)** if P is unsatisfiable in $\mathcal{T}'$ and P is satisfiable in every sub-TBox $\mathcal{T}'' \subsetneq \mathcal{T}'$.

Definition 14 ((Justification) (Similar to Reference [48])). Let $\mathcal{T}$ be a consistent TBox and $\mathcal{T} \models \psi$. A set of axioms $\mathcal{T}' \subseteq \mathcal{T}$ is a justification for ψ in $\mathcal{T}$ if $\mathcal{T}' \models \psi$ and $\forall \mathcal{T}'' \subsetneq \mathcal{T}' : \mathcal{T}'' \not\models \psi$.

Definition 15 ((MIPS) [89]). Let $\mathcal{T}$ be an incoherent TBox. A TBox $\mathcal{T}' \subseteq \mathcal{T}$ is a minimal incoherence-preserving sub-TBox (MIPS) if $\mathcal{T}'$ is incoherent and every sub-TBox $\mathcal{T}'' \subsetneq \mathcal{T}'$ 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 (Definition 16) of the set of MIPSs and remove the hitting set from the TBox.

Definition 16 ((Hitting Set) [81]). Let $\mathcal{T}$ be a collection of sets. A hitting set for $\mathcal{T}$ is a set $H \subseteq \cup_{S \in \mathcal{T}} S$ such that $\forall S \in \mathcal{T} : H \cap S \neq \emptyset$.

In Reference [89] these hitting sets are called pinpoints, but note that they may not be minimal hitting sets.[20]

[18] In model-based diagnosis approaches, using the definition of diagnosis according to Reference [91], a diagnosis can be computed on the basis of conflicts sets, where a conflict set is a set of axioms that is incoherent. A minimal diagnosis is a diagnosis for which no proper subset is a diagnosis. It was shown in Reference [81] that the set of diagnoses corresponds to the collection of minimal hitting sets for the minimal conflict sets. The notion of MUPS defined in this section refers to minimal conflict set for one particular unsatisfiable concept.

[19] For some restricted description logics, justifications are enumerable with polynomial delay. We note that these papers do not only address the question of generating all justifications, but also related questions, such as generating justifications in a specific order or checking whether a set of axioms is a justification for an axiom.

[20] Assume we have a set {{ax10, ax11}, {ax12, ax13}, {ax14, ax10}, {ax14, ax12}} of MIPSs, then {ax10, ax12, ax14} is a possible pinpoint, even though {ax10, ax12} is a minimal hitting set and pinpoint [89].

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

In general, there may be several hitting sets for the set of MIPS. Different approaches use different heuristics for ranking the possible repairs.

5.1.2 Approaches and Systems. The first tableau-based algorithm for debugging of an ontology was proposed in References [89, 90]. (For an overview of how a tableau-based reasoner works, see, e.g., Reference [6].) A glass-box approach was used for an $\mathcal{ALC}$ reasoner. The branches in the tableau-based reasoner were used to compute MUPS. The MIPS were computed by taking a subset-reduction of the union of all MUPSs. The subset-reduction of a set S of TBoxes is the smallest subset of S such that, for all TBoxes $\mathcal{T}$ in S, there is a TBox $\mathcal{T}'$ in the subset-reduction that is a subset of $\mathcal{T}$ [90]. Computing MUPS and MIPS for an unfoldable $\mathcal{ALC}$ TBox was shown to be in PSPACE, where an unfoldable TBox is a TBox where the left-hand sides of the GCIs are atomic concepts and the right-hand sides contain no reference (direct or indirect) to the defined atomic concept.

Computing a pinpoint from MIPS takes linear time (non-minimal case), while the problem of computing a minimal hitting set from MIPS is NP-complete [89]. This approach was implemented for unfoldable $\mathcal{ALC}$ TBoxes in the system MUPSter [91]. The tableau algorithm in Reference [70] can be seen as an extension of this work. It computes maximally satisfiable sub-TBoxes and does not require individual steps for computing MUPS and applying the hitting set algorithm. Also, the approach in Reference [70] finds maximally coherent sub-TBoxes and presents an EXP-TIME algorithm for unfoldable $\mathcal{ALC}$ TBoxes. The DION system [91] uses a bottom-up algorithm to compute MUPS. For an unsatisfiable concept P it computes two sets of axioms Σ and S such that P is satisfiable in S, but not in $\Sigma \cup S$. Then, subsets S' of Σ are computed such that P is unsatisfiable in $S \cup S'$. By checking for minimality, we obtain MUPS. For efficiency reasons not all sets of axioms are checked, but the search is guided by a relevance function, e.g., by using only axioms that are in some way relevant to the unsatisfiable concept. In Reference [50] a glass-box technique is used to compute MUPS (called set of support in Reference [50]) for OWL ontologies ($\mathcal{SHOIN}$). In Reference [48] a method was proposed to calculate all justifications of an unsatisfiable concept. Both a glass-box and black-box technique are presented for computing a single justification. The glass-box technique is an extension from Reference [50], while the black-box technique is based on an expansion stage where axioms are added to an initially empty set until a concept becomes unsatisfiable and a shrinking step where extraneous axioms are removed. Then, given an initial justification, a black-box method computes all justifications using a variation of the hitting set tree algorithm [81]. We note that, based on experiments in Reference [36], computing all justifications in inconsistent ontologies may be more difficult than computing all justifications in consistent ontologies (which are the ones referred to in Definition 14), a possible reason being that defects can be dealt with one at a time in consistent ontologies, but for inconsistent ontologies the only information that we have is that the ontology is inconsistent [36]. The BEACON system [1] implements an algorithm for $\mathcal{EL}^+$ ontologies based on a translation of the normalized TBox (i.e., the TBox is first rewritten into a specific format) into Horn clauses. Axioms that need a justification have then a Horn clause counterpart. For these Horn clauses, minimal correction subsets are computed, which in their turn refer to repairs in the normalized TBox. The work is based on techniques proposed in Reference [94] and uses SAT-solvers. Another approach using a translation to Horn clauses, but using a resolution-based approach, is implemented in the PULi system [51].

Experiments compared PULi with SAT-based algorithms on $\mathcal{EL}$ ontologies and showed comparable or better performance.

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., Reference [41]). In Reference [89] a heuristic is used stating that axioms appearing in more MIPSs are likely to be more erroneous. Therefore, axioms appearing in the most MIPSs are removed first (or, in other words, are first added to the hitting set). In Reference [49], an arity-based heuristic is used that is similar to the heuristic in Reference [89]. Further, Reference [49] introduces heuristics based on the impact on the ontology, in terms of entailment of subsumption and disjointness for concepts and instantiation of concepts, when an axiom is removed. Another approach is 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 Reference [50] 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 derived concepts. Repairing root concepts may automatically repair derived concepts. The idea of root concepts is used in Reference [71] 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 root concepts is also used in the ORE system [61]. It implements an algorithm to find dependencies between the reasons for unsatisfiability for different concepts. One concept may be unsatisfiable because of the unsatisfiability of another concept. The proposed algorithm is sound (all found dependencies are correct), but incomplete (not all dependencies are found). Also, Reference [111] uses root concepts within so-called clash modules that are incoherent sub-TBoxes built around negated concepts. In Reference [43] a notion of relevance between axioms is defined and used to guide the computation of justifications. In Reference [40] a number of patterns, which can explain unsatisfiability of concepts in normalized TBoxes, are proposed. These patterns are used instead of invoking description logic reasoners frequently. This leads to a fast algorithm for finding MUPS is sound, but the algorithm is not complete.

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., Reference [96]). As shown in Reference [85], 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 [88], which implements a number of black-box algorithms [31, 35, 96, 97] for computing repairs and guides the user using queries to find a final repair.

In Reference [30] the focus is on automatically learned ontologies. According to Reference [30] many of these ontologies share the facts that logically redundant axioms are generated and that the generated axioms only need restricted expressiveness of the representation language. For such

a restricted language (a lightweight description logic), an approach based on the idea of truth maintenance systems is proposed. In this approach, 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.

An approach that has not been proposed earlier, but that follows naturally from the definitions and preferences defined in Section 4, is to use the oracle for the axioms in the MIPSs. For every MIPS, remove the axioms ψ such that $Or(\psi)$ = true. If at least one of the MIPS becomes the empty set, then there is no repair unless we are willing to remove correct information. Assuming we have non-empty MIPSs after the removal of correct axioms, a hitting set of the MIPSs would result in a repair. When redundancy in the repair 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 MIPSs (as for these axioms ψ, we have that $Or(\psi)$ = false). This repair is less or equally incorrect than the repairs obtained using hitting sets.

There are also approaches that map the debugging problem into a revision problem (e.g., References [32, 72, 82]). A revision state [72] is a tuple of ontologies ($\mathcal{O}$, $\mathcal{O}^{\models}$, $\mathcal{O}^{\not\models}$) where $\mathcal{O}^{\models} \subseteq \mathcal{O}$, $\mathcal{O}^{\not\models} \subseteq \mathcal{O}$, and $\mathcal{O}^{\models} \cap \mathcal{O}^{\not\models} = \emptyset$. $\mathcal{O}^{\models}$ represents the wanted consequences of $\mathcal{O}$, while $\mathcal{O}^{\not\models}$ represents the unwanted consequences. In a complete revision state, we also have that $\mathcal{O} = \mathcal{O}^{\models} \cup \mathcal{O}^{\not\models}$. For a CDP, $\mathcal{O}^{\not\models}$ could be initialized with W (and when dealing with completion, $\mathcal{O}^{\models}$ could be initialized with M). The approach in Reference [72] 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 Reference [32] a MIPS approach is used in the definition of the revision operator. In Reference [82] 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 Definition 1, $\forall\ \psi_m \in M$: $Or(\psi_p)$ = 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 $\mathcal{T} \cup M$ 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 in the detection phase, this is essentially all that can be done (except for removing redundancy, if so desired). If not all missing is-a relations were detected—and this is the common case—there are different ways to repair the ontology that are not all equally interesting and we can use the earlier defined preference relations.

As these approaches do not deal with correctness, Definition 3 is not used, and $\subset_D$ should be removed in Definition 6. In Definitions 10 and 12, $\mathcal{P}$ = {more complete, $\subset$}. In Definition 11, "less incorrect" should be removed. In this case, the semantically maximal solutions in Reference [108] 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 [57, 58, 60], for $\mathcal{EL}$ TBoxes [23, 60, 108], and for $\mathcal{ALC}$ TBoxes [54]. All algorithms compute logically correct solutions that then need to be validated for correctness in the domain by a domain expert. It is assumed that the axioms in M and A represent subsumption between atomic concepts in the

ontologies. The algorithms for taxonomies and (normalized) $\mathcal{EL}$ TBoxes 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 $P_1 \sqsubseteq P_2$ by adding axioms of the form $P_1' \sqsubseteq P_2'$ where $P_1 \sqsubseteq P_1'$ and $P_2' \sqsubseteq P_2$. For $\mathcal{EL}$, additionally, is-a relations of the form $\exists r.P_1 \sqsubseteq \exists r.P_2$ are repaired by repairing $P_1 \sqsubseteq P_2$. 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 optional 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 References [60, 108]. In Reference [54] an approach is proposed for $\mathcal{ALC}$ TBoxes by modifying a tableau-based reasoner. Repairs 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 $x : P$ and $x : \neg N$ (representing that individual x belongs to concept P as well as to concept $\neg N$), and asserting then that $P \sqsubseteq N$. Additionally, the same technique as for taxonomies is applied.

If we consider the case where, in test-driven debugging, the positive test cases would be added to the ontology (see footnote 15), then these approaches (discussed in Section 5.1) also deal with completion.

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 $\mathcal{SHOIN}(D)$) is proposed in Reference [25]. 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 References [37, 55] 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 Section 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.

As mentioned in the previous section, if the positive test cases would be added to the ontology in test-driven debugging (discussed in Section 5.1), then these approaches deal with debugging and completion.

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. Also for ontology networks, using completely automatic systems may reduce the quality of the final ontology in terms of correctness and completeness [78].

Our definitions in Sections 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 [28] 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., References [21, 28, 47, 59, 73, 98, 99] and at the ontology matching web site.[21] 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. 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 weighted-sum strategy assigns a weight to each matcher and computes the similarity between two entities as a weighted sum over the similarities computed by the matchers. The maximum-based strategy retains the highest similarity value computed by the matchers. The most common filtering strategy is the threshold filtering where pairs of entities with a similarity value higher than or equal to a given threshold are retained as candidate mappings. 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 Reference [21]). 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 (e.g., for weights for matchers or thresholds for filtering) in the system. A system that integrates all of these is discussed in Reference [56].

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 Reference [68], and the RaDON [42], ALCOMO [67], LogMap [44, 45], and AgreementMakerLight [87] systems. Other heuristics than the ones in Section 5 could be used. For instance, the conservativity principle [101] states that the integrated ontology should not induce any change in the concept hierarchies of the

[21] http://www.ontologymatching.org.

input ontologies. In References [46, 67, 87] the confidence values of the mappings are taken into account, and in Reference [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., References [26, 69, 80]. 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 Section 5, is discussed in References [37, 55].

7 OPPORTUNITIES

In this article, 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 optimizations 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. Thus, involving a domain expert in the generation and validation of repairs, as done by some interactive systems such as RepOSE [55, 63] and OntoDebug [88, 96], seems to be necessary. There is relatively little work on dealing with completeness. There is still a need for new approaches and more interactive systems. Even less work deals with completeness and correctness. We need work on algorithms guaranteeing different kinds of preferred repairs. For instance, RepOSE, a 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 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 that represents information about the relevance or importance of parts of the ontology [33]. In Reference [96], 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 Reference [33].

In this survey, a repair consists of a set of axioms that are added and a set of axioms that are deleted from the ontology (Definition 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 References [24, 35]. A method that rewrites the axioms in the ontology into simpler axioms and debugs these is shown in Reference [49]. In Reference [53] 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 References [5, 16, 105]. However, as recently shown in Reference [63], this issue has not been studied fully yet. For instance, when several defects need to be repaired, the order of dealing with the defects may influence the repair. The work in Reference [63] also provides the first approach integrating removing (but not full debugging), completion and weakening. In Reference [83] a concept contraction method for $\mathcal{EL}$ is proposed. In this case, for concepts C and D such that D subsumes C, a new concept is obtained that is as similar as possible to C but that is not subsumed by D. Such a contracted concept can be used for weakening by changing a concept in part of the axiom to the contracted concept. 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 Definition 1, 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 (e.g., References [3, 19, 27, 61, 74, 92, 93, 95, 112]). 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.

An earlier study with users on ontology authoring concluded that debugging is difficult [106]. 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. Although newer interactive systems, such as different versions of RepOSE [23, 55, 63] and OntoDebug [88], have graphical user interfaces, support debugging and completion, and allow for configuration with respect to used algorithms and heuristics, there are still issues that can be addressed. As ontology alignment can be seen as a kind of completion (Section 6), we can consult the guidelines for ontology alignment systems that may thus also be valid for the more general completion and debugging systems. Recommendations for user support for ontology alignment systems regarding user interfaces (partly based on Reference [29]) as well as infrastructure and algorithms are given in Reference [38]. 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 Reference [62] 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 Reference [39] 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 Section 4.1.2). This has been reported often, and for instance, the study in Reference [84] 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 Reference [9], 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 Reference [84], 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 Reference [62], 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 Reference [33].

Further, in Reference [9] 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. As mentioned earlier, one of the few that has different steps regarding completion and debugging within the general framework is an extension of the eXtreme Design Methodology [22].

8 CONCLUSION

As semantically enabled applications require high-quality ontologies, developing and maintaining as correct and complete as possible ontologies is an important, yet 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 article, 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 we point to new opportunities for further research to advance the field.

APPENDIX

In Figure A.1 (and visually in Figure A.2), we show an example of a complete-debug-problem inspired by the Galen ontology[22] in the $\mathcal{EL}$ language. For the ontology represented by the TBox, we assume that we have detected two missing axioms Endocarditis ⊑ PathologicalPhenomenon and GranulomaProcess ⊑ NonNormalProcess. Further, we have detected that we can infer the wrong axiom PathologicalProcess ⊑ GranulomaProcess from the ontology.

$\mathcal{T}$ = { CardioVascularDisease ⊑ PathologicalPhenomenon,
Fracture ⊑ PathologicalPhenomenon,
∃hasAssociatedProcess.PathologicalProcess ⊑ PathologicalPhenomenon,
Endocarditis ⊑ Carditis,
Endocarditis ⊑ ∃hasAssociatedProcess.InflammationProcess,
PathologicalProcess ⊑ NonNormalProcess,
PathologicalProcess ⊑ InflammationProcess,
InflammationProcess ⊑ GranulomaProcess }

Atomic concepts in $\mathcal{T}$ = { GranulomaProcess, CardioVascularDisease, PathologicalPhenomenon,
Fracture, Endocarditis, Carditis, InflammationProcess, PathologicalProcess, NonNormalProcess}
Atomic roles in $\mathcal{T}$ = { hasAssociatedProcess }

M = { Endocarditis ⊑ PathologicalPhenomenon,
GranulomaProcess ⊑ NonNormalProcess }

W = { PathologicalProcess ⊑ GranulomaProcess }

The following axioms are correct in the domain, i.e.,
Or returns $true$ for:
GranulomaProcess ⊑ InflammationProcess,
GranulomaProcess ⊑ PathologicalProcess,
GranulomaProcess ⊑ NonNormalProcess,
InflammationProcess ⊑ PathologicalProcess,
InflammationProcess ⊑ NonNormalProcess,
PathologicalProcess ⊑ NonNormalProcess,
CardioVascularDisease ⊑ PathologicalPhenomenon,
Fracture ⊑ PathologicalPhenomenon,
Endocarditis ⊑ PathologicalPhenomenon,
Endocarditis ⊑ Carditis,
Endocarditis ⊑ CardioVascularDisease,
Carditis ⊑ PathologicalPhenomenon,
Carditis ⊑ CardioVascularDisease,
∃hasAssociatedProcess.PathologicalProcess ⊑ PathologicalPhenomenon,
∃hasAssociatedProcess.InflammationProcess ⊑ PathologicalPhenomenon,
Endocarditis ⊑ ∃hasAssociatedProcess.InflammationProcess,
Endocarditis ⊑ ∃hasAssociatedProcess.PathologicalPhenomenon.
Note that for an oracle that does not make mistakes,
if Or(P ⊑ Q) = true, then also Or(∃r.P ⊑ ∃r.Q)=true.
and if Or(P ⊑ Q)=true, then also Or(P ⊓ O ⊑ Q)=true.
For other axioms P ⊑ Q with P, Q ∈ C, Or(P ⊑ Q) = false.

Fig. A.1. $\mathcal{EL}$ example. ($\mathcal{T}$ is a TBox representing the ontology. M is a set of missing axioms. W is the set of wrong axioms. Or is the oracle representing the domain expert.)

[22] http://www.openclinical.org/prj_galen.html.

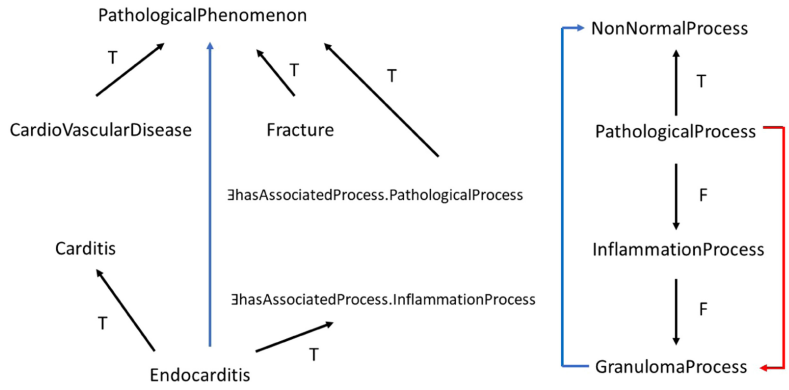

Fig. A.2. Visualization of the example complete-debug-problem in Figure A.1. The axioms in the TBox are represented with black arrows. The detected missing axioms are represented in blue. The detected wrong axiom is represented in red. The oracle's knowledge about the axioms in the ontology is marked with T (true) or F (false) at the arrows.

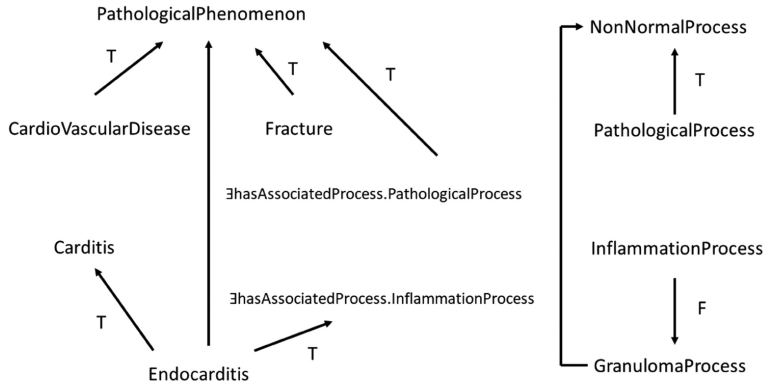

Fig. A.3. Result after repairing the ontology in the complete-debug-problem in Figure A.1 with repair R1. R1 = (A_1,D_1) with A_1 = { Endocarditis ⊑ PathologicalPhenomenon, GranulomaProcess ⊑ NonNormalProcess }, D_1 = { PathologicalProcess ⊑ InflammationProcess }.

Repairs

There are different possible repairs of which we show some. Figure A.3 shows the ontology after repair R1 is executed, where R1 = (A_1,D_1) with A_1 = { Endocarditis ⊑ PathologicalPhenomenon, GranulomaProcess ⊑ NonNormalProcess } and D_1 = { PathologicalProcess ⊑ 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*(Endocarditis ⊑ PathologicalPhenomenon) = true, and *Or*(GranulomaProcess ⊑ NonNormalProcess) = true. (ii) For the statement in D_1, we know that *Or*(PathologicalProcess ⊑ InflammationProcess) = false. (iii) $(\mathcal{T} \cup A_1) \setminus D_1$ is consistent as $\mathcal{EL}$ TBoxes are consistent. (iv) $(\mathcal{T} \cup A_1) \setminus D_1 \models$ Endocarditis ⊑ PathologicalPhenomenon, as Endocarditis ⊑ PathologicalPhenomenon has been explicitly added. $(\mathcal{T} \cup A_1) \setminus D_1 \models$ GranulomaProcess ⊑ NonNormalProcess, as GranulomaProcess ⊑ NonNormalProcess has been explicitly added. (v) $(\mathcal{T} \cup A_1) \setminus D_1 \not\models$ PathologicalProcess ⊑ GranulomaProcess. The only justification of PathologicalProcess ⊑ GranulomaProcess in the original ontology was

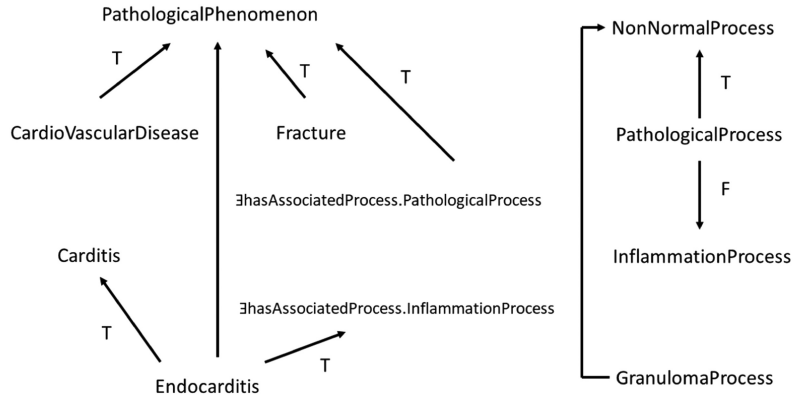

Fig. A.4. Result after repairing the ontology in the complete-debug-problem in Figure A.1 with repair R2. R2 = (A_2,D_2) with A_2 = { Endocarditis ⊑ PathologicalPhenomenon, GranulomaProcess ⊑ NonNormalProcess }, D_2 = { InflammationProcess ⊑ GranulomaProcess }.

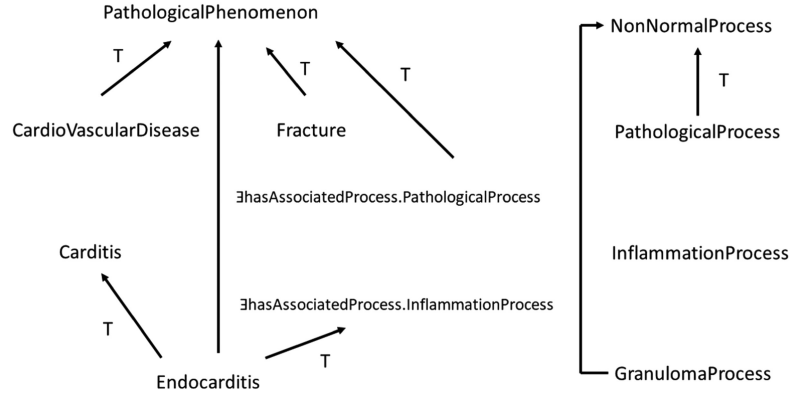

Fig. A.5. Result after repairing the ontology in the complete-debug-problem in Figure A.1 with repair R3. R3 = (A_3,D_3) with A_3 = { Endocarditis ⊑ PathologicalPhenomenon, GranulomaProcess ⊑ NonNormalProcess }, D_3 = { PathologicalProcess ⊑ InflammationProcess, InflammationProcess ⊑ GranulomaProcess }.

{ PathologicalProcess ⊑ InflammationProcess, InflammationProcess ⊑ GranulomaProcess } and this is not available anymore after the repair as PathologicalProcess ⊑ InflammationProcess is removed. The newly added axioms do not give rise to new justifications for PathologicalProcess ⊑ GranulomaProcess.

Figure A.4 shows the ontology after repair R2 is executed, where R2 = (A_2,D_2) with A_2 = A_1 = { Endocarditis ⊑ PathologicalPhenomenon, GranulomaProcess ⊑ NonNormalProcess } and D_2 = { InflammationProcess ⊑ GranulomaProcess }. We show that R2 is a repair. (i) Same as R1. (ii) For the statement in D_2, we know that *Or*(InflammationProcess ⊑ GranulomaProcess) = false. (iii) Same as R1. (iv) Same as R1. (v) $(\mathcal{T} \cup A_2) \setminus D_2 \not\models$ 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 InflammationProcess ⊑ GranulomaProcess is removed. The newly added axioms do not give rise to new justifications for PathologicalProcess ⊑ GranulomaProcess.

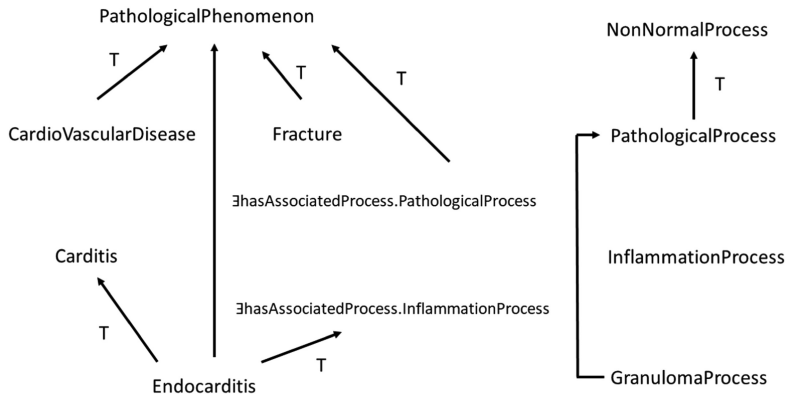

Fig. A.6. Result after repairing the ontology in the complete-debug-problem in Figure A.1 with repair R4. R4 = (A_4,D_4) with A_4 = { Endocarditis ⊑ PathologicalPhenomenon, GranulomaProcess ⊑ PathologicalProcess }, D_4 = { PathologicalProcess ⊑ InflammationProcess, InflammationProcess ⊑ GranulomaProcess }.

Figure A.5 shows the ontology after repair R3 is executed, where R3 = (A_3,D_3) with A_3 = A_1 = A_2 = { Endocarditis ⊑ PathologicalPhenomenon, GranulomaProcess ⊑ NonNormalProcess } and D_3 = D_1 ∪ D_2 = { PathologicalProcess ⊑ InflammationProcess, InflammationProcess ⊑ GranulomaProcess }. We show that R3 is a repair. (i) Same as R1. (ii) For the statements in D_3, 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) $(\mathcal{T} \cup A_3) \setminus D_3 \not\models$ 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.

Figure A.6 shows the ontology after repair R4 is executed, where R4 = (A_4,D_4) with A_4 = { Endocarditis ⊑ PathologicalPhenomenon, GranulomaProcess ⊑ PathologicalProcess } and D_4 = D_3 = { PathologicalProcess ⊑ InflammationProcess, InflammationProcess ⊑ GranulomaProcess }. We show that R4 is a repair. (i) Regarding the statements in A_4, we know that *Or*(Endocarditis ⊑ PathologicalPhenomenon) = true, and *Or*(GranulomaProcess ⊑ PathologicalProcess) = true. (ii) Same as R3. (iii) Same as R1. (iv) $(\mathcal{T} \cup A_4) \setminus D_4 \models$ Endocarditis ⊑ PathologicalPhenomenon, as Endocarditis ⊑ PathologicalPhenomenon has been explicitly added. $(\mathcal{T} \cup A_4) \setminus D_4 \models$ 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.

Figure A.7 shows the ontology after repair R5 is executed, where R5 = (A_5,D_5) with A_5 = { Carditis ⊑ PathologicalPhenomenon, GranulomaProcess ⊑ PathologicalProcess } and D_5 = D_3 = D_4 = { PathologicalProcess ⊑ InflammationProcess, InflammationProcess ⊑ GranulomaProcess }. We show that R5 is a repair. (i) Regarding the statements in A_5, we know that *Or*(Carditis ⊑ PathologicalPhenomenon) = true, and *Or*(GranulomaProcess ⊑ PathologicalProcess) = true. (ii) Same as R3. (iii) Same as R1. (iv) $(\mathcal{T} \cup A_5) \setminus D_5 \models$ Endocarditis ⊑ PathologicalPhenomenon, as this can be derived from the axiom Endocarditis ⊑ Carditis that was already in the TBox and the added Carditis ⊑ PathologicalPhenomenon. $(\mathcal{T} \cup A_3) \setminus D_3 \models$ GranulomaProcess ⊑ NonNormalProcess, for the same reason as in R4. (v) Same as R3.

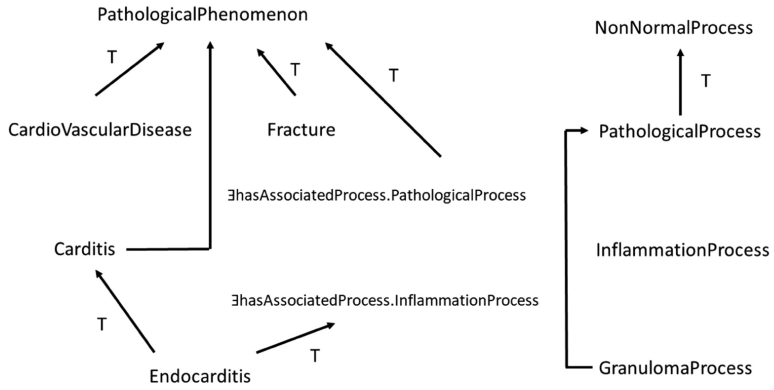

Fig. A.7. Result after repairing the ontology in the complete-debug-problem in Figure A.1 with repair R5. R5 = (A_5,D_5) with A_5 = { Carditis ⊑ PathologicalPhenomenon, GranulomaProcess ⊑ PathologicalProcess }, D_5 = { PathologicalProcess ⊑ InflammationProcess, InflammationProcess ⊑ GranulomaProcess }.

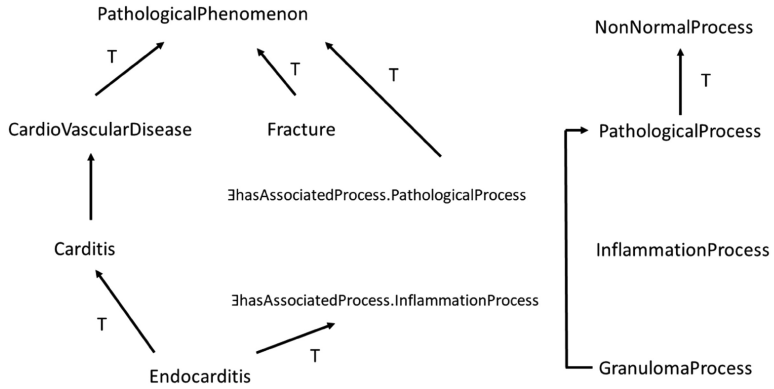

Fig. A.8. Result after repairing the ontology in the complete-debug-problem in Figure A.1 with repair R6. R6 = (A_6,D_6) with A_6 = { Carditis ⊑ CardioVascularDisease, GranulomaProcess ⊑ PathologicalProcess }, D_6 ={ PathologicalProcess ⊑ InflammationProcess, InflammationProcess ⊑ GranulomaProcess }.

Figure A.8 shows the ontology after repair R6 is executed, where R6 = (A_6,D_6) with A_6 = { Carditis ⊑ CardioVascularDisease, GranulomaProcess ⊑ PathologicalProcess } and D_6 = D_3 = D_4 = D_5 = { PathologicalProcess ⊑ InflammationProcess, InflammationProcess ⊑ GranulomaProcess }. We show that R6 is a repair. (i) Regarding the statements in A_6, we know that *Or*(Carditis ⊑ CardioVascularDisease) = true, and *Or*(GranulomaProcess ⊑ PathologicalProcess) = true. (ii) Same as R3. (iii) Same as R1. (iv) $(\mathcal{T} \cup A_6) \setminus D_6 \models$ Endocarditis ⊑ PathologicalPhenomenon, as this can be derived from the existing axiom Endocarditis ⊑ Carditis, the added axiom Carditis ⊑ CardioVascularDisease and the existing axiom CardioVascularDisease ⊑ PathologicalPhenomenon. $(\mathcal{T} \cup A_6) \setminus D_6 \models$ GranulomaProcess ⊑ NonNormalProcess, for the same reason as in R4. (v) Same as R3.

Figure A.9 shows the ontology after repair R7 is executed, where R7 = (A_7,D_7) with A_7 = { GranulomaProcess ⊑ InflammationProcess, InflammationProcess ⊑ PathologicalProcess } and D_7 = D_3 = D_4 = D_5 = D_6 = { PathologicalProcess ⊑ InflammationProcess, InflammationProcess ⊑ GranulomaProcess }. We show that R7 is a repair. (i) Regarding the statements in A_7, we know that *Or*(GranulomaProcess ⊑ InflammationProcess) = true, and *Or*(InflammationProcess ⊑

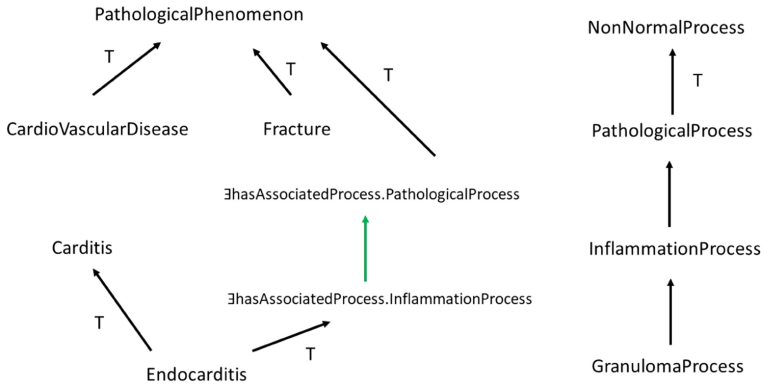

Fig. A.9. Result after repairing the ontology in the complete-debug-problem in Figure A.1 with repair R7. R7 = (A_7,D_7) with A_7 = { GranulomaProcess ⊑ InflammationProcess, InflammationProcess ⊑ PathologicalProcess }, D_7 = { PathologicalProcess ⊑ InflammationProcess, InflammationProcess ⊑ GranulomaProcess }.

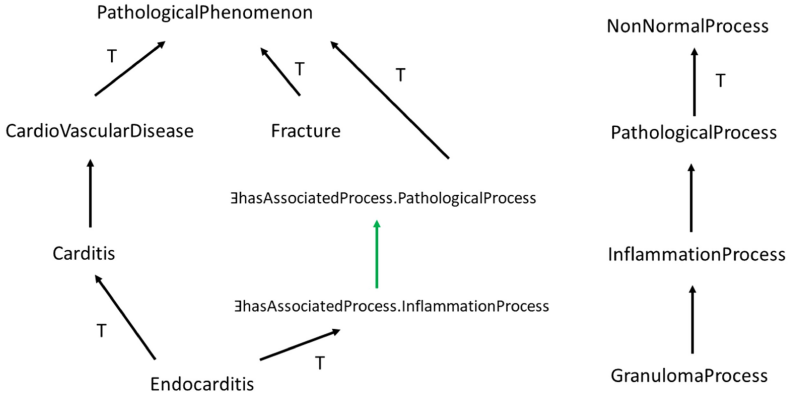

Fig. A.10. Result after repairing the ontology in the complete-debug-problem in Figure A.1 with repair R8. R8 = (A_8,D_8) with A_8 = { Carditis ⊑ CardioVascularDisease, GranulomaProcess ⊑ InflammationProcess, InflammationProcess ⊑ PathologicalProcess }, D_8 = { PathologicalProcess ⊑ InflammationProcess, InflammationProcess ⊑ GranulomaProcess }.

PathologicalProcess) = true. (ii) Same as R3. (iii) Same as R1. (iv) $(\mathcal{T} \cup A_7) \setminus D_7 \models$ Endocarditis ⊑ PathologicalPhenomenon, as this can be derived from the existing axiom Endocarditis ⊑ ∃hasAssociatedProcess.InflammationProcess, the newly derivable axiom ∃hasAssociatedProcess.InflammationProcess ⊑ ∃hasAssociatedProcess.PathologicalProcess and the existing axiom ∃hasAssociatedProcess.PathologicalProcess ⊑ PathologicalPhenomenon. The axiom ∃hasAssociatedProcess.InflammationProcess ⊑ ∃hasAssociatedProcess.Pathological Process can be derived from InflammationProcess ⊑ PathologicalProcess, which is newly added. $(\mathcal{T} \cup A_7) \setminus D_7 \models$ 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.

Figure A.10 shows the ontology after repair R8 is executed, where R8 = (A_8,D_8) with A_8 = { Carditis ⊑ CardioVascularDisease, GranulomaProcess ⊑ InflammationProcess, Inflammation-

Process ⊑ PathologicalProcess } and D_8 = D_3 = D_4 = D_5 = D_6 = D_7 = { PathologicalProcess ⊑ InflammationProcess, InflammationProcess ⊑ GranulomaProcess }. We show that R8 is a repair. (i) Regarding the statements in A_8, 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) $(\mathcal{T} \cup A_8) \setminus D_8 \models$ Endocarditis ⊑ PathologicalPhenomenon, as this can be derived from the existing axiom Endocarditis ⊑ ∃hasAssociatedProcess.InflammationProcess, the newly derivable axiom ∃hasAssociatedProcess.InflammationProcess ⊑ ∃hasAssociatedProcess.PathologicalProcess and the existing ∃hasAssociatedProcess.PathologicalProcess ⊑ PathologicalPhenomenon. The axiom ∃hasAssociatedProcess.InflammationProcess ⊑ ∃hasAssociatedProcess.PathologicalProcess can be derived from InflammationProcess ⊑ 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.) $(T \cup A_8) \setminus D_8 \models$ 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_{ax} 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 ⊓ O ⊑ 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_{ax}. Furthermore, all repairs need to make sure the axioms in M are derivable and thus also these are in R_{ax}. Then, R_{ax} = { CardioVascularDisease ⊑ PathologicalPhenomenon, Fracture ⊑ PathologicalPhenomenon, ∃hasAssociatedProcess.PathologicalProcess ⊑ PathologicalPhenomenon, Endocarditis ⊑ Carditis, Endocarditis ⊑ ∃hasAssociatedProcess.InflammationProcess, PathologicalProcess ⊑ NonNormalProcess, GranulomaProcess ⊑ NonNormalProcess, Endocarditis ⊑ PathologicalPhenomenon }.

The set of axioms derivable from the ontology repaired by R1 is $R1_{ax}$ = R_{ax} ∪ { 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.

For R2 and R3 the corresponding sets of axioms are $R2_{ax}$ = $R3_{ax}$ = R_{ax}, so no additional correct axioms.

$R4_{ax}$ = R_{ax} ∪ { GranulomaProcess ⊑ PathologicalProcess}.

$R5_{ax}$ = R_{ax} ∪ { GranulomaProcess ⊑ PathologicalProcess, Carditis ⊑ PathologicalPhenomenon } = $R4_{ax}$ ∪ { Carditis ⊑ PathologicalPhenomenon }.

$R6_{ax}$ = R_{ax} ∪ { GranulomaProcess ⊑ PathologicalProcess, Carditis ⊑ PathologicalPhenomenon, Carditis ⊑ CardioVascularDisease, Endocarditis ⊑ CardioVascularDisease } = $R5_{ax}$ ∪ { Carditis ⊑ CardioVascularDisease, Endocarditis ⊑ CardioVascularDisease }.

$R7_{ax}$ = R_{ax} ∪ { GranulomaProcess ⊑ InflammationProcess, GranulomaProcess ⊑ PathologicalProcess, InflammationProcess ⊑ PathologicalProcess, InflammationProcess ⊑ NonNormalProcess,

∃hasAssociatedProcess.InflammationProcess ⊑ PathologicalPhenomenon, Endocarditis ⊑ ∃hasAssociatedProcess.PathologicalPhenomenon } = $R1_{ax}$ ∪ { GranulomaProcess ⊑ InflammationProcess, GranulomaProcess ⊑ PathologicalProcess, InflammationProcess ⊑ PathologicalProcess, ∃hasAssociatedProcess.InflammationProcess ⊑ PathologicalPhenomenon, Endocarditis ⊑ ∃hasAssociatedProcess.PathologicalPhenomenon }

$R8_{ax}$ = R_{ax} ∪ { GranulomaProcess ⊑ InflammationProcess, GranulomaProcess ⊑ PathologicalProcess, InflammationProcess ⊑ PathologicalProcess, InflammationProcess ⊑ NonNormalProcess, Carditis ⊑ PathologicalPhenomenon, Carditis ⊑ CardioVascularDisease, Endocarditis ⊑ CardioVascularDisease, ∃hasAssociatedProcess.InflammationProcess ⊑ PathologicalPhenomenon, Endocarditis ⊑ ∃hasAssociatedProcess.PathologicalPhenomenon } = $R6_{ax}$ ∪ $R7_{ax}$

Thus, R_{ax} = $R2_{ax}$ = $R3_{ax}$ ⊊ $R1_{ax}$ ⊊ $R7_{ax}$ ⊊ $R8_{ax}$ and R_{ax} = $R2_{ax}$ = $R3_{ax}$ ⊊ $R4_{ax}$ ⊊ $R5_{ax}$ ⊊ $R6_{ax}$ ⊊ $R8_{ax}$. From this, we conclude that R8 is more complete than R7, which is more complete than R1, which is more complete than R2 and R3. Further, R8 is more complete than R6, which is more complete than R5, which is more complete than R4, which is more complete then R2 and R3.

Subset

For the add and delete sets for the repairs R1–R8, we obtain the following:
A_1 = A_2 = A_3, A_7 ⊊ A_8, D_1 ⊊ D_3 = D_4 = D_5 = D_6 = D_7 = D_8, and D_2 ⊊ D_3 = D_4 = D_5 = D_6 = D_7 = D_8. Therefore, R1 ⊂ R3, R2 ⊂ R3 and R7 ⊂ R8. R3 deletes more wrong information than R1 and R2, respectively, but for repairing this is redundant (although R3 is less incorrect than R1 and R2). R8 adds additional correct axioms compared to R7, but this is redundant for the repairing (although R8 is more complete than R7).

Preferred Repairs

R8 is maximally complete, as the ontology repaired by R8 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.

R1 and R2 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 R8. R8 is more complete-optimal with respect to { less incorrect }, as the ontology repaired by R8 does not contain wrong information. It is not more complete-optimal with respect to { ⊂ }, as there are repairs that are also preferred with respect to more complete, but that remove one fewer wrong axiom. However, R8 is more complete-optimal with respect to { less incorrect, ⊂ }. If there would be a preferred repair with respect to more complete that dominates R8 with respect to { less incorrect, ⊂ }, then it would have to be more preferred to R8 with respect to ⊂, as it cannot be more preferred with respect to less incorrect. However, removing an added axiom from R8 would make the repair not preferred with respect to more complete and removing fewer deleted axioms would make the repair less incorrect than R8.

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, ⊂ }. 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 { ⊂ }, 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 $\subset$-optimal, the candidates are R1 and R2. R1 is more complete than R2, so R2 cannot be $\subset$-optimal with respect to { more complete }. Also, R1 is not $\subset$-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 $\sqsubseteq$ CardioVascularDisease in the add set instead of Endocarditis $\sqsubseteq$ PathologicalPhenomenon). For similar reasons, R1 and R2 are not $\subset$-optimal with respect to { more complete, less incorrect }. They are, however, $\subset$-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 skyline-optimal repairs that are not X-optimal. For instance, R1 is not more-complete-optimal with respect to { $\subset$ } nor $\subset$-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.

ACKNOWLEDGMENTS

We thank Mina Abd Nikooie Pour, Ying Li, and Chunyan Wang for proofreading and checking the examples.

REFERENCES

[1] M. Fareed Arif, Carlos Mencía, Alexey Ignatiev, Norbert Manthey, Rafael Peñaloza, and João Marques-Silva. 2016. BEACON: An efficient SAT-based tool for debugging $\mathcal{EL}^+$ ontologies. In *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5–8, 2016, Proceedings (Lecture Notes in Computer Science)*, Nadia Creignou and Daniel Le Berre (Eds.), Vol. 9710. Springer, 521–530. DOI:https://doi.org/10.1007/978-3-319-40970-2_32

[2] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider (Eds.). 2003. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press.

[3] Franz Baader, Bernhard Ganter, Baris Sertkaya, and Ulrike Sattler. 2007. Completing description logic knowledge bases using formal concept analysis. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6–12, 2007*, Manuela M. Veloso (Ed.). 230–235. Retrieved from http://ijcai.org/Proceedings/07/Papers/035.pdf.

[4] Franz Baader, Ian Horrocks, Carsten Lutz, and Uli Sattler (Eds.). 2017. *An Introduction to Description Logic.* Cambridge University Press.

[5] Franz Baader, Francesco Kriegel, Adrian Nuradiansyah, and Rafael Peñaloza. 2018. Making repairs in description logics more gentle. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference, KR 2018, Tempe, Arizona, 30 October–2 November 2018.*, Michael Thielscher, Francesca Toni, and Frank Wolter (Eds.). 319–328. Retrieved from https://aaai.org/ocs/index.php/KR/KR18/paper/view/18056.

[6] Franz Baader and Werner Nutt. 2003. Basic description logics. In *The Description Logic Handbook*, Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider (Eds.). Cambridge University Press, 43–95.

[7] Franz Baader and Rafael Peñaloza. 2010. Automata-based axiom pinpointing. *J. Autom. Reason.* 45, 2 (2010), 91–129. DOI: https://doi.org/10.1007/s10817-010-9181-2

[8] Franz Baader and Rafael Peñaloza. 2010. Axiom pinpointing in general tableaux. *J. Logic Computat.* 20, 1 (2010), 5–34. DOI: https://doi.org/10.1093/logcom/exn058

[9] Franz Baader and Baris Sertkaya. 2009. Usability issues in description logic knowledge base completion. In *Formal Concept Analysis, 7th International Conference, ICFCA 2009, Darmstadt, Germany, May 21–24, 2009, Proceedings (Lecture Notes in Computer Science)*, Sébastien Ferré and Sebastian Rudolph (Eds.), Vol. 5548. Springer, 1–21. DOI: https://doi.org/10.1007/978-3-642-01815-2_1

[10] Michael Bada and Lawrence Hunter. 2008. Identification of OBO nonalignments and its implications for OBO enrichment. *Bioinformatics* 24, 12 (2008), 1448–1455. DOI: https://doi.org/10.1093/bioinformatics/btn194

[11] Elena Beisswanger and Udo Hahn. 2012. Towards valid and reusable reference alignments—Ten basic quality checks for ontology alignments and their application to three different reference data sets. *J. Biomed. Semant.* 3, Suppl 1 (2012), S4:1–S4:14. DOI: https://doi.org/10.1186/2041-1480-3-S1-S4

[12] David A. Bell, Guilin Qi, and Weiru Liu. 2007. Approaches to inconsistency handling in description-logic based ontologies. In *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops, OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, OTM Academy Doctoral Consortium, MONET, OnToContent, ORM, PerSys, PPN, RDDS, SSWS, and SWWS 2007, Vilamoura, Portugal, November 25–30, 2007, Proceedings, Part II (Lecture Notes in Computer Science)*, Robert Meersman, Zahir Tari, and Pilar Herrero (Eds.), Vol. 4806. Springer, 1303–1311. DOI: https://doi.org/10.1007/978-3-540-76890-6_58

[13] Stephan Börzsönyi, Donald Kossmann, and Konrad Stocker. 2001. The skyline operator. In *Proceedings 17th International Conference on Data Engineering*. 421–430. DOI: https://doi.org/10.1109/ICDE.2001.914855

[14] Paul Buitelaar, Philipp Cimiano, and Bernado Magnini. 2005. *Ontology Learning from Text: Methods, Evaluation and Applications*. IOS Press.

[15] Philipp Cimiano, Andreas Hotho, and Steffen Staab. 2005. Learning concept hierarchies from text corpora using formal concept analysis. *J. Artif. Intell. Res.* 24 (2005), 305–339. DOI: https://doi.org/10.1613/jair.1648

[16] Roberto Confalonieri, Pietro Galliani, Oliver Kutz, Daniele Porello, Guendalina Righetti, and Nicolas Troquard. 2020. Towards even more irresistible axiom weakening. In *Proceedings of the 33rd International Workshop on Description Logics (DL 2020) (CEUR Workshop Proceedings)*, Stefan Borgwardt and Thomas Meyer (Eds.), Vol. 2663. Retrieved from https://ceur-ws.org/Vol-2663/paper-8.pdf.

[17] Oscar Corcho, Catherine Roussey, Luis Manuel Vilches Blázquez, and Iván Pérez. 2009. Pattern-based OWL ontology debugging guidelines. In *Workshop on Ontology Patterns (CEUR Workshop Proceedings)*, Eva Blomqvist, Kurt Sandkuhl, Francois Scharffe, and Vojtech Svatek (Eds.), Vol. 516. 68–82. Retrieved from http://ceur-ws.org/Vol-516/pap02.pdf.

[18] Mathieu d'Aquin and Natalya F. Noy. 2012. Where to publish and find ontologies? A survey of ontology libraries. *J. Web Semant.* 11 (2012), 96–111. DOI: https://doi.org/10.1016/j.websem.2011.08.005

[19] Thinh Dong, Chan Le Duc, and Myriam Lamolle. 2017. Tableau-based revision for expressive description logics with individuals. *J. Web Semant.* 45 (2017), 63–79. DOI: https://doi.org/10.1016/j.websem.2017.09.001

[20] Zlatan Dragisic, Valentina Ivanova, Patrick Lambrix, Daniel Faria, Ernesto Jiménez-Ruiz, and Catia Pesquita. 2016. User validation in ontology alignment. In *The Semantic Web - ISWC 2016, 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part I (Lecture Notes in Computer Science)*, Paul Groth, Elena Simperl, Alasdair Gray, Marta Sabou, Markus Krötzsch, Freddy Lecue, Fabian Flöck, and Yolanda Gil (Eds.), Vol. 9981. Springer, 200–217. DOI: https://doi.org/10.1007/978-3-319-46523-4_13

[21] Zlatan Dragisic, Valentina Ivanova, Huanyu Li, and Patrick Lambrix. 2017. Experiences from the anatomy track in the ontology alignment evaluation initiative. *J. Biomed. Semant.* 8, 1 (2017), 56:1–56:28. DOI: https://doi.org/10.1186/s13326-017-0166-5

[22] Zlatan Dragisic, Patrick Lambrix, and Eva Blomqvist. 2015. Integrating ontology debugging and matching into the eXtreme design methodology. In *Proceedings of the 6th Workshop on Ontology and Semantic Web Patterns (CEUR Workshop Proceedings)*, Eva Blomqvist, Pascal Hitzler, Adila Krisnadhi, Tom Narock, and Monika Solanki (Eds.), Vol. 1461. Retrieved from http://ceur-ws.org/Vol-1461/WOP2015_paper_1.pdf.

[23] Zlatan Dragisic, Ying Li, and Patrick Lambrix. 2021. RepOSE-CTab - A Protégé plugin for completing ontologies. In *Proceedings of the Sixth International Workshop on the Visualization and Interaction for Ontologies and Linked Data co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual Conference, 2021 (CEUR Workshop Proceedings)*, Patrick Lambrix, Catia Pesquita, and Vitalis Wiens (Eds.), Vol. 3023. CEUR-WS.org, 56–62. Retrieved from http://ceur-ws.org/Vol-3023/paper1.pdf.

[24] Jianfeng Du, Guilin Qi, and Xuefeng Fu. 2014. A practical fine-grained approach to resolving incoherent OWL 2 DL terminologies. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3–7, 2014*, Jianzhong Li, Xiaoyang Sean Wang, Minos N. Garofalakis, Ian Soboroff, Torsten Suel, and Min Wang (Eds.). 919–928. DOI: https://doi.org/10.1145/2661829.2662046

[25] Jianfeng Du, Hai Wan, and Huaguan Ma. 2017. Practical TBox abduction based on justification patterns. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4–9, 2017, San Francisco, California, USA*. Satinder P. Singh and Shaul Markovitch (Eds.). 1100–1106. Retrieved from https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14402.

[26] Jérôme Euzenat. 2015. Revision in networks of ontologies. *Artif. Intell.* 228 (2015), 195–216. DOI: https://doi.org/10.1016/j.artint.2015.07.007

[27] Jérôme Euzenat. 2017. Interaction-based ontology alignment repair with expansion and relaxation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19–25, 2017*, Carles Sierra (Ed.). 185–191. DOI: https://doi.org/10.24963/ijcai.2017/27

[28] Jerome Euzenat and Pavel Shvaiko. 2007. *Ontology Matching*. Springer.

[29] Sean M. Falconer and Margaret-Anne D. Storey. 2007. A cognitive support framework for ontology mapping. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11–15, 2007 (Lecture Notes in Computer Science)*, Karl Aberer, Key-Sun Choi, Natasha Fridman Noy, Dean Allemang, Kyung-Il Lee, Lyndon J. B. Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux (Eds.), Vol. 4825. Springer, 114–127. DOI: https://doi.org/10.1007/978-3-540-76298-0_9

[30] Daniel Fleischhacker, Christian Meilicke, Johanna Völker, and Mathias Niepert. 2013. Computing incoherence explanations for learned ontologies. In *Web Reasoning and Rule Systems - 7th International Conference, RR 2013, Mannheim, Germany, July 27–29, 2013. Proceedings (Lecture Notes in Computer Science)*, Wolfgang Faber and Domenico Lembo (Eds.), Vol. 7994. Springer, 80–94. DOI: https://doi.org/10.1007/978-3-642-39666-3_7

[31] Gerhard Friedrich and Kostyantyn Shchekotykhin. 2005. A general diagnosis method for ontologies. In *The Semantic Web – ISWC 2005*, Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen (Eds.). Springer, Berlin, 232–246.

[32] Xuefeng Fu, Guilin Qi, Yong Zhang, and Zhangquan Zhou. 2016. Graph-based approaches to debugging and revision of terminologies in DL-Lite. *Knowl.-based Syst.* 100 (2016), 1–12. DOI: https://doi.org/10.1016/j.knosys.2016.01.039

[33] Peter Haase and Guilin Qi. 2007. An analysis of approaches to resolving inconsistencies in DL-based ontologies. In *International Workshop on Ontology Dynamics*, Giorgos Flouris and Mathieu d'Aquin (Eds.). 97–109. Retrieved from http://kmi.open.ac.uk/events/iwod/papers/paper-13.pdf.

[34] Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*. 539–545. DOI: https://doi.org/10.3115/992133.992154

[35] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. 2008. Laconic and precise justifications in OWL. In *The Semantic Web - ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26–30, 2008. Proceedings (Lecture Notes in Computer Science)*, Amit P. Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy W. Finin, and Krishnaprasad Thirunarayan (Eds.), Vol. 5318. Springer, 323–338. DOI: https://doi.org/10.1007/978-3-540-88564-1_21

[36] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. 2009. Explaining inconsistencies in OWL ontologies. In *Scalable Uncertainty Management, Third International Conference, SUM 2009, Washington, DC, USA, September 28–30, 2009. Proceedings (Lecture Notes in Computer Science)*, Lluís Godo and Andrea Pugliese (Eds.), Vol. 5785. Springer, 124–137. DOI: https://doi.org/10.1007/978-3-642-04388-8_11

[37] Valentina Ivanova and Patrick Lambrix. 2013. A unified approach for aligning taxonomies and debugging taxonomies and their alignments. In *The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26–30, 2013. Proceedings*, Philipp Cimiano, Óscar Corcho, Valentina Presutti, Laura Hollink, and Sebastian Rudolph (Eds.). 1–15. DOI: https://doi.org/10.1007/978-3-642-38288-8_1

[38] Valentina Ivanova, Patrick Lambrix, and Johan Åberg. 2015. Requirements for and evaluation of user support for large-scale ontology alignment. In *The Semantic Web. Latest Advances and New Domains - 12th European Semantic Web Conference, ESWC 2015, Portoroz, Slovenia, May 31–June 4, 2015. Proceedings (Lecture Notes in Computer Science)*, Fabien Gandon, Marta Sabou, Harald Sack, Claudia d'Amato, Philippe Cudré-Mauroux, and Antoine Zimmermann (Eds.), Vol. 9088. Springer, 3–20. DOI: https://doi.org/10.1007/978-3-319-18818-8_1

[39] Valentina Ivanova, Patrick Lambrix, Steffen Lohmann, and Catia Pesquita. 2019. Visualization and interaction for ontologies and linked data—Editorial. *J. Web Semant.* 55 (2019), 145–149. DOI: https://doi.org/10.1016/j.websem.2018.10.001

[40] Qiu Ji, Zhiqiang Gao, Zhisheng Huang, and Man Zhu. 2011. An efficient approach to debugging ontologies based on patterns. In *The Semantic Web - Joint International Semantic Technology Conference, JIST 2011, Hangzhou, China,*

December 4–7, 2011. Proceedings, Jeff Z. Pan, Huajun Chen, Hong-Gee Kim, Juanzi Li, Zhe Wu, Ian Horrocks, Riichiro Mizoguchi, and Zhaohui Wu (Eds.). 425–433. DOI: https://doi.org/10.1007/978-3-642-29923-0_33

[41] Qiu Ji, Zhiqiang Gao, Zhisheng Huang, and Man Zhua. 2014. Measuring effectiveness of ontology debugging systems. *Knowl.-based Syst.* 71 (2014), 169–186. DOI: https://doi.org/10.1016/j.knosys.2014.07.023

[42] Qiu Ji, Peter Haase, Guilin Qi, Pascal Hitzler, and Steffen Stadtmüller. 2009. RaDON—Repair and Diagnosis in Ontology Networks. In *The Semantic Web: Research and Applications. ESWC 2009.* Springer, 863–867. DOI: https://doi.org/10.1007/978-3-642-02121-3_71

[43] Qiu Ji, Guilin Qi, and Peter Haase. 2009. A relevance-directed algorithm for finding justifications of DL entailments. In *The Semantic Web, Fourth Asian Conference, ASWC 2009, Shanghai, China, December 6–9, 2009. Proceedings*, Asunción Gómez-Pérez, Yong Yu, and Ying Ding (Eds.). 306–320. DOI: https://doi.org/10.1007/978-3-642-10871-6_21

[44] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. 2011. LogMap: Logic-based and scalable ontology matching. In *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23–27, 2011, Proceedings, Part I*, Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Fridman Noy, and Eva Blomqvist (Eds.). 273–288. DOI: https://doi.org/10.1007/978-3-642-25073-6_18

[45] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Yujiao Zhou, and Ian Horrocks. 2012. Large-scale interactive ontology matching: Algorithms and implementation. In *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27–31, 2012*, Luc De Raedt, Christian Bessière, Didier Dubois, Patrick Doherty, Paolo Frasconi, Fredrik Heintz, and Peter J. F. Lucas (Eds.). 444–449. DOI: https://doi.org/10.3233/978-1-61499-098-7-444

[46] Ernesto Jiménez-Ruiz, Christian Meilicke, Bernardo Cuenca Grau, and Ian Horrocks. 2013. Evaluating mapping repair systems with large biomedical ontologies. In *Informal Proceedings of the 26th International Workshop on Description Logics, Ulm, Germany, July 23–26, 2013 (CEUR Workshop Proceedings)*, Thomas Eiter, Birte Glimm, Yevgeny Kazakov, and Markus Krötzsch (Eds.), Vol. 1014. 246–257. Retrieved from http://ceur-ws.org/Vol-1014/paper_63.pdf.

[47] Yannis Kalfoglou and Marco Schorlemmer. 2003. Ontology mapping: The state of the art. *Knowl. Eng. Rev.* 18, 1 (2003), 1–31. DOI: https://doi.org/10.1017/S0269888903000651

[48] Aditya Kalyanpur, Bijan Parsia, Matthew Horridge, and Evren Sirin. 2007. Finding all justifications of OWL DL entailments. In *The Semantic Web - 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11–15, 2007. Proceedings*, Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudre-Mauroux (Eds.). 267–280. DOI: https://doi.org/10.1007/978-3-540-76298-0_20

[49] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and Bernardo Cuenca-Grau. 2006. Repairing unsatisfiable concepts in OWL ontologies. In *The Semantic Web: Research and Applications 3rd European Semantic Web Conference, ESWC 2006 Budva, Montenegro, June 11–14, 2006 Proceedings*, York Sure and John Domingue (Eds.). 170–184. DOI: https://doi.org/10.1007/11762256_15

[50] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and James Hendler. 2005. Debugging unsatisfiable classes in OWL ontologies. *J. Web Semant.* 3, 4 (2005), 268–293. DOI: https://doi.org/10.1016/j.websem.2005.09.005

[51] Yevgeny Kazakov and Peter Skocovský. 2018. Enumerating justifications using resolution. In *Automated Reasoning - 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14–17, 2018, Proceedings (Lecture Notes in Computer Science)*, Didier Galmiche, Stephan Schulz, and Roberto Sebastiani (Eds.), Vol. 10900. Springer, 609–626. DOI: https://doi.org/10.1007/978-3-319-94205-6_40

[52] C. Maria Keet. 2012. Detecting and revising flaws in OWL object property expressions. In *Knowledge Engineering and Knowledge Management - 18th International Conference, EKAW 2012, Galway City, Ireland, October 8–12, 2012. Proceedings*, Annette ten Teije, Johanna Völker, Siegfried Handschuh, Heiner Stuckenschmidt, Mathieu d'Aquin, Andriy Nikolov, Nathalie Aussenac-Gilles, and Nathalie Hernandez (Eds.). 252–266. DOI: https://doi.org/10.1007/978-3-642-33876-2_23

[53] Joey Sik Chun Lam, Derek H. Sleeman, Jeff Z. Pan, and Wamberto Weber Vasconcelos. 2008. A fine-grained approach to resolving unsatisfiable ontologies. *J. Data Semant.* 10 (2008), 62–95. DOI: https://doi.org/10.1007/978-3-540-77688-8_3

[54] Patrick Lambrix, Zlatan Dragisic, and Valentina Ivanova. 2012. Get my pizza right: Repairing missing is—A relations in ALC ontologies. In *Semantic Technology, Second Joint International Conference, JIST 2012, Nara, Japan, December 2–4, 2012. Proceedings*, Hideaki Takeda, Yuzhong Qu, Riichiro Mizoguchi, and Yoshinobu Kitamura (Eds.). 17–32. DOI: https://doi.org/10.1007/978-3-642-37996-3_2

[55] Patrick Lambrix and Valentina Ivanova. 2013. A unified approach for debugging is-a structure and mappings in networked taxonomies. *J. Biomed. Semant.* 4 (2013), 10. DOI: https://doi.org/10.1186/2041-1480-4-10

[56] Patrick Lambrix and Rajaram Kaliyaperumal. 2017. A session-based ontology alignment approach enabling user involvement. *Semant Web J.* 8, 2 (2017), 225–251. DOI: https://doi.org/10.3233/SW-160243

[57] Patrick Lambrix and Qiang Liu. 2013. Debugging the missing is-a structure within taxonomies networked by partial reference alignments. *Data Knowl. Eng.* 86 (2013), 179–205. DOI: https://doi.org/10.1016/j.datak.2013.03.003

[58] Patrick Lambrix, Qiang Liu, and He Tan. 2009. Repairing the missing is-a structure of ontologies. In *The Semantic Web, Fourth Asian Conference, ASWC 2009, Shanghai, China, December 6–9, 2009. Proceedings (Lecture Notes in Computer Science)*, Asuncion Gomez-Perez, Yong Yu, and Ying Ding (Eds.), Vol. 5926. Springer, 76–90. DOI: https://doi.org/10.1007/978-3-642-10871-6_6

[59] Patrick Lambrix, Lena Strömbäck, and He Tan. 2009. Information integration in bioinformatics with ontologies and standards. In *Semantic Techniques for the Web: The REWERSE perspective*, Francois Bry and Jan Maluszynski (Eds.). *Lecture Notes in Computer Science*, Vol. 5500. Springer, 343–376. DOI: https://doi.org/10.1007/978-3-642-04581-3_8

[60] Patrick Lambrix, Fang Wei-Kleiner, and Zlatan Dragisic. 2015. Completing the is-a structure in light-weight ontologies. *J. Biomed. Semant.* 6 (2015), 12:1–12:26. DOI: https://doi.org/10.1186/s13326-015-0002-8

[61] Jens Lehmann and Lorenz Bühmann. 2010. ORE—A tool for repairing and enriching knowledge bases. In *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7–11, 2010, Revised Selected Papers, Part II (Lecture Notes in Computer Science)*, Peter F. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang, Jeff Z. Pan, Ian Horrocks, and Birte Glimm (Eds.), Vol. 6497. Springer, 177–193. DOI: https://doi.org/10.1007/978-3-642-17749-1_12

[62] Huanyu Li, Zlatan Dragisic, Daniel Faria, Valentina Ivanova, Ernesto Jiménez-Ruiz, Patrick Lambrix, and Catia Pesquita. 2019. User validation in ontology alignment: Functional assessment and impact. *Knowl. Eng. Rev.* 34 (2019), e15. DOI: https://doi.org/10.1017/S0269888919000080

[63] Ying Li and Patrick Lambrix. 2023. Repairing $\mathcal{EL}$ ontologies using weakening and completing. In *The Semantic Web - 20th International Conference, ESWC 2023, Hersonissos, Crete, Greece, May 28–June 1, 2023, Proceedings (Lecture Notes in Computer Science)*, Catia Pesquita, Ernesto Jimenez-Ruiz, Jamie McCusker, Daniel Faria, Mauro Dragoni, Anastasia Dimou, Raphael Troncy, and Sven Hertling (Eds.), Vol. 13870. Springer, 298–315. DOI: https://doi.org/10.1007/978-3-031-33455-9_18

[64] Qiang Liu and Patrick Lambrix. 2010. A system for debugging missing is-a structure in networked ontologies. In *Data Integration in the Life Sciences, 7th International Conference, DILS 2010, Gothenburg, Sweden, August 25–27, 2010. Proceedings (Lecture Notes in Computer Science)*, Patrick Lambrix and Graham Kemp (Eds.), Vol. 6254. Springer, 50–57. DOI: https://doi.org/10.1007/978-3-642-15120-0_5

[65] A. Maedche, V. Pekar, and S. Staab. 2003. Ontology learning part one—On discovering taxonomic relations from the web. In *Web Intelligence*, Zhong, Liu, and Yao (Eds.). Springer, 301–320. DOI: https://doi.org/10.1007/978-3-662-05320-1_14

[66] Alexander Maedche and Steffen Staab. 2000. Discovering conceptual relations from text. In *ECAI 2000, Proceedings of the 14th European Conference on Artificial Intelligence, Berlin, Germany, August 20–25, 2000*, Werner Horn (Ed.). 321–325. Retrieved from http://frontiersinai.com/ecai/ecai2000/pdf/p0321.pdf.

[67] Christian Meilicke. 2011. *Alignment Incoherence in Ontology Matching*. Ph.D. Dissertation. University of Mannheim.

[68] Christian Meilicke, Heiner Stuckenschmidt, and Andrei Tamilin. 2007. Repairing ontology mappings. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22–26, 2007, Vancouver, British Columbia, Canada*, Robert C. Holte and Adele Howe (Eds.). 1408–1413. Retrieved from https://www.aaai.org/Papers/AAAI/2007/AAAI07-223.pdf.

[69] Christian Meilicke, Heiner Stuckenschmidt, and Andrei Tamilin. 2009. Reasoning support for mapping revision. *J. Logic Computat.* 19, 5 (2009), 807–829. DOI: https://doi.org/10.1093/logcom/exn047

[70] Thomas Meyer, Kevin Lee, Richard Booth, and Jeff Pan. 2006. Finding maximally satisfiable terminologies for the description logic ALC. In *AAAI'06 - Proceedings of the 21st National Conference on Artificial Intelligence*, Anthony Cohn (Ed.). AAAI Press, 269–274. Retrieved from https://www.aaai.org/Papers/AAAI/2006/AAAI06-043.pdf.

[71] Kodylan Moodley, Thomas Meyer, and Ivan José Varzinczak. 2011. Root justifications for ontology repair. In *Web Reasoning and Rule Systems - 5th International Conference, RR 2011, Galway, Ireland, August 29–30, 2011. Proceedings (Lecture Notes in Computer Science)*, Sebastian Rudolph and Claudio Gutiérrez (Eds.), Vol. 6902. Springer, 275–280. DOI: https://doi.org/10.1007/978-3-642-23580-1_24

[72] Nadeschda Nikitina, Sebastian Rudolph, and Birte Glimm. 2012. Interactive ontology revision. *J. Web Semant.* 12 (2012), 118–130. DOI: https://doi.org/10.1016/j.websem.2011.12.002

[73] Natalya F. Noy. 2004. Semantic integration: A survey of ontology-based approaches. *SIGMOD Rec.* 33(4) (2004), 65–70. DOI: https://doi.org/10.1145/1041410.1041421

[74] Ana Ozaki, Cosimo Persia, and Andrea Mazzullo. 2020. Learning query inseparable $\mathcal{ELH}$ ontologies. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7–12, 2020*. AAAI Press, 2959–2966. Retrieved from https://ojs.aaai.org/index.php/AAAI/article/view/5688.

[75] Rafael Peñaloza and Baris Sertkaya. 2010. Complexity of axiom pinpointing in the DL-lite family of description logics. In *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16–20, 2010, Proceedings*, Helder Coelho, Rudi Studer, and Michael J. Wooldridge (Eds.). 29–34. DOI: https://doi.org/10.3233/978-1-60750-606-5-29

[76] Rafael Peñaloza and Baris Sertkaya. 2010. On the complexity of axiom pinpointing in the EL family of description logics. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Twelfth International Conference, KR 2010, Toronto, Ontario, Canada, May 9–13, 2010*, Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski (Eds.). 280–289. Retrieved from https://www.aaai.org/ocs/index.php/KR/KR2010/paper/viewFile/1345/1629.

[77] Rafael Peñaloza and Baris Sertkaya. 2017. Understanding the complexity of axiom pinpointing in lightweight description logics. *Artif. Intell.* 250 (2017), 80–104. DOI: https://doi.org/10.1016/j.artint.2017.06.002

[78] Catia Pesquita, Daniel Faria, Emanuel Santos, and Francisco M. Couto. 2013. To repair or not to repair: Reconciling correctness and coherence in ontology reference alignments. In *Proceedings of the 8th International Workshop on Ontology Matching (CEUR Workshop Proceedings)*, Pavel Shvaiko, Jérôme Euzenat, Kavitha Srinivas, Ming Mao, and Ernesto Jiménez-Ruiz (Eds.), Vol. 1111. 13–24. Retrieved from http://ceur-ws.org/Vol-1111/om2013_Tpaper2.pdf.

[79] María Poveda-Villalón, Asunción Gómez-Pérez, and Mari Carmen Suárez-Figueroa. 2014. OOPS! (OntOlogy pitfall scanner!): An on-line tool for ontology evaluation. *Int. J. Semant. Web Inf. Syst.* 10, 2 (2014), 7–34. DOI: https://doi.org/10.4018/ijswis.2014040102

[80] Guilin Qi, Qiu Ji, and Peter Haase. 2009. A conflict-based operator for mapping revision. In *The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25–29, 2009. Proceedings*, Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan (Eds.). 521–536. DOI: https://doi.org/10.1007/978-3-642-04930-9_33

[81] Raymond Reiter. 1987. A theory of diagnosis from first principles. *Artif. Intell.* 32, 1 (1987), 57–95. DOI: https://doi.org/10.1016/0004-3702(87)90062-2

[82] Márcio Moretto Ribeiro and Renata Wassermann. 2009. Base revision for ontology debugging. *J. Logic Computat.* 19, 5 (2009), 721–743. DOI: https://doi.org/10.1093/logcom/exn048

[83] Tjitze Rienstra, Claudia Schon, and Steffen Staab. 2020. Concept contraction in the description logic EL. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12–18, 2020*, Diego Calvanese, Esra Erdem, and Michael Thielscher (Eds.). 723–732. DOI: https://doi.org/10.24963/kr.2020/74

[84] Patrick Rodler, Dietmar Jannach, Konstantin Schekotihin, and Philipp Fleiss. 2019. Are query-based ontology debuggers really helping knowledge engineers? *Knowl-based Syst.* 179 (2019), 92–107. DOI: https://doi.org/10.1016/j.knosys.2019.05.006

[85] Patrick Rodler and Wolfgang Schmid. 2018. On the impact and proper use of heuristics in test-driven ontology debugging. In *Rules and Reasoning - Second International Joint Conference, RuleML+RR 2018, Luxembourg, September 18–21, 2018, Proceedings (Lecture Notes in Computer Science)*, Christoph Benzmüller, Francesco Ricca, Xavier Parent, and Dumitru Roman (Eds.), Vol. 11092. Springer, 164–184. DOI: https://doi.org/10.1007/978-3-319-99906-7_11

[86] Catherine Roussey and Ondrej Zamazal. 2013. Antipattern detection: How to debug an ontology without a reasoner. In *Proceedings of the Second International Workshop on Debugging Ontologies and Ontology Mappings, Montpellier, France, May 27, 2013 (CEUR Workshop Proceedings)*, Patrick Lambrix, Guilin Qi, Matthew Horridge, and Bijan Parsia (Eds.), Vol. 999. CEUR-WS.org, 45–56. Retrieved from http://ceur-ws.org/Vol-999/paper4.pdf.

[87] Emanuel Santos, Daniel Faria, Catia Pesquita, and Francisco M. Couto. 2015. Ontology alignment repair through modularization and confidence-based heuristics. *PLoS One* 10(12):e0144807) (2015), 1–19. DOI: https://doi.org/10.1371/journal.pone.0144807

[88] Konstantin Schekotihin, Patrick Rodler, and Wolfgang Schmid. 2018. OntoDebug: Interactive ontology debugging plug-in for Protégé. In *Foundations of Information and Knowledge Systems - 10th International Symposium, FoIKS 2018, Budapest, Hungary, May 14–18, 2018, Proceedings (Lecture Notes in Computer Science)*, Flavio Ferrarotti and Stefan Woltran (Eds.), Vol. 10833. Springer, 340–359. DOI: https://doi.org/10.1007/978-3-319-90050-6_19

[89] Stefan Schlobach. 2005. Debugging and semantic clarification by pinpointing. In *The Semantic Web: Research and Applications - Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29–June 1, 2005. Proceedings*, Asuncion Gomez-Perez and Jerome Euzenat (Eds.). 226–240. DOI: https://doi.org/10.1007/11431053_16

[90] Stefan Schlobach and Ronald Cornet. 2003. Non-standard reasoning services for the debugging of description logic terminologies. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, Georg Gottlob and Toby Walsh (Eds.). 355–360. Retrieved from https://www.ijcai.org/Proceedings/03/Papers/053.pdf.

[91] Stefan Schlobach, Zhisheng Huang, Ronald Cornet, and Frank van Harmelen. 2007. Debugging incoherent terminologies. *J. Autom. Reason.* 39, 3 (2007), 317–349. DOI: https://doi.org/10.1007/s10817-007-9076-z

[92] Claudia Schon and Steffen Staab. 2017. Towards SPARQL instance-level update in the presence of OWL-DL TBoxes. In *Proceedings of the Joint Ontology Workshops 2017 Episode 3: The Tyrolean Autumn of Ontology, Bozen-Bolzano,*

Italy, September 21–23, 2017 (CEUR Workshop Proceedings), Stefano Borgo, Oliver Kutz, Frank Loebe, Fabian Neuhaus, Kemo Adrian, Mihailo Antovic, Valerio Basile, Martin Boeker, Diego Calvanese, Tommaso Caselli, Giorgio Colombo, Roberto Confalonieri, Laura Daniele, Jérôme Euzenat, Antony Galton, Dagmar Gromann, Maria M. Hedblom, Heinrich Herre, Inge Hinterwaldner, Andrea Janes, Ludger Jansen, Kris Krois, Antonio Lieto, Claudio Masolo, Rafael Peñaloza, Daniele Porello, Daniele Paolo Radicioni, Emilio M. Sanfilippo, Daniel Schober, Rossella Stufano, and Amanda Vizedom (Eds.). Vol. 2050. CEUR-WS.org. Retrieved from http://ceur-ws.org/Vol-2050/DEW_paper_2.pdf.

[93] Claudia Schon, Steffen Staab, Patricia Kügler, Philipp Kestel, Benjamin Schleich, and Sandro Wartzack. 2018. Metaproperty-guided deletion from the instance-level of a knowledge base. In *Knowledge Engineering and Knowledge Management - 21st International Conference, EKAW 2018, Nancy, France, November 12–16, 2018, Proceedings (Lecture Notes in Computer Science)*, Catherine Faron-Zucker, Chiara Ghidini, Amedeo Napoli, and Yannick Toussaint (Eds.), Vol. 11313. Springer, 407–423. DOI: https://doi.org/10.1007/978-3-030-03667-6_26

[94] Roberto Sebastiani and Michele Vescovi. 2009. Axiom pinpointing in lightweight description logics via Horn-SAT encoding and conflict analysis. In *Automated Deduction - CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, August 2–7, 2009. Proceedings (Lecture Notes in Computer Science)*, Renate A. Schmidt (Ed.), Vol. 5663. Springer, 84–99. DOI: https://doi.org/10.1007/978-3-642-02959-2_6

[95] Baris Sertkaya. 2009. OntoComP: A Protégé plugin for completing OWL ontologies. In *The Semantic Web: Research and Applications, 6th European Semantic Web Conference, ESWC 2009, Heraklion, Crete, Greece, May 31–June 4, 2009, Proceedings (Lecture Notes in Computer Science)*, Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Paslaru Bontas Simperl (Eds.), Vol. 5554. Springer, 898–902. DOI: https://doi.org/10.1007/978-3-642-02121-3_78

[96] Kostyantyn M. Shchekotykhin, Gerhard Friedrich, Philipp Fleiss, and Patrick Rodler. 2012. Interactive ontology debugging: Two query strategies for efficient fault localization. *J. Web Semant.* 12 (2012), 88–103. DOI: https://doi.org/10.1016/j.websem.2011.12.006

[97] Kostyantyn M. Shchekotykhin, Gerhard Friedrich, Patrick Rodler, and Philipp Fleiss. 2014. Sequential diagnosis of high cardinality faults in knowledge-bases by direct diagnosis generation. In *ECAI'14: Proceedings of the Twenty-first European Conference on Artificial Intelligence*. IOS Press, 813–818. DOI: https://doi.org/10.3233/978-1-61499-419-0-813

[98] Pavel Shvaiko and Jerome Euzenat. 2005. A survey of schema-based matching approaches. *J. Data Semant.* IV (2005), 146–171. DOI: https://doi.org/10.1007/11603412_5

[99] Pavel Shvaiko and Jerome Euzenat. 2013. Ontology matching: State of the art and future challenges. *IEEE Trans. Knowl. Data Eng.* 25, 1 (2013), 158–176. DOI: https://doi.org/10.1109/TKDE.2011.253

[100] Elena Simperl, Malgorzata Mochol, and Tobias Bürger. 2010. Achieving maturity: The state of practice in ontology engineering in 2009. *Int. J. Comput. Sci. Applic.* 7, 1 (2010), 45–65.

[101] Alessandro Solimando, Ernesto Jiménez-Ruiz, and Giovanna Guerrini. 2014. Detecting and correcting conservativity principle violations in ontology-to-ontology mappings. In *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19–23, 2014. Proceedings, Part II*, Peter Mika, Tania Tudorache, Abraham Bernstein, Chris Welty, Craig A. Knoblock, Denny Vrandecic, Paul T. Groth, Natasha F. Noy, Krzysztof Janowicz, and Carole A. Goble (Eds.). 1–16. DOI: https://doi.org/10.1007/978-3-319-11915-1_1

[102] Vassilis Spiliopoulos, George A. Vouros, and Vangelis Karkaletsis. 2010. On the discovery of subsumption relations for the alignment of ontologies. *J. Web Semant.* 8 (2010), 69–88. DOI: https://doi.org/10.1016/j.websem.2010.01.001

[103] Steffen Staab and Rudi Studer. 2009. *Handbook on Ontologies*. Springer-Verlag. DOI: https://doi.org/10.1007/978-3-540-92673-3

[104] Robert Stevens, Carole A. Goble, and Sean Bechhofer. 2000. Ontology-based knowledge representation for bioinformatics. *Brie. Bioinform.* 1, 4 (2000), 398–414. DOI: https://doi.org/10.1093/bib/1.4.398

[105] Nicolas Troquard, Roberto Confalonieri, Pietro Galliani, Rafael Peñaloza, Daniele Porello, and Oliver Kutz. 2018. Repairing ontologies via axiom weakening. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2–7, 2018*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). 1981–1988. Retrieved from https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17189.

[106] Markel Vigo, Samantha Bail, Caroline Jay, and Robert D. Stevens. 2014. Overcoming the pitfalls of ontology authoring: Strategies and implications for tool design. *Int. J. Hum.-comput. Stud.* 72, 12 (2014), 835–845. DOI: https://doi.org/10.1016/j.ijhcs.2014.07.005

[107] Peng Wang and Baowen Xu. 2008. Debugging ontology mappings: A static approach. *Comput. Inform.* 27 (2008), 21–36. Retrieved from http://www.cai.sk/ojs/index.php/cai/article/view/271/220.

[108] Fang Wei-Kleiner, Zlatan Dragisic, and Patrick Lambrix. 2014. Abduction framework for repairing incomplete EL ontologies: Complexity results and algorithms. In *Proceedings of the 28th National Conference on Artificial Intelligence*. 1120–1127. Retrieved from https://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8239/8547.

[109] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J. G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A. C't Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. 2016. The FAIR guiding principles for scientific data management and stewardship. *Scient. Data* 3 (2016), 160018:1–9. DOI: https://doi.org/10.1038/sdata.2016.18

[110] Elias Zavitsanos, Georgios Paliouras, George A. Vouros, and Sergios Petridis. 2007. Discovering subsumption hierarchies of ontology concepts from text corpora. In *IEEE/WIC/ACM International Conference on Web Intelligence*. 402–408. DOI: https://doi.org/10.1109/WI.2007.55

[111] Yu Zhang, Ruxian Yao, Dantong Ouyang, Jinfeng Gao, and Fang Liu. 2021. Debugging incoherent ontology by extracting a clash module and identifying root unsatisfiable concepts. *Knowl.-based Syst.* 223 (2021), 107043. DOI: https://doi.org/10.1016/j.knosys.2021.107043

[112] Dmitriy Zheleznyakov, Evgeny Kharlamov, Werner Nutt, and Diego Calvanese. 2019. On expansion and contraction of DL-Lite knowledge bases. *J. Web Semant.* 57 (2019). DOI: https://doi.org/10.1016/j.websem.2018.12.002

Received 21 April 2022; revised 7 January 2023; accepted 11 April 2023