Towards Visual Decision Support in Interactive Repair (Extended Abstract)

Christian Alrabbaa
Institute of Theoretical Computer Science, TU Dresden, Germany

Franz Baader
Institute of Theoretical Computer Science, TU Dresden, Germany

Raimund Dachselt
Interactive Media Lab Dresden, TU Dresden, Germany

Pratistha Kansakar
Institute of Theoretical Computer Science, TU Dresden, Germany

Julián Méndez
Interactive Media Lab Dresden, TU Dresden, Germany

Afshin Zanganeh
Interactive Media Lab Dresden, TU Dresden, Germany

Abstract

In the literature, interactive ontology repair approaches assume that users can always decide whether an axiom should be included in a repair, i.e., either kept or removed. In this paper, we present an interactive approach for repairing ontologies in which users are assisted by various *decision-support features* that provide additional information in cases of uncertainty. This work has been accepted at the 39th International Workshop on Description Logics (DL 2026)[1].

2012 ACM Subject Classification Theory of computation → Description logics; Human-centered computing → Visualization toolkits

Keywords and phrases Ontology Repair, Decision Support, Visualization, Explanations

Acknowledgements This work is funded by Deutsche Forschungsgemeinschaft (DFG) under Germany's Excellence Strategy: EXC 2050/2, 390696704 – "Centre for Tactile Internet with Human-in-the-Loop" (CeTI); by DFG grant 389792660 as part of TRR 248 – CPEC, see `https://perspicuous-computing.science`; and by Bundesministerium für Bildung und Forschung (BMBF) and Saxon State Ministry for Science, Culture and Tourism (SMWK) in Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI, SCADS22B).

1 Introduction

Even carefully hand-crafted ontologies may contain errors, and this problem is exacerbated for ontologies built (semi)automatically using machine learning or information retrieval tools. For example, in earlier versions of the large medical ontology SNOMED CT,[2] which at that time was a large hand-crafted TBox written in the Description Logic (DL) $\mathcal{EL}$, the concept "amputation of finger" was classified as a subconcept of "amputation of hand", which should clearly not be the case [24, 8]. As in the case of this example, errors in ontologies are usually noticed when reasoning produces a consequence that follows from the ontology, but does not hold in the application domain modeled by it. Getting rid of such an unwanted consequence by removing a minimal amount of information is usually called repair in the

[1] `https://dl-2026.github.io/`
[2] `https://www.snomed.org/`

DL literature [21, 12, 22, 15, 5, 3, 1], where the repair approaches differ with respect to how "removing a minimal amount of information" is interpreted. Optimal classical repairs (in the terminology of [5]) are maximal subsets of the ontology that do not have the unwanted consequence. Following Reiter's approach for model-based diagnosis [20], such repairs can be produced as follows. First, compute all minimal subsets of the ontology that still have the unwanted consequence, usually called justifications in the DL literature [21, 10, 11, 6]. Then compute all minimal hitting sets of the collection of all justifications, i.e., sets that contain at least one element of each justification, called diagnoses. The optimal classical repairs are obtained by removing the elements of one of the diagnoses from the ontology.

This and also the other cited repair approaches may in the worst case compute exponentially many repairs in the size of the original ontology, of which some will not even make sense in the application domain. It is not a good idea to compute all these repairs first, and then ask the ontology engineer building the ontology or the domain expert that has spotted the error (both called users in the following) to choose, among potentially exponentially many computed repairs, one that does make sense. For this reason, interactive repair approaches have been developed [19, 16, 25, 4, 14], where the user makes certain decisions (e.g., whether an axiom in a justification should be part of the diagnosis or not), which prune the search space for an appropriate application-conforming repair.

The order in which the user is asked to make the relevant decisions influences how many decisions need to be made overall before a repair is reached. For this reason, several of the interactive repair approaches cited above are concerned with defining appropriate orders, and this issue is also addressed by our repair tools. For example, when computing a hitting set, it makes sense to favor axioms that occur in many justifications, though this heuristic does not guarantee that a *minimal* hitting set is found. This kind of frequency sorting of axioms was, e.g., used in [12], and is also employed in our repair tools. In addition, we consider an order that uses the number of occurrences of axioms in diagnoses to compute an entropy score. As shown in [23], ordering axioms by entropy score helps to reduce the number of questions the user needs to answer.

Another problem that requires attention is that users may need help in deciding which decision (keep the axiom or remove it) to make. We had already observed this issue when working with our interactive ontology completion tool based on formal concept analysis [2], where users sometimes made wrong decisions, which had to be retracted in an appropriate way [7]. To address this problem, our interactive repair tools are equipped with several decision-support features, which provide the user with additional information that they cannot obtain from simply looking at the axiom in question. On the one hand, we allow the user to define certain important entailments that should be retained as far as possible. The user is then informed of the effect of their decision on these entailments: for each such entailment, our system computes the number of still reachable repairs in which the entailment is preserved after the decision. On the other hand, we provide information on how much the concept hierarchy or the ontology itself changes based on the decision to remove or keep an axiom. For the hierarchy, this information indicates which parts are affected by the current decision and how they change. For the ontology as a whole, we measure how much the best still-reachable repair differs from the current ontology with respect to the Hamming distance, where "best" refers to the repair that retains the largest number of axioms.

We have implemented these ideas in two interactive repair tools. The first is a standalone command-line tool, which displays the additional information generated by our decision-support features in a text-based way. The second is a web-based interactive prototype that visually supports the user along the repair process through a number of views: (1) a list of

Algorithm 1 Interactive Ontology Repair with Decision Support – Basic Algorithm

Input: $\mathcal{T}, \delta, \mathcal{E}, \mathit{sortMethod}$
Output: $\mathcal{R}$
$\mathfrak{J} \coloneqq \text{compAllJustifications}(\mathcal{T}, \delta)$; $\mathfrak{D} \coloneqq \text{compAllMinDiagnoses}(\mathfrak{J})$; $\mathit{toRemove} \coloneqq \emptyset$;
$\mathfrak{R} \coloneqq \text{compAllMaxRepairs}(\mathcal{T}, \mathfrak{D})$; $\mathcal{M} \coloneqq \text{compSortedAxioms}(\mathfrak{J}, \mathfrak{D}, \mathit{sortMethod})$;
$\mathcal{R} \coloneqq \mathcal{T}$;
foreach $\alpha \in \mathcal{M}$ **do**
  $\mathit{ans} \coloneqq \text{getUserAnswer}(\alpha)$;
  while $\mathit{ans} \notin \{\mathsf{Yes}, \mathsf{No}\}$ **do**
    $\mathcal{F} \coloneqq \text{getUserSelection}(\{\mathsf{Rate}, \mathsf{Dist}, \mathsf{Diff}\})$;
    if $\mathsf{Rate} \in \mathcal{F}$ **then**
      $\text{display}(\text{compRates}(\mathcal{R}, \mathfrak{R}, \mathcal{E}, \alpha))$;
    if $\mathsf{Dist} \in \mathcal{F}$ **then**
      $\text{display}(\text{compDistances}(\mathcal{R}, \text{getBestRepairs}(\mathcal{R}, \mathfrak{R}, \mathcal{E}, \alpha)))$;
    if $\mathsf{Diff} \in \mathcal{F}$ **then**
      $\text{display}(\text{compDifferences}(\mathcal{R}, \alpha))$;
    $\mathit{ans} \coloneqq \text{getUserAnswer}(\alpha)$;
  if $\mathit{ans} = \mathsf{No}$ **then**
    $\mathit{toRemove} \coloneqq \mathit{toRemove} \cup \{\alpha\}$;
  $\mathcal{R} \coloneqq \mathcal{T} \setminus \mathit{toRemove}$;
  if $\exists \mathcal{R}' \in \mathfrak{R}$ *such that* $\mathcal{R} \subseteq \mathcal{R}'$ **then**
    return $\mathcal{R}$;
return $\emptyset$;

justification axioms to decide upon, (2) a decision tree that can be expanded on demand, (3) multivariate views that visually encode the rates of preserving user-defined important entailments, (4) a class hierarchy diff view detailing the changes resulting from removing an axiom, and (5) comparative bars and shortcuts to the discussed distances and corresponding best repairs. Together, these views provide analytical capabilities not easily achievable otherwise. We provide static examples (Hospital, Pizza) on a live version of our visualization prototype at https://imldresden.github.io/vds-repair.

2 Interactive Ontology Repair with Decision Support

In this section, we present our interactive repair approach by first providing an overview and then describing the individual decision-support features in detail. Formally, we focus only on repairing the terminological part of an ontology; however, for readability, we occasionally use the terms TBox and ontology interchangeably.

We assume that we are given a TBox $\mathcal{T}$ and an undesired consequence δ (defect) of $\mathcal{T}$, i.e., $\mathcal{T} \models \delta$. In addition, we assume that the user specifies a set of important entailments $\mathcal{E}$ that are to be preserved as much as possible. As shown in Algorithm 1, we start by computing the set $\mathfrak{J}$ of all *justifications* [21, 10, 11, 13] of $\mathcal{T} \models \delta$. To construct a repair $\mathcal{R}$, the user must decide which axioms from the original ontology should be removed (*toRemove*), in particular those that contribute to the entailment of the defect. These axioms are precisely those that occur in the justifications. However, it is not always necessary to consider all of them. For instance, if $\mathfrak{J}$ contains a single justification, removing any one of its axioms is sufficient to obtain a repair. Therefore, we aim to minimize the number of questions while ensuring that

all justifications are hit. To this end, assume that the *sortMethod* allows us to compute a *sorted set* $\mathcal{M}$ of all axioms occurring in the justifications, ordered such that decisions about axioms that appear earlier increases the likelihood of reaching a repair sooner.

For every justification axiom $\alpha \in \mathcal{M}$, the user can decide to either keep or remove it. In cases where this decision is unclear, the user may instead select a combination of *decision-support features* $\mathcal{F}$ to assist them. The first feature provides information for each important entailment in $\mathcal{E}$. Specifically, it shows the *percentage* of repairs in which the entailment holds when α is retained, compared to the percentage of repairs in which it holds when α is removed. The purpose of this feature is to illustrate how the user's decision affects the preservation of important entailments. The second feature highlights the effect of the user's decision on the *class hierarchy* by showing how removing α changes those parts of the hierarchy whose entailments depend on α. The last feature informs the user about the best outcome they can achieve with respect to their decision. This is conveyed by showing how much the best repairs—one corresponding to retaining α and one to removing it—differ from the current TBox. Here, a best repair is defined as a repair that preserves the largest number of axioms from the original TBox, and this difference is quantified using a normalized *Hamming distance* over TBox axioms. Lastly, if the user's decisions preserve too many axioms of $\mathcal{T}$, thereby preventing a repair from being reached, the algorithm returns the empty set.

In the following, we present the three decision-support features in more detail.

Entailment Preservation Rate. When repairing a TBox $\mathcal{T}$, the user can provide an additional set $\mathcal{E}$ containing important entailments. Note that these entailments are not necessarily part of $\mathcal{T}$, but should be consequences of it. The purpose of this is to allow the user to incorporate domain knowledge into the repair process, which can then be used to provide additional insights. Specifically, whenever the user is unsure whether to keep or remove an axiom α, they can request the important *entailments preservation rate*. For each $\eta \in \mathcal{E}$, this feature provides two values: the percentage of reachable repairs in which η remains entailed when α is retained, and when α is removed. This feature allows the user to better understand how their decision affects the set of consequences they consider important.

Hierarchy Difference. When repairing $\mathcal{T}$, changes to its structure are inevitable. For instance, removing a subsumption axiom between concept names may alter the class hierarchy by modifying the set of *direct* super-concept relations between concept names, causing some direct relations to disappear while previously indirect ones become direct. Therefore, the purpose of this feature is to leverage these changes to provide additional information about the role of α in the hierarchical structure of $\mathcal{T}$. When the user is uncertain whether to keep or remove α, they are presented with two DAGs representing the class hierarchies of two possibilities: $\mathcal{T}_{\mathsf{In}} = \mathcal{T} \setminus \mathit{toRemove}$ and $\mathcal{T}_{\mathsf{Out}} = \mathcal{T} \setminus (\mathit{toRemove} \cup \{\alpha\})$, where vertices represent concept names and edges represent subsumption relations between them. The differences between the two structures are highlighted as follows: If an edge appears in the graph of $\mathcal{T}_{\mathsf{In}}$ but not in the graph of $\mathcal{T}_{\mathsf{Out}}$, it is marked as a removed edge. Conversely, if an edge appears in the graph of $\mathcal{T}_{\mathsf{Out}}$ but not in that of $\mathcal{T}_{\mathsf{In}}$, it is marked as an added edge. In essence, this feature allows the user to localize the effect of their decision on the structure of $\mathcal{T}$.

Dissimilarity Measure. Although a repair that alters the original TBox $\mathcal{T}$ as little as possible may still not be the correct repair for the user, it is nevertheless desirable to obtain a repair that does not differ significantly from $\mathcal{T}$. The goal of this feature is to provide the user with additional information that helps them compare the effect of their decision (whether to

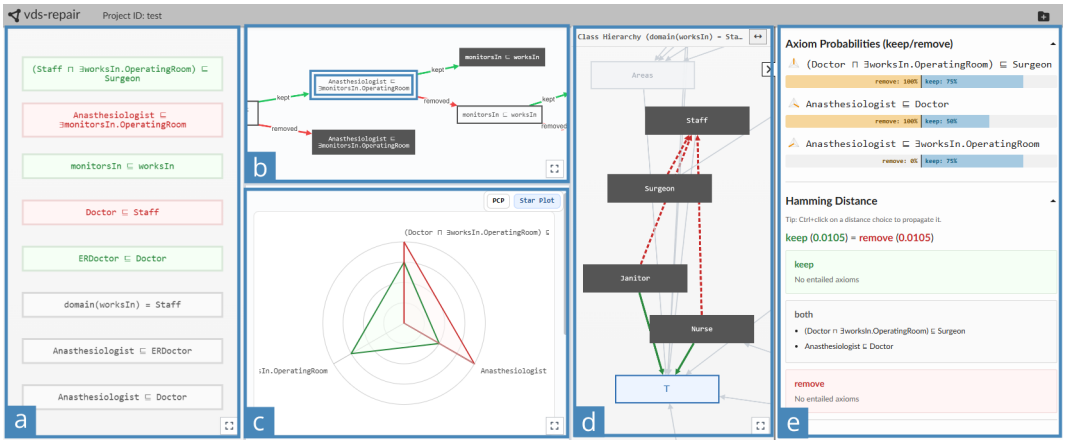

Figure 1 Screenshot of our prototype for visual decision support in interactive repair. The *process view* (a) shows the axioms for which the user must make a decision (green = **kept**, red = **removed**, gray = **undecided**). The *decision tree* (b) explicitly shows the possible repair paths depending on the binary decisions for each axiom. The *starplot* (c) compares the preservation rates of important entailments. The *diff view* (d) show the effect of the user's decision on the class hierarchy, where **removed relations** are denoted in red and **new ones** in green. Additionally, the interactive panel (e) shows, for each alternative—retaining or removing the current axiom—the dissimilarity of the best reachable repair to the current state, measured using the Hamming distance. The decisions leading to each repair can be propagated to the other views.

keep or remove α) based on the best reachable repairs in each case, where the best repairs are those most similar to $\mathcal{T}$. This is achieved using the Hamming distance, which quantifies the dissimilarity between TBoxes based on their axioms. We denote by $\mathcal{T}_c$ the current TBox obtained by applying the user's decisions so far to $\mathcal{T}$. The user is provided with two distance values: one between the current state TBox $\mathcal{T}_c$ and one of the best repairs reachable when α is retained, and the other between $\mathcal{T}_c$ and one of the best repairs reachable when α is removed. Note that there may be multiple reachable repairs in each case that remove the same minimal number of axioms from $\mathcal{T}$. Therefore, we randomly select one of them as a reference for computing the distance. We lift the similarity measure between concepts defined in [9, 17] to the level of axioms, since our repair approach considers only the removal of axioms rather than their modification. The Hamming distance between two TBoxes $\mathcal{T}_c$ and $\mathcal{T}'$ is defined based on this similarity measure as: $\text{hamming}(\mathcal{T}_c, \mathcal{T}') = 1 - \frac{|\mathcal{T}_c \cap \mathcal{T}'|}{|\mathcal{T}_c \cup \mathcal{T}'|}$, where the distance ranges between 0 (no changes at all) to 1 (no similarities at all).

Lastly, regardless of which combination of decision-support features the user chooses, they are always notified whenever retaining α prevents a repair from being reached.

3 Conclusion and Future Work

We present a novel interactive approach to ontology repair that supports users in making decisions under uncertainty through dedicated decision-support features—entailment preservation rate, hierarchy difference, and dissimilarity measure—implemented in both a command-line tool and a web-based prototype. As future work, we plan to evaluate the approach experimentally and through a user study, and to further develop and integrate it into our tool Evonne [18].

References

1 Christian Alrabbaa, Franz Baader, Raimund Dachselt, Tamara Flemisch, and Patrick Koopmann. Visualising proofs and the modular structure of ontologies to support ontology repair. In *Proceedings of the 33rd International Workshop on Description Logics (DL 2020) co-located with the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020), Online Event [Rhodes, Greece], September 12th to 14th, 2020*, CEUR Workshop Proceedings. CEUR-WS.org, 2020. URL: https://ceur-ws.org/Vol-2663/paper-2.pdf.

2 Franz Baader, Bernhard Ganter, Baris Sertkaya, and Ulrike Sattler. Completing description logic knowledge bases using formal concept analysis. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 230–235, 2007. URL: http://ijcai.org/Proceedings/07/Papers/035.pdf.

3 Franz Baader, Patrick Koopmann, and Francesco Kriegel. Optimal repairs in the description logic $\mathcal{EL}$ revisited. In *Logics in Artificial Intelligence - 18th European Conference, JELIA 2023, Dresden, Germany, September 20-22, 2023, Proceedings*, Lecture Notes in Computer Science, pages 11–34. Springer, 2023. doi:10.1007/978-3-031-43619-2_2.

4 Franz Baader and Francesco Kriegel. Pushing optimal ABox repair from $\mathcal{EL}$ towards more expressive horn-dls. In *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022, Haifa, Israel, July 31 - August 5, 2022*, 2022. doi:10.24963/kr.2022/3.

5 Franz Baader, Francesco Kriegel, Adrian Nuradiansyah, and Rafael Peñaloza. 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*, pages 319–328. AAAI Press, 2018. URL: https://aaai.org/papers/36-18056-making-repairs-in-description-logics-more-gentle/.

6 Franz Baader, Rafael Peñaloza, and Boontawee Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}^+$. In *KI 2007: Advances in Artificial Intelligence, 30th Annual German Conference on AI, KI 2007, Proceedings*, Lecture Notes in Computer Science, pages 52–67. Springer, 2007. doi:10.1007/978-3-540-74565-5_7.

7 Franz Baader and Baris Sertkaya. Usability issues in description logic knowledge base completion. In *Formal Concept Analysis, 7th International Conference, ICFCA 2009, Proceedings*, Lecture Notes in Computer Science, pages 1–21. Springer, 2009. doi:10.1007/978-3-642-01815-2_1.

8 Franz Baader and Boontawee Suntisrivaraporn. Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In *Proceedings of the Third International Conference on Knowledge Representation in Medicine, Phoenix, Arizona, USA, May 31st - June 2nd, 2008*, CEUR Workshop Proceedings. CEUR-WS.org, 2008. URL: https://ceur-ws.org/Vol-410/Paper01.pdf.

9 Jérôme David and Jérôme Euzenat. Comparison between ontology distances (preliminary results). 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, pages 245–260. Springer, 2008. doi:10.1007/978-3-540-88564-1_16.

10 Matthew Horridge, Bijan Parsia, and Ulrike Sattler. 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, pages 323–338. Springer, 2008. doi:10.1007/978-3-540-88564-1_21.

11 Aditya Kalyanpur, Bijan Parsia, Matthew Horridge, and Evren Sirin. 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*, Lecture Notes in Computer Science, pages 267–280. Springer, 2007. doi:10.1007/978-3-540-76298-0_20.

12 Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and Bernardo Cuenca Grau. Repairing unsatisfiable concepts in OWL ontologies. In York Sure and John Domingue, editors, *The Semantic*

Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11-14, 2006, Proceedings, Lecture Notes in Computer Science, pages 170–184. Springer, 2006. doi:10.1007/11762256_15.

13 Yevgeny Kazakov, Markus Krötzsch, and Frantisek Simancík. The incredible ELK - from polynomial procedures to efficient reasoning with $\mathcal{EL}$ Ontologies. *J. Autom. Reason.*, 53(1):1–61, 2014. doi:10.1007/S10817-013-9296-3.

14 Francesco Kriegel. Beyond optimal: Interactive identification of better-than-optimal repairs. In *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing, SAC 2025*, pages 1019–1026. ACM, 2025. doi:10.1145/3672608.3707750.

15 Joey Sik Chun Lam, Derek H. Sleeman, Jeff Z. Pan, and Wamberto Weber Vasconcelos. A fine-grained approach to resolving unsatisfiable ontologies. *Journal on Data Semantics*, 10:62–95, 2008. doi:10.1007/978-3-540-77688-8_3.

16 Ying Li and Patrick Lambrix. A system for repairing $\mathcal{EL}$ ontologies using weakening and completing. In *The Semantic Web: ESWC 2023 Satellite Events - Hersonissos, Crete, Greece, May 28 - June 1, 2023, Proceedings*, Lecture Notes in Computer Science, pages 101–105. Springer, 2023. doi:10.1007/978-3-031-43458-7_19.

17 Alexander Maedche and Steffen Staab. Measuring similarity between ontologies. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 251–263. Springer, 2002.

18 Julián Méndez, Christian Alrabbaa, Patrick Koopmann, Ricardo Langner, Franz Baader, and Raimund Dachselt. Evonne: A visual tool for explaining reasoning with OWL ontologies and supporting interactive debugging. *Comput. Graph. Forum*, 42(6), 2023. doi:10.1111/CGF.14730.

19 Nadeschda Nikitina, Sebastian Rudolph, and Birte Glimm. Interactive ontology revision. *J. Web Semant.*, 12:118–130, 2012. doi:10.1016/J.WEBSEM.2011.12.002.

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

21 Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 355–362. Morgan Kaufmann, 2003. URL: http://ijcai.org/Proceedings/03/Papers/053.pdf.

22 Stefan Schlobach, Zhisheng Huang, Ronald Cornet, and Frank van Harmelen. Debugging incoherent terminologies. *J. Autom. Reason.*, 39(3):317–349, 2007. doi:10.1007/S10817-007-9076-Z.

23 Kostyantyn Shchekotykhin, Gerhard Friedrich, Philipp Fleiss, and Patrick Rodler. Interactive ontology debugging: Two query strategies for efficient fault localization. *Journal of Web Semantics*, 12:88–103, 2012. doi:10.1016/j.websem.2011.12.006.

24 Boontawee Suntisrivaraporn, Franz Baader, Stefan Schulz, and Kent A. Spackman. Replacing SEP-triplets in SNOMED CT using tractable description logic operators. In *Artificial Intelligence in Medicine, 11th Conference on Artificial Intelligence in Medicine, AIME 2007, Amsterdam, The Netherlands, July 7-11, 2007, Proceedings*, Lecture Notes in Computer Science, pages 287–291. Springer, 2007. doi:10.1007/978-3-540-73599-1_38.

25 Tommaso Zendron and Rafael Peñaloza. Guiding interactive ontology repair through prolific and relevant axioms. In *Proceedings of the Workshop on Foundations and Future of Change in Artificial Intelligence (FCAI 2025)*, volume 4069, pages 8–26. CEUR-WS, 2025. URL: https://ceur-ws.org/Vol-4069/paper2.pdf.