Union and Intersection of all Justifications (Extended Abstract) ⋆

Jieying Chen[1], Yue Ma[2], Rafael Peñaloza[3], and Hui Yang[2]

[1] SIRIUS, Department of Information, University of Oslo, Norway
`jieyingc@ifi.uio.no`
[2] LISN, Univ. Paris-Sud, CNRS, Université Paris-Saclay, Orsay, France
`{ma, yang}@lri.fr`
[3] IKR3 Lab, University of Milano-Bicocca, Milan, Italy
`rafael.penaloza@unimib.it`

As ontologies grow in size and complexity, it becomes ever more important to understand the underlying causes for a consequence followed from an ontology, specially if this consequence is unexpected or unwanted for the representation domain. A usual approach in description logics (DL) is to compute one or all *justifications*; that is, the minimal subontologies that still entail the consequence under consideration. Several approaches have been developed for computing these justifications. Black-box approaches—which repeatedly call an unmodified reasoner—dominate the scene for more expressive DLs, while glass-box approaches modifying the underlying behaviour of the reasoner have been implemented mainly for lightweight DLs only [12].

Recent work has highlighted the importance of approximating the set of all justifications through two sets: the intersection and union of all justifications. The intersection (also known as the *core*) provides a lower approximation, with axioms that are *necessary* for the entailment. In fact, removing any axiom from the core automatically removes the consequence. The union, on the other hand, yields an upper approximation containing all *relevant* axioms. That is, as long as one is only interested in the consequence at hand, one can safely ignore all the axioms that do not belong to this union. In other words, this union is the smallest possible justification-preserving module.

There is, to date, a lack of approaches for computing these upper and lower approximations, specially in the case of the union of justifications, which has an underlying computationally-hard task. We fill this gap by presenting new methods for computing the core and the union of justifications.

Core. The algorithm for computing the core follows a simple black-box approach based on the notion of necessity. In essence, an axiom α is necessary (and hence belongs to the core) if and only if removing it from the ontology means getting rid of the consequence. Our algorithm simply checks necessity of α by calling a reasoner with the ontology minus $\{\alpha\}$: if the consequence does not

⋆ Jieying Chen is supported by the SIRIUS centre, which is funded by the Norwegian Research Council, project number 237898 and is co-funded by its partner companies. Copyright © 2021 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

hold anymore, the α is added to the core. This approach requires one entailment test for each axiom, and thus incurs in a linear overhead over standard reasoning. To avoid unnecessary checks, we run the algorithm over a justification-preserving module [3, 4, 9, 13].

Union of all Justifications. We introduce two algorithms for computing the union of all justifications. The first one is a black-box approach inspired by Reiter's Hitting Set Tree algorithm [14], following the ideas similar to those presented in [8, 16]. Due to its black-box nature, it can be applied to ontologies with any expressivity, as long as a reasoner is available. The idea is to collect the union of all justifications while computing all justifications and prune the search space when all remaining justifications are fully contained in the union computed. Additionally, we use the pre-computed intersection of all justifications to reduce the search space. The algorithm runs in exponential time in size of the given ontology in the worst case.

The second algorithm reduces the problem of computing the union of all justifications to a related problem over propositional formulas, through three main steps. First, we compute a CNF formula ϕ using the consequence-based reasoner *condor* [2]. Then, we check for each clause if it belongs to a Minimal Unsatisfiable Subsets (MUS) of the formula or not using the SAT-tool *cmMUS* [7]. Finally, we extract the union of all justifications from the original axioms corresponding to clauses that are members of MUSes. Contrary to the black-box approach, this method requires the ontology to be in a language which *condor* can handle. In our current implementation, the *condor* can accept $\mathcal{ALC}$-TBoxes.

Repair. A repair of an ontology is a maximal sub-ontology that does not preserve the consequence. We propose a new repair notion called *optimal repair*. Generally speaking, an *optimal repair* is a repair that removes the least amount of axioms from the original ontology. We say $\mathcal{S}$ is the smallest minimal hitting set if $|\mathcal{S}|$ is the smallest among all minimal hitting sets. The following proposition shows how we can compute the set of all optimal repairs through a hitting set computation [1, 10, 15].

Proposition 1. *Let* $\mathrm{Just}(\mathcal{O}, \alpha)$ *be the set of all justifications for the GCI* α *w.r.t. the ontology* $\mathcal{O}$*. If* $\mathbb{S}$ *is the set of all smallest minimal hitting sets for* $\mathrm{Just}(\mathcal{O}, \alpha)$*, then* $\{\mathcal{O} \setminus \mathcal{S} \mid \mathcal{S} \in \mathbb{S}\}$ *is the set of all optimal repairs for* $\mathcal{O} \models \alpha$*.*

Specially, if the core is not empty, a smallest minimal hitting set for all justifications is a singleton set that contains only one axiom from the core.

Evaluation. We built a prototypical implementation to evaluate the performance of our algorithms in real-world ontologies. The black-box algorithm uses the OWL API [6] to access ontologies and HermiT [5] as a standard reasoner. The MUS-membership algorithm (MUS-MEM) calls cmMUS [7] to detect whether a clause is a member of MUSes. We computed a single justification ($\mathcal{J}$), the core ($\mathcal{C}$), and the union of justifications for all atomic subsumptions entailed by 95 ontologies from the 2014 ORE competition [11].

	$\mathcal{J}$	$\mathcal{C}$
min	0.001s	0.001s
max	226.608s	341.560s
mean	0.400s	0.456s
median	0.009s	0.002s

Table 1: Comp. time of $\mathcal{J}$ vs $\mathcal{C}$

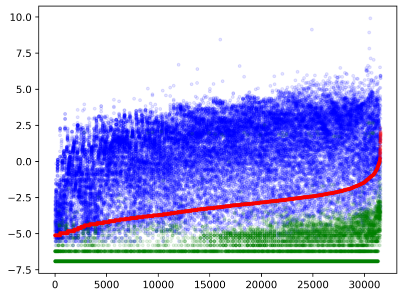

Fig. 1: Comp. time of the union

Note that in terms of computational complexity computing the core and one justification are equally hard problems. But the black-box method for the latter should intuitively be faster than the former as it reduces the size of the ontology throughout the execution. Indeed, the latter method removes all unnecessary axioms, while the former always calls the reasoner with the original ontology minus one axiom. Our experiments tend to confirm this view, as shown in Table 1.

To evaluate the computation of the union of justifications, we compared against the canonical approach of finding all justifications and our approach on computing the union. Figure 1 plots the logarithmic computation time of the union (in the vertical axis) of each test instance (in the horizontal axis). Each blue, green or red dot corresponds to the computation time of the union by OWL API, the black-box algorithm or the MUS-MEM algorithm for a conclusion respectively. In our experiments, our black-box approach clearly outperformed the MUS-based translation and the enumeration of justifications.

In the end, we also analysed the proportional sizes of cores, justifications, and unions of justifications. Interestingly, 85% of the test instances have only one justification, and even among the others, 84% of cores are non-empty, which means that we could be able to use core to compute the set of all optimal repairs for most cases. In general, the size of the union tends not to be much larger than that of the core.

Conclusion In this paper, we presented algorithms for computing the core and the union of all justifications for a given DL consequence. As an application, we study how to compute optimal repairs effectively, through the information provided by the core and the union of all justifications.

In the future, we plan to further detailed analyse the experimental results and provide better insights to explain the results that we have shown in the evaluation. Currently, the MUS-based approach for computing the union of all justifications only supports $\mathcal{ALC}$ ontologies. However, we would expect that this approach could be further generalised to more expressive languages.

References

1. F. Baader and R. Peñaloza. Axiom pinpointing in general tableaux. *J. Log. Comput.*, 20(1):5–34, 2010.
2. A. Bate, B. Motik, B. C. Grau, D. T. Cucala, F. Simančík, and I. Horrocks. Consequence-based reasoning for description logics with disjunctions and number restrictions. *J. Artif. Int. Res.*, 63(1):625–690, Sept. 2018.
3. J. Chen, M. Ludwig, Y. Ma, and D. Walther. Zooming in on ontologies: Minimal modules and best excerpts. In *Proc. of ISWC'17, Part I*, volume 10587 of *Lecture Notes in Computer Science*, pages 173–189. Springer, 2017.
4. J. Chen, M. Ludwig, and D. Walther. Computing minimal subsumption modules of ontologies. In *Proc. of GCAI'18*, pages 41–53, 2018.
5. B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang. HermiT: an OWL 2 reasoner. *Journal of Automated Reasoning*, 53(3):245–269, 2014.
6. M. Horridge and S. Bechhofer. The OWL API: A Java API for OWL ontologies. *Semantic Web*, 2(1):11–21, 2011.
7. M. Janota and J. Marques-Silva. cmMUS: A tool for circumscription-based mus membership testing. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 266–271. Springer, 2011.
8. A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of OWL DL entailments. In *Proceedings of ISWC 2007 & ASWC 2007*, volume 4825 of *LNCS*, pages 267–280. Springer, 2007.
9. P. Koopmann and J. Chen. Deductive module extraction for expressive description logics. In C. Bessiere, editor, *Proceedings of IJCAI'20*, pages 1636–1643. ijcai.org, 2020.
10. M. H. Liffiton and K. A. Sakallah. On finding all minimally unsatisfiable subformulas. In F. Bacchus and T. Walsh, editors, *Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT 2005)*, volume 3569 of *Lecture Notes in Computer Science*, pages 173–186. Springer, 2005.
11. B. Parsia, N. Matentzoglu, R. S. Gonçalves, B. Glimm, and A. Steigmiller. The OWL reasoner evaluation (ORE) 2015 competition report. *Journal of Automated Reasoning*, pages 1–28, 2015.
12. R. Peñaloza. Axiom pinpointing. In G. Cota, M. Daquino, and G. L. Pozzato, editors, *Applications and Practices in Ontology Design, Extraction, and Reasoning*, volume 49 of *Studies on the Semantic Web*, pages 162–177. IOS Press, 2020.
13. R. Peñaloza, C. Mencía, A. Ignatiev, and J. Marques-Silva. Lean kernels in description logics. In E. Blomqvist, D. Maynard, A. Gangemi, R. Hoekstra, P. Hitzler, and O. Hartig, editors, *Proceeding of ESWC'17*, volume 10249 of *Lecture Notes in Computer Science*, pages 518–533, 2017.
14. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.
15. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In G. Gottlob and T. Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 355–362. Morgan Kaufmann, 2003.
16. B. Suntisrivaraporn, G. Qi, Q. Ji, and P. Haase. A modularization-based approach to finding all justifications for OWL DL entailments. In J. Domingue and C. Anutariya, editors, *The Semantic Web*, pages 1–15, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.