Guiding Interactive Ontology Repair through Prolific and Relevant Axioms

Tommaso Zendron[1], Rafael Peñaloza[2]

[1]Vrije Universiteit Amsterdam, Amsterdam, Netherlands
[2]University of Milano-Bicocca, Milan, Italy

Abstract

Knowledge engineering is an error-prone task and the need for automated tools to help correct existing errors has been long recognised. While many ontology repair methods have been proposed, most ignore a fundamental issue: only a domain expert can state whether an ontology reflects the domain knowledge or not. In this paper, we consider an interactive ontology repair procedure where a domain expert guides the system by signalling axioms that are identified as erroneous from sets of suspicious axioms. To avoid overwhelming the human expert with the number and complexity of information requests, we propose methods to reduce the class of potentially suspicious axioms: first, the expert may signal a class of consequences they desire to preserve in the corrected ontology; second, we introduce a new notion of prolificacy to understand the impact axioms have in the derivation of consequences.

Keywords

ontology repair, description logics, iterative debugging, user interaction, EL

1. Introduction

Description logics (DLs) [1] are a family of knowledge representation formalisms which provide a clear and simple syntax and formal semantics based on first-order logic. One of the greatest successes is the use of the light-weight DLs from the $\mathcal{EL}$ family [2] as the basis for representing large ontologies, specially within the life-sciences fields, like the medical ontology Snomed [3], which uses over half a million $\mathcal{EL}$ axioms.

There are many reasons why knowledge engineering is an error-prone task: experts may disagree on some information, details may be lost in the translation from human expertise to logical formalism, or simply syntactic misunderstandings and typos may occur. The fact is that it is not uncommon to face errors in an ontology. For instance, the Snomed knowledge base was found to entail that every *amputation of finger* is also an *amputation of hand*, which is obviously unwanted in the modelled domain. Managing these errors, specially in large ontologies, requires the help of specialised automated tools.

Throughout the last two decades, many methods have been proposed to explain [4, 5, 6, 7] and repair [8, 9, 10, 11] ontologies, and to allow for sound reasoning in the presence of errors [12, 13, 14], where an error can be any consequence that does not mirror the application domain. These proposals are not fully adequate from a knowledge engineering point of view. Indeed, error-tolerant reasoning is a short-term solution to keep using an ontology in production after some error has been identified, but by no means it is intended to be a final solution to the existence of these errors. On the other hand, errors may have multiple, mutually dependent, causes. This in particular means that there are many possible repairs—potentially exponentially many on the size of the ontology. Deciding which one of them to choose is far from obvious.

To-date, most proposals for selecting one ontology which corrects the observed errors follow an idea akin to the minimal-change restriction used in belief revision [15]: select the ontology that is the

Workshop on the Foundations and Future of Change in Artificial Intelligence (FCAI 2025) at the 28th European Conference on Artificial Intelligence (ECAI 2025, October 25 - 30), Bologna, Italy
t.zendron@vu.nl (T. Zendron); rafael.penaloza@unimib.it (R. Peñaloza)
https://rpenalozan.github.io/ (R. Peñaloza)
0009-0009-3564-8783 (T. Zendron); 0000-0002-2693-5790 (R. Peñaloza)


"closest" to the original one, for an appropriate notion of closeness. The formal definition of a repair—a subset-maximal sub-ontology which does not entail the error—is already a step in this direction. To reduce the number of choices, it has been proposed to consider the repairs of maximum *cardinality*. On the other hand, minimal change could also be interpreted as preserving most of the original knowledge, which could also be *implicit*. Thus, rather than eliminating axioms, it has been proposed to modify them through syntactic refinement operators [16] to obtain a weaker ontology [9, 10], or to compute semantically optimal repairs [17, 18] among many others.

While all these methods are adequately motivated and provide some guarantees, they still ignore a fundamental element of error correction: *correctness* of an ontology—in the sense of its alignment to the knowledge domain it is supposed to represent—is a meta-logical property, and no amount of syntactic or semantic logical manipulation can access this "ground truth." Indeed, only domain experts can state whether an ontology correctly reflects the domain knowledge or not. Thus, to fully correct an ontology, we cannot escape asking for the expert feedback.

Since human expert access is limited and expensive, ontology repair requires a delicate balance between automation and interaction. While too much automation may lead to incorrect ontologies, human experts may at some point be unable to answer too many or too complex requests for feedback. Indeed, while in theory one could automatically compute all possible repairs and ask the expert to choose among them, the number of existing solutions makes this approach infeasible in practice. The goal is then to build an iterative procedure in which a knowledge engineer (henceforth called the *user*) is asked a series of questions designed to yield a correct repair. The idea is somehow reminiscent of iterative ontology completion methods [19, 20]. The main difference is with the kinds of questions asked. In our case, the underlying simple idea is to present the user with one justification at a time (that is, a set of axioms that produce the error, guaranteeing that at least one of them is erroneous) and ask them to identify the wrong axiom until a full diagnosis is obtained.

The goal of this paper is to devise a technique to reduce the number and size of the sets presented to the user for verification, without first computing all possible repairs. To this end, we propose two complementary techniques to reduce the search space for the "correct" repair. The first is a manual phase, where the user is allowed to provide a class of consequences which *must* be preserved. These in general refer to well-established, uncontroversial, and clear knowledge from the representation domain, which might be implicitly derivable from the ontology, even if the user does not know exactly from which ontology axioms. Given this information, the class of potentially wrong axioms is reduced, and the potential for deriving an answer that does not align with the domain is diminished.

The second phase is an automated one, which relies on a new notion of *prolificacy* of axioms. In a nutshell, prolificacy refers to how many consequences depend on the axiom in question. In the spirit of minimal change, most prolific axioms are those that preserve the most original consequences, and hence most desired in a final repair. Interestingly, if restricted to atomic consequences only, prolificacy of all axioms in ontologies from the $\mathcal{EL}$ family is computable in polynomial time. In order to decide how many, and which, prolific axioms to save, we focus on preserving the prolificacy of justifications (see Algorithm 2 for details).

These two phases combined allow us to focus on fewer axioms to present to the user, at each step and in total, and construct a better ordering strategy. We emphasise that for this paper we focus mainly on these pruning strategies, but the overarching goal is to deal with the full iterative repair process. In particular, the pruning phase can be repeated every time that the user has identified an erroneous axiom, to further decrease the number of remaining questions to be asked to them.

2. Preliminaries

We first introduce all the preliminaries needed for understanding the rest of this work. Specifically, we briefly explain the description logic $\mathcal{EL}^{\bot}$ as a representative of the $\mathcal{EL}$ family, and the notions of justifications and repairs.

2.1. The Description Logic $\mathcal{EL}^\bot$

$\mathcal{EL}^\bot$ [2] is a lightweight member from the family of description logics (DLs) [1], which is well known for its polynomial time reasoning problems. The main building blocks of $\mathcal{EL}^\bot$ are *concepts* (unary predicates from first-order logic) and *roles* (binary predicates). Its syntax and semantics are formalised next.

Consider three countable, mutually disjoint sets N_I, N_C, and N_R of *individual names*, *concept names*, and *role names*, respectively. $\mathcal{EL}^\bot$ *concepts* are built through the grammar rule $C ::= A \mid \top \mid \bot \mid C \sqcap C \mid \exists r.C$, where $A \in N_C$ and $r \in N_R$. The semantics is based on *interpretations*, which are pairs $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$, where $\Delta^\mathcal{I}$ is a non-empty (potentially infinite) set called the *domain*, and $\cdot^\mathcal{I}$ is the *interpretation function* which maps every individual name $a \in N_I$ to an object $a^\mathcal{I} \in \Delta^\mathcal{I}$; every concept name $A \in N_C$ to a set $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$; and every role name $r \in N_R$ to a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$. This function is extended to arbitrary concepts by setting: $\top^\mathcal{I} := \Delta^\mathcal{I}$; $\bot^\mathcal{I} := \emptyset$; $(C \sqcap D)^\mathcal{I} := C^\mathcal{I} \cap D^\mathcal{I}$; and $(\exists r.C)^\mathcal{I} := \{\delta \in \Delta^\mathcal{I} \mid \exists \eta \in C^\mathcal{I}.(\delta, \eta) \in r^\mathcal{I}\}$.

The goal of DLs is to represent the knowledge of a domain in a formal way which allows for automated reasoning. This is achieved by introducing restrictions on the class of interpretations of interest; namely, those which comply with what is known about the knowledge domain.

Definition 1 (ontology). *A* general concept inclusion *(GCI) is an expression of the form $C \sqsubseteq D$ where C, D are $\mathcal{EL}^\bot$ concepts. A* TBox *is a finite set of GCIs. An* assertion *is an expression of the form $C(a)$ or $r(a, b)$ where C is a concept, $r \in N_R$, and $a, b \in N_I$. An* ABox *is a finite set of assertions. An* ontology *is a pair $\mathcal{O} = (\mathcal{A}, \mathcal{T})$ consisting of an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$. For brevity, we will often use the term* axiom *to refer generally to GCIs and assertions alike.*

The interpretation $\mathcal{I}$ satisfies *the GCI $C \sqsubseteq D$ iff $C^\mathcal{I} \subseteq D^\mathcal{I}$. It* satisfies *the assertion $A(a)$ iff $a^\mathcal{I} \in A^\mathcal{I}$ and the assertion $r(a, b)$ iff $(a^\mathcal{I}, b^\mathcal{I}) \in r^\mathcal{I}$. $\mathcal{I}$ is a* model *of the ABox, TBox, or ontology $\mathcal{K}$ (represented as $\mathcal{I} \models \mathcal{K}$) iff it satisfies all the axioms in $\mathcal{K}$.*

In general, we say that an ontology $\mathcal{O}$ is *consistent* iff it has a model, and it *entails* an axiom α (written $\mathcal{O} \models \alpha$) iff every model of $\mathcal{O}$ is also a model of $\{\alpha\}$. It is well known that consistency and entailments for $\mathcal{EL}^\bot$ ontologies can be decided in polynomial time through consequence-based algorithms which make implicit consequences explicit [2, 21].

When dealing with $\mathcal{EL}^\bot$ it is often useful to assume that the ontology axioms are limited to a kind of normal form. We say that a TBox is in *normal form* if all GCIs are of one of the forms

$$A \sqsubseteq B, \quad A_1 \sqcap A_2 \sqsubseteq B, \quad A \sqsubseteq \exists r.B, \quad \exists r.A \sqsubseteq B,$$

where $A, A_1, A_2 \in N_C \cup \{\top\}$ and $B \in N_C \cup \{\top, \bot\}$. An ABox is in *normal form* if concept assertions are atomic (i.e., refer to concept names only). An ontology is in *normal form* if its TBox and ABox are both in normal form. It is easy to see that any ontology can be transformed in polynomial time into another one that preserves the same consequences over the original vocabulary, but is in normal form [2]. In a nutshell, a complex concept C can be substituted with new fresh concept name X, which is then required to be interpreted as C by adding adequate GCIs to the TBox. Similarly, a complex assertion $C(a)$ can be substituted by $X(a)$ with X a fresh concept name, by adding $X \sqsubseteq C$ to the TBox.

Ontologies written in $\mathcal{EL}^\bot$ and other variants from the $\mathcal{EL}$ family have been successfully used to represent various application domains, particularly from the biomedical fields. The complexity of these domains, and the simplicity of the language (specially if axioms are presented in normal form), means that real-world ontologies tend to have tens of thousands—or hundreds of thousands—axioms, making the process of constructing them error-prone, and corrections very difficult. Hence techniques for helping understand and correct unwanted consequences have been developed.

2.2. Justifications and Repairs

Finding the axioms responsible for an error in an ontology among several thousands is infeasible without the help of some automated deduction tool. In monotonic languages (like all classical DLs

including $\mathcal{EL}^{\bot}$), this corresponds to a task known as *axiom pinpointing*; that is, identifying one or all minimal sub-ontologies that entail the consequence—also known as justifications [22].

Definition 2 (justification). *Let $\mathcal{O}$ be an ontology and α an axiom such that $\mathcal{O} \models \alpha$. An ontology $\mathcal{M}$ is called a* justification *for α w.r.t. $\mathcal{O}$ iff (i) $\mathcal{M} \subseteq \mathcal{O}$; (ii) $\mathcal{M} \models \alpha$; and (iii) for all $\mathcal{M}' \subsetneq \mathcal{M}$, $\mathcal{M}' \not\models \alpha$.*

Note that this definition includes justifications for ontology inconsistency, by considering the axiom $\top \sqsubseteq \bot$ as the consequence α.

Justifications can be seen as means to logically "explain" the consequence α by retaining only the necessary information for deriving it and removing all superfluous knowledge. Yet, justifications are not unique; in fact, a consequence may have exponentially many justifications w.r.t. a single ontology $\mathcal{O}$ (on the size of $\mathcal{O}$) [8].

If the goal is to get rid of an unwanted consequence, it makes sense to rather consider the notion of a repair; that is, maximal sub-ontologies which do not entail this consequence [8].

Definition 3 (repair). *Let $\mathcal{O}$ be an ontology and α an axiom such that $\mathcal{O} \models \alpha$. An ontology $\mathcal{R}$ is a* repair *for α w.r.t. $\mathcal{O}$ iff (i) $\mathcal{R} \subseteq \mathcal{O}$; (ii) $\mathcal{R} \not\models \alpha$; and (iii) for all $\mathcal{R} \subsetneq \mathcal{R}' \subseteq \mathcal{O}$, $\mathcal{R}' \models \alpha$.*

Justifications and repairs are dual not only from a definitorial point of view, but in fact they can be computed from each other. Indeed, if $\mathfrak{J}$ is the class of all justifications, then any minimal hitting set of $\mathfrak{J}$ is the complement of a repair (also known as a *diagnosis*). Dually, the minimal hitting sets of the class of the diagnoses form all the justifications. Moreover, the number of diagnoses and repairs can also be exponential on the size of the ontology. All these are well-known classical results; see e.g. [23, 24]. Intuitively, to guarantee that a consequence cannot be derived any more from an ontology, one must make sure that no justification is contained in it. This is equivalent to removing at least one axiom from each justification, a process simulated by the computation of the hitting sets and complementation.

Methods for computing justifications and repairs have been studied over the years, along with techniques for reasoning in the presence of errors or inconsistencies. However, from the point of view of an ontology life-cycle, we cannot be content with using an ontology which is known to contain errors. Rather, we want to correct the ontology for future use. In the next section, we provide a brief overview of existing ontology repair methods.

3. Ontology Repair

From the point of view of knowledge engineering, we are not interested in one arbitrary repair or in all repairs to correct an ontology. We are rather interested in the "right" one, which actually fixes the issues.

The task of finding an adequate repair is similar to *contraction* from belief revision [15]. In this setting, the goal is to modify an ontology so that a consequence is not entailed any more, while satisfying some rationality postulates and typically also a *minimal change* condition: the contraction should be as close to the original ontology as possible. Different solutions have been proposed depending on the interpretation given to the notion of "closeness."

Every repair, as in Definition 3, can be considered to be minimally different from the original ontology, as no further axioms can be re-introduced without reintroducing the error. It has also been proposed to consider only repairs of maximal cardinality, as they minimize the *number* of removed axioms. Yet, these are not only hard to compute [23], but also suffer from the same limitations as general repairs: there can be exponentially many of them, and may still remove too much (desired) knowledge.

A different perspective comes from trying to extend the class of models of the ontology, without necessarily preserving its full syntactic structure. From this point of view, one idea is to *weaken* (i.e., make more general or less restrictive) the axioms in a diagnosis through syntactic refinement operators. These *gentle repairs* preserve more of the original consequences, and do not damage too much the original structure, by substituting an axiom by one of its maximally strong weakenings. On the other

hand, so-called *optimal repairs* aim to semantically preserve as many of the original consequences as possible, with less regard towards the ontology syntax [14, 17].

The main limitation of all these approaches is that they are purely logical, and hence cannot really take into account the domain knowledge to generate the ontology which best coincides with the *intent* of the modeller. Indeed, correctness of an ontology is a meta-logical property, and logic alone cannot distinguish which of all the possible repairs available best describes the domain knowledge. To find the "true" repair, the only solution is to ask domain experts which are able to identify erroneous constraints. However, the sheer number of options available makes it impossible for any human expert to select among them. Thus, we propose an iterative approach which relies on user-input to identify the best possible ontology repair. The main difficulties and proposed solutions are described in the next section.

We recall that the final goal is to *repair* an ontology; yet, as repairs are almost as big as the original ontologies (with up to hundreds of thousands of axioms), presenting a user with the choice between two repairs (or more) is not helpful. Rather, we take advantage of the duality between justifications and repairs and focus on the computation of the former—which are typically of a much smaller size—and the diagnoses that arise from them. A naïve idea is to enumerate all the justifications, one at a time, and ask the user to identify one axiom to be removed, until at least one axiom from each justification has been deleted. This idea is not only impractical given the potentially exponential number of justifications available, but might also lead to the deletion of too many axioms if the user was unable to identify that a different choice would "touch" other yet-unseen justifications. Indeed, by studying one justification at a time, the user cannot grasp the full picture of the repair process, which forces them to think about individual axioms instead of their global consequences. Note that despite being a domain expert, the user is not *omniscient* towards all the consequences that are affected by the removal of an axiom.

4. User-guided Ontology Repair

We present in detail our novel approach for interactively repairing ontologies. The basic idea is very simple: ask a sequence of questions from the knowledge engineer which will allow to identify the *correct* repair among the large number of candidates. However, as human experts are scarce, costly, and tire easily, we should try to minimise the number of questions and their difficulty, thus maximising the investment on human work. Moreover, the order in which the questions are asked may influence the quality of the final result.

The first step towards an adequate user-guided ontology repair is to prune the search space; i.e., reduce the amount of answers required by the user. We introduce a two-phase procedure. In the first phase (*interactive pruning*), the user specifies some consequences and axioms which should be preserved in the repair. In the second phase (*automated pruning*) the system identifies and protects structurally central axioms from the ontology. After pruning, the approach goes into a *diagnosis computation phase*, which can be performed either manually, by interacting with the user, or fully automatically, letting the system using predefined strategies to select the most appropriate diagnosis according to the desired repair goals. Yet, an important element of our approach is that we do not want to compute the whole set of justifications or repairs either *a-priori* or during the interaction. Instead, we try to minimise this computational effort as well.

To simplify the procedure and its exposition, we assume that the ontology is in normal form (see Section 2). If this is not the case, we can always first pre-process the ontology through a normalisation method like the ones suggested in [1, 25]. We note that most ideas work, albeit in a much more technical manner, with general ontologies.

It can be seen that conjuncts appearing in a normalised ontology are necessarily syntactically disjoint; i.e., they refer to different syntactic (concept) names even if they may have implicit semantic connections derivable from the ontology. This syntactic disjointness and the simplicity of the normal form allow for more fine grained repairs, without removing more information than what is needed. This property is guaranteed in gentle repairs, since the maximally strong weakening would preserve as much information as possible without any need for preprocessing. Nevertheless, the preprocessing is useful in general,

as it allows to present to the user smaller axioms and to avoid the problem of having justifications consisting of very few, but very complex axioms. Normalisation renders axioms and justifications more comprehensible for the user, who can manage them more efficiently, leading to an improvement in the user experience.

4.1. Pruning the Search Space

As we have seen, the naïve approach to repairing the ontology may lead to a deletion of too many axioms. The main issue is that presenting one justification or one axiom at a time to the user may not let them understand the implications of its removal. Thus, our algorithm should select a subset of relevant questions from all the possible ones, structuring them in a convenient way to let the user express their preferences obtaining the repair that better fits their needs.

First note that only axioms which appear in at least one justification will ever need to be removed to obtain a repair. Hence we start with the set $\mathcal{U}$ defined as the union of all justifications. Although it is expensive to compute this set, we can do that process once, in the background, before any interaction with the user. The next step is to identify which axioms are *more important* than others for reasoning with the ontology. Once these important axioms are identified, we can prevent them from being excluded from the repair, by removing them from the set $\mathcal{U}$. This effectively ensures that the repair, whatever it will be, will maintain such axioms and their consequences. We call this technique of identifying important axioms that should be retained in the final repair *justification pruning*. As we will see, this pruning allows to overcome many of the limitations of the naïve approach. However, we still have not formally defined what does it mean for an axiom to be important. As we explain next, in the context of interactive repair, it is useful to distinguish between two complementary notions of axiom importance, which guide the reduction of the repair search space: relevance and prolificacy.

We call an axiom *relevant* when it is identified as such by the user. This could be either because the user explicitly stated its importance for the ontology; because it is frequently used in deployed reasoning tasks; or because it is required to derive other specific consequences that the user wishes to preserve. The importance of relevant axioms is thus subjective and driven by the user's intent. In the context of our strategy, we identify relevant axioms as those that appear in at least one justification for a user-specified desired consequence. These axioms are considered important precisely because they enable the derivation of such consequence.

On the other hand, the notion of *prolificacy* assesses the objective importance of an axiom in the knowledge base for deducing consequences. Informally, we say that an axiom α_1 is more *prolific* than an axiom α_2 if the number of consequences for which α_1 is necessary is greater than the number for which α_2 is necessary. In this case, we say that α_1's prolificacy is greater than α_2's. In our framework, most prolific axioms are automatically identified and protected during the automated pruning phase; see Section 4.3 for details.

By combining both perspectives, the repair process benefits from a dual safeguard: relevant axioms, computed via desired consequences, ensure the preservation of the consequences relevant to the user; whereas the identification of most prolific axioms avoids removing semantically meaningful knowledge when less expensive alternatives are available.

Before going into details, note that in general it seems reasonable to assume that prolific axioms (from which many consequences can be derived) are also relevant; that is, are also subjectively important to the user. However, this correlation depends on how the ontology is used and on the user's specific goals. If this overlap were guaranteed, there would be no need for an interactive approach, and it would suffice to compute the repair that maximises prolificacy. Since we need to understand what are the specific preferences of the user, we implement two distinct strategies for identifying important axioms, corresponding to two phases of justification pruning: interactive pruning, which preserves axioms based on user specified desired consequences and feedback; and automated pruning, which protects most prolific axioms in a justification according to objective metrics.

To order these strategies, we assume that the user preferences are more important than the objective

entailments of single axioms, as they represent the *ground truth* for the domain knowledge. Thus, the starting point is to ask the user for desired consequences. In doing so, we establish that axiom relevance is more important than prolificacy. In particular this means that, whenever it is not possible to preserve relevance while maximising prolificacy, priority will be given to the former, as exemplified next.

Example 4. *Consider the ontology $\mathcal{O}$ composed of the ABox $\mathcal{A} := \{\alpha_1 : A(a),\ \alpha_5 : r(a,b)\}$ and the TBox $\mathcal{T} := \{\alpha_2 : A \sqsubseteq B,\ \alpha_3 : B \sqsubseteq \bot,\ \alpha_4 : \exists r.\top \sqsubseteq \bot\}$, where the prefixes α_i are just names given to the axioms for brevity of presentation. It is easy to see that $\mathcal{O}$ is inconsistent, and that there are two justifications for inconsistency; namely, $\mathcal{J}_1 := \{\alpha_1, \alpha_2, \alpha_3\}$ and $\mathcal{J}_2 := \{\alpha_4, \alpha_5\}$; this means that there are six diagnoses and repairs for this inconsistency.*

Suppose now that the user expresses that $B \sqcap \exists r.\top(a)$ is a desired consequence, which should be preserved. Since the only justification of this consequence (i.e., the only necessary axioms for deriving it) is $\{\alpha_1, \alpha_2, \alpha_5\}$, these three axioms are marked as relevant*, and must be protected through the repair process.*

At this point, the system moves to the automated pruning phase. Consider that the system identifies α_4 as the most prolific axiom. Protecting it from removal would result in preserving the full justification $\mathcal{J}_2$ (as α_5 is already protected) and hence the inability to repair $\mathcal{O}$, or on removing α_5, and thus the consequence requested by the user. As the user preference is prioritised, the system does not protect the axiom α_4. This illustrates the precedence of interactive pruning over automated pruning: the system should prioritise the will of the user over automatic choices. In fact, considering this precedence, the system could have avoided (in this simple example) computing the prolificacy of α_3 and α_4 and conclude that they form the only available diagnosis that satisfies the user's request.

In brief, justification pruning is a method which allows to protect some of the axioms from $\mathcal{U}$ so that they cannot appear in any computed diagnosis—and by extension, are guaranteed to appear in the final repair found. This has the advantage of reducing the number and complexity of questions to be asked to the user in the repair process, and ultimately reduce the human effort. Note that there is no pre-defined number of axioms that should be saved in the repair: this number varies depending on the specific case. However, $\mathcal{U}$ is always sufficient for computing a diagnosis after pruning.

Having introduced justification pruning, we can now explore more in detail each of the two phases. In the following, as done before, we will use the expressions "save," "preserve," and "protect" to indicate that some axioms are prevented from being included in the diagnosis. We will sometimes express the same idea by saying that these axioms are removed from the union of all justifications $\mathcal{U}$. This is motivated by the fact that diagnoses will be built as subsets of $\mathcal{U}$.

4.2. Interactive Pruning: Desired Consequences

As seen, justification pruning is needed to reduce the search space for a repair by reducing the amount of questions to submit to the user. In the interactive pruning phase, what we do is to interact with the user, asking them to freely express *desired* consequences and axioms that they want to preserve in their repair. That is, user expresses desiderata by stating knowledge that they know to be correctly reflecting the representation domain.

If the user knows that an axiom is important for their scopes, they are free to instruct the system to protect such axiom, and it will immediately be removed from the set $\mathcal{U}$ of the axioms from where a diagnosis is being built. Yet, the user may not want to point to specific axioms, but rather to more general consequences of the ontology. In this case, we propose that, after a desired consequence β is indicated by the user, the system removes one justification for β from the set $\mathcal{U}$. This effectively reduces the number of potential axioms to be verified by the user. Computing one justification for a consequence requires polynomial time, so is completely feasible in $\mathcal{EL}^\bot$ [8]. Note that this is a heuristic, which allows the computations to be performed efficiently and avoids further interactions with the user. Effectively, this heuristic guarantees that all desired consequences are preserved in the final repair, without any need for further verification throughout the process. To guarantee that a solution exists, we must assume that all such requests from the user are error-free. In practice, we expect the user to state only a few, simple, desiderata referring to the fundamental knowledge to preserve.

Algorithm 1 Modified Black-Box Algorithm for Desired Consequences

Require: Ontology $\mathcal{O}$, set $\mathcal{U}$, desired consequence β
Ensure: Set S of axioms to be protected to preserve β

```
S ← ∅
if O \ U ⊭ β then
    for each axiom α ∈ U do
        if O \ {α} ⊨ β then
            O ← O \ {α}
        else
            S ← S ∪ {α}
            if (O \ U) ∪ S ⊨ β then
                Break
            end if
        end if
    end for
end if
return S
```

Pinpointing the desired consequences is a smart strategy to restrict the search space for repairs with minimal efforts from the user, since they do not have to compare different options, but can directly indicate their preferences on consequences. Indeed, thanks to this idea of specifying desired consequences, the user is free to input whatever consequence they want (as long as it is correct). In this sense, we have addressed the problem of structuring questions. Indeed, this design is way better than presenting sets of axioms or one justification at a time as in the naïve approach, that does not allow to reason about implicit knowledge.

To compute the justification for β we can use the well-known black-box algorithm that makes a linear number of calls to a standard reasoner. The algorithm tries to remove one axiom at a time from the ontology, while β is still entailed. While irrelevant for correctness, the order in which the axioms are removed influences the justification found. Here, two conflicting forces come into place. On the one hand, we would like to reduce $\mathcal{U}$ and hence, try to keep as many axioms from $\mathcal{U}$ as possible in the justification. On the other hand, if β is entailed already by $\mathcal{O} \setminus \mathcal{U}$, then we know that β is preserved in all repairs, meaning that no special handling of β is needed. Under these conditions, we prioritise preserving more potential repairs, as this helps guarantee that the "correct" one (from the domain knowledge point of view) can still be computed. This has the added advantage of preserving as many of the original consequences as possible as well. Intuitively, we want to order the axioms so that those belonging to $\mathcal{U}$ are tested (and potentially excluded from the justification) first.

As our interest is mainly on the axioms that will be removed from $\mathcal{U}$, and not on the precise justifications of β, we modify the black-box algorithm to further reduce the number of calls to the reasoner. First, we verify whether $\mathcal{O} \setminus \mathcal{U} \models \beta$. In that case, we know that there is a justification w.r.t. $\mathcal{O} \setminus \mathcal{U}$ and no axioms need to be protected. Otherwise, we apply the standard black-box algorithm, but only verifying whether the axioms in $\mathcal{U}$ can be removed. This idea is described more formally in Algorithm 1 and exemplified next.

Example 5. *Consider the ontology $\mathcal{O}$ with an empty ABox and the TBox*

$$\begin{aligned}\mathcal{T} := \{&\alpha_1 : A \sqsubseteq D, \quad \alpha_2 : D \sqsubseteq \exists r.C, \quad \alpha_3 : A \sqsubseteq \exists r.B, \quad \alpha_4 : \top \sqsubseteq C,\\ &\alpha_5 : D \sqsubseteq E, \quad \alpha_6 : \exists s.A \sqsubseteq E, \quad \alpha_7 : A \sqsubseteq \exists s.A, \quad \alpha_8 : D \sqsubseteq X,\\ &\alpha_9 : A \sqsubseteq \exists t.D, \quad \alpha_{10} : \exists r.C \sqsubseteq F, \quad \alpha_{11} : \exists t.F \sqsubseteq X \qquad \}\end{aligned}$$

having the erroneous *consequence $\epsilon := A \sqsubseteq \exists r.C$. The error ϵ has two justifications; namely, $\mathcal{J}_1^\epsilon := \{\alpha_1, \alpha_2\}$ and $\mathcal{J}_2^\epsilon := \{\alpha_3, \alpha_4\}$. So, $\mathcal{U} := \{\alpha_1, \ldots, \alpha_4\}$.*

Suppose that $\beta := A \sqsubseteq E$ is a desired consequence provided by the user, which can be seen to have two justifications: $\mathcal{J}_1^\beta := \{\alpha_1, \alpha_5\}$ and $\mathcal{J}_2^\beta := \{\alpha_6, \alpha_7\}$.

Applying Algorithm 1, we see that $\mathcal{O} \setminus \mathcal{U} \models \beta$ because $\mathcal{J}_2^\beta \subseteq \mathcal{O} \setminus \mathcal{U}$. Hence, β is preserved in any repair, and no axiom needs to be protected. Note that without the first check, the black-box algorithm could have computed the justification $\mathcal{J}_2^\beta$, thus (unnecessarily) protecting α_1.

*Consider now the desired consequence $\gamma := A \sqsubseteq \exists s.D$ whose only justification is $\mathcal{J}^\gamma := \{\alpha_1, \alpha_7\}$. Algorithm 1 first verifies that $\mathcal{O} \setminus \mathcal{U} \not\models \gamma$ and enters the **for** loop with the axioms in $\mathcal{U}$. In the first iteration, $\mathcal{O} \setminus \{\alpha_1\} \not\models \gamma$, and hence enters the **else** in line 6, updating S to $\{\alpha_1\}$. At this point, the algorithm verifies that $(\mathcal{O} \setminus \mathcal{U}) \cup S = \{\alpha_1, \alpha_5, \ldots, \alpha_{11}\} \models \gamma$ which breaks the **for** loop and terminates the algorithm with output $\{\alpha_1\}$ as expected.*

If the user wishes to preserve multiple consequences, a simple idea would be to apply Algorithm 1 to each of them sequentially, updating the set $\mathcal{U}$ between calls to take into account the known protected axioms. Yet, as we see next, this may lead to erroneous results.

Example 6. *Continuing Example 5, suppose that the user specifies as desired consequences $\gamma := A \sqsubseteq \exists s.D$ and $\delta := A \sqsubseteq X$. As seen already, γ has only one justification $\mathcal{J}^\gamma = \{\alpha_1, \alpha_7\}$. Instead, δ has justifications: $\mathcal{J}_1^\delta := \{\alpha_1, \alpha_8\}$ and $\mathcal{J}_2^\delta := \{\alpha_2, \alpha_9, \alpha_{10}, \alpha_{11}\}$.*

At the end of Example 5 we saw that α_1 needs to be protected, so it can be removed from $\mathcal{U}$ before calling Algorithm 1 for δ. Hence, at the moment of the call, $\mathcal{U}$ contains only α_2, α_3, and α_4 and the first check verifies that $\mathcal{O} \setminus \mathcal{U} \models \delta$ and no further axioms need to be protected.

Note that if we did not remove α_1 from $\mathcal{U}$ before the call, we run the risk of computing the justification $\mathcal{J}_2^\delta$ which would require α_2 to be protected. But in that case, we would preserve a justification for the unwanted consequence ϵ and no repair could be computed.

However, note that the order in which the desired consequences are analysed influences the final result. Indeed, in the previous example, if δ had been analysed before γ, we could have protected α_2, which, as we have seen, would later lead to a wrong path. To solve this, we further extend the algorithm so that the justification is not computed for a single desired consequence, but for all simultaneously; that is, instead of considering an input β, it considers the whole set Γ of desired consequences to be entailed in each test. This guarantees that all requirements are satisfied, minimising the number of protected axioms.

Other modifications can be applied to try to improve efficiency. For instance, one may focus on justifications or diagnoses of minimal cardinality, either for protecting fewer axioms or for diminishing the potential options remaining. Alternatively, one may consider axioms that appear in the most justifications, as they allow for smaller hitting sets, and thus smaller diagnoses, as illustrated in the following example.

Example 7. *Consider the ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ formed by*

$$\mathcal{T} := \{C_i \sqsubseteq E \mid 1 \leq i \leq n\} \cup \{A \sqsubseteq E\} \cup \{A \sqcap C_i \sqsubseteq B \mid 1 \leq i \leq n\}$$
$$\mathcal{A} := \{A(a)\} \cup \{C_i(a) \mid 1 \leq i \leq n\}$$

and suppose that $B(a)$ is an erroneous consequence. The justifications for this error are

$$\{\{A(a),\ C_i(a),\ A \sqcap C_i \sqsubseteq \mathsf{B}\} \mid 1 \leq i \leq n\}.$$

Suppose now that the user specifies a desired consequence $\eta = E(a)$, which has the justifications

$$\mathcal{J}_i^\eta := \{C_i(a),\ C_i \sqsubseteq E\}, \qquad 1 \leq i \leq n;$$
$$\mathcal{J}_{n+1}^\eta := \{A(a),\ A \sqsubseteq E\}$$

Without further instructions, Algorithm 1 may compute $\mathcal{J}_{n+1}^\eta$, thus pruning $\mathcal{U}$ by protecting $A(a)$ and $A \sqsubseteq E$. As a consequence, the diagnoses left for $B(a)$ are all the possible hitting sets of the justifications

for this unwanted consequence which do not contain $A(a)$. In other words, the diagnosis will necessarily contain n axioms. On the other hand, if the black-box algorithm is instructed to remove first the axioms that are shared with other justifications from the set $\mathcal{U}$, then the axioms $C_i(a), C_i \sqsubseteq E$ will be protected for some i, making it possible to reach the smallest diagnosis $\{A(a)\}$ which has only one axiom to remove from the ontology to reach a repair.

According to the definition of justification, it cannot be the case that a justification for an error entails another justification for the same error. However, this property may no longer hold after the interactive pruning phase.

Example 8. *Consider the ontology $\mathcal{O}$ with an empty ABox, and a large TBox $\mathcal{T}$ containing the axioms*

$$\alpha_1 : A \sqsubseteq B, \quad \alpha_2 : B \sqsubseteq C, \quad \alpha_3 : C \sqsubseteq \exists r.E, \quad \alpha_4 : C \sqsubseteq \exists r.D, \quad \alpha_5 : D \sqsubseteq E, \quad \ldots, \quad \alpha_{60} : E \sqsubseteq F, \ldots$$

The justifications for the error $A \sqsubseteq \exists r.E$ are

$$\mathcal{J}_1 = \{\alpha_1, \alpha_2, \alpha_3\}, \quad \mathcal{J}_2 = \{\alpha_1, \alpha_2, \alpha_4, \alpha_5\}.$$

Suppose that the user specifies the desired consequence $\beta : C \sqsubseteq \exists r.F$, and the system computes the justification $\mathcal{J} = \{\alpha_3, \alpha_{60}\}$ for β. As a result, α_3 is marked as relevant and protected. Looking into $\mathcal{J}_1$, we see that only α_1 and α_2 could be selected in a diagnosis to compute a repair. Since both of those axioms appear also in $\mathcal{J}_2$, any of these choices is already a hitting set of the justifications and no further analysis is necessary.

This example highlights an important effect of interactive pruning: the decision to preserve an axiom in one justification can directly influence others by narrowing the available options for removal. In particular, when two justifications share a subset of axioms, and pruning removes one of such axioms in order to maintain a desired consequence, the remaining axioms in the second justification may no longer need to be considered during the repair process. As a result, some axioms can be automatically preserved or ignored, further narrowing the set $\mathcal{U}$.

4.3. Automated Pruning: Prolific Axioms

We now change our attention to the second stage of axiom pruning, which allows to automatically prune justifications of size greater than or equal to two. It should be clear that we cannot prune justifications of size one, as any diagnosis must contain them (otherwise, it would not be a hitting set of all justifications). As mentioned earlier, we want to prune justifications according to the prolificacy of their axioms, where prolificacy is a measure of axiom importance for deducing consequences. Defining and assessing axiom prolificacy is not straightforward. Intuitively, prolificacy refers to the number of consequences that depend on the axiom; however, as the number of consequences of an axiom or an ontology is typically infinite, we need a way to restrict to only relevant information. In $\mathcal{EL}^{\bot}$, we focus on *atomic subsumptions* (i.e., GCIs of the form $A \sqsubseteq B$ with $A, B \in N_C$) and *concept name instances* (assertions $A(a)$ with $A \in N_C$ and $a \in N_I$) as basic consequences of interest. The class of these consequences has polynomially many objects in the vocabulary of the ontology. For the rest of this paper, we use $\mathfrak{C}$ to refer to the (finite) class of relevant consequences of interest. In addition, one should also consider the interactions between axioms in the derivation of these consequences.

Example 9. *Consider the very simple ontology $\mathcal{O} := \{\alpha_1 : A \sqsubseteq B,\ \alpha_2 : B \sqsubseteq C,\ \alpha_3 : C \sqsubseteq D\}$. In addition to the axioms, $\mathcal{O}$ entails only the atomic subsumptions $\epsilon : A \sqsubseteq C$, $\gamma : A \sqsubseteq D$, and $\delta : B \sqsubseteq D$.*

We see that none of the axioms, by itself, suffices to derive any of the consequences, yet, in pairs and triples they entail some or all of the consequences:

$$\{\alpha_1, \alpha_2\} \models \{\alpha_1, \alpha_2, \epsilon\}$$
$$\{\alpha_1, \alpha_3\} \models \{\alpha_1, \alpha_3\}$$

$$\{\alpha_2, \alpha_3\} \models \{\alpha_2, \alpha_3, \delta\}$$
$$\{\alpha_1, \alpha_2, \alpha_3\} \models \{\alpha_1, \alpha_2, \alpha_3, \epsilon, \gamma, \delta\}$$

This analysis clearly shows how much each axiom contributes to verify consequences.

Assuming that ϵ is the unwanted consequence, it should be excluded from the number of relevant entailments. We can see that α_1 contributes only to the derivation of γ (and the error), while α_2 and α_3 contribute also to the derivation of δ. Hence, we consider these latter axioms as the most prolific from $\mathcal{O}$.

Without consider prolificacy, any of the two diagnoses $\mathcal{D}_1 := \{\alpha_1\}$ and $\mathcal{D}_2 := \{\alpha_2\}$ is an equally good solution, from a logical point of view, for getting rid of the error. Yet, in light of the entailments computed, we observe that opting for $\mathcal{D}_2$, i.e. removing α_2 from the ontology leads to a repair which does not entail γ, while removing $\mathcal{D}_1$ from $\mathcal{O}$ still yields γ as a consequence. As the goal is to preserve the most consequences, we prefer to use the diagnosis $\mathcal{D}_1$ or, more in general, to remove first the least prolific axioms.

This example has illustrated that including a prolific axiom in a diagnosis may result in the loss of relevant knowledge. To avoid this problem, we suggest to identify a set of highly prolific axioms to preserve in the repair. Moreover, it has explained that to assess the prolificacy of an axiom we should look at how it interacts with others in the knowledge base. There are different possibilities to assess the prolificacy of an axiom: usage statistics (frequency of use in inference tests or user tasks); structural measures (centrality in the graph); impact of removal (found by simulating removal and measuring loss of desired inferences); and many others.

Usage statistics are difficult to obtain, since they require a set of queries that are submitted by users, which typically are not available, nor easy to produce. For what concerns the structural measures, it is possible to estimate the importance of an axiom by analysing the dependency graph or the atomic decomposition the ontology. In such a graph, the knowledge base is partitioned into modules, each containing a set of axioms supporting related entailments [26]. The axioms that appear in many or central modules can be considered prolific, as they contribute to multiple parts of the inferred knowledge. This solution is valuable since the graph clearly displays not just the single axioms, but also how set of axioms interact in a knowledge base, showing the relation of concepts and role assertions. However, analysing such dependency graphs allows just to consider axioms from the TBox, without considering those of the ABox. Instead, our interest is in considering all the axioms. For this reason, we propose impact of removal as an appropriate way to understand the effects of removing one axiom, since it computes which entailments are lost when a given axiom is removed from the ontology. Based on this analysis, we now provide a formal definition of prolificacy, which refers to the number of consequences that are lost if the axiom is removed from the ontology.

Definition 10 (prolificacy)**.** *Let $\mathsf{cl}(\mathcal{O}) = \{\delta \in \mathfrak{C} \mid \mathcal{O} \models \delta\}$ be the* closure *of $\mathcal{O}$ over $\mathfrak{C}$; that is, set of all the consequences in $\mathfrak{C}$ entailed by the ontology $\mathcal{O}$. The* prolificacy *of an axiom $\alpha \in \mathcal{O}$ is defined as*

$$P(\alpha) = |\mathsf{cl}(\mathcal{O})| - |\mathsf{cl}(\mathcal{O} \setminus \{\alpha\})|.$$

Note that this definition of $\mathrm{P}(\alpha)$ makes sense just if $|\mathsf{cl}(\mathcal{O})| < \infty$, which is the reason why we restrict to relevant consequences. Still we emphasise that within the logics of the $\mathcal{EL}$ family, it is common to focus on atomic consequences only. Interestingly, in $\mathcal{EL}^{\bot}$ the full classification (i.e., computing all the atomic subsumptions) and all the basic entailed instances can be computed in polynomial time in the size of the ontology. Thus overall it is possible to compute the prolificacy of all axioms in polynomial time. Yet, to improve efficiency, we focus on some of these axioms only. Recall that the aim of the prolificacy computation is to reduce the search space for finding a repair. For this reason, we only care about the prolificacy of the axioms in $\mathcal{U}$, from which diagnoses are being created. However, looking at prolificacy alone may still lead to errors, specially when some axioms are shared among justifications. This problem is avoided by verifying that a diagnosis still exists, before protecting an axiom from $\mathcal{U}$. However, there is also the issue that prolificacy may be relative among justifications; that is, the most prolific axioms in one justification may be the least prolific in another one. In that case, it is not clear which justification should take precedence in deciding which axioms to protect.

Algorithm 2 Automated Pruning of Prolific Axioms

Require: Justification $\mathcal{J}$ of size > 1, threshold λ
Ensure: $\mathcal{J}' \subseteq \mathcal{J}$ of minimal cardinality such that $P(\mathcal{J}') \geq \lambda \cdot P(\mathcal{J})$.

```
Sort J into {α_1, ..., α_n} such that P(α_1) ≥ P(α_2) ≥ ··· ≥ P(α_n)
J' ← ∅
T ← 0
i ← 1
while T < λ · P(J) do
    J' ← J' ∪ {α_i}
    T ← T + P(α_i)
    i ← i + 1
end while
return J'
```

To avoid these problems of suboptimal solutions and the impossibility of finding a diagnosis, it would be more informative and precise to compute the impact of removal of entire diagnoses, and check the differences among them. However, this would contrast with our aim of not calculating all the diagnoses. So, the solution we propose is to compute the prolificacy of only the axioms in a justification without shared axioms. Computing the impact of removal for such axioms, is actually a good estimate of the role they would play in the repair. In doing so, we avoid all the problems caused by axioms that are shared, and we reduce the number of axioms for which we compute $P(\alpha)$. Next, we describe the algorithm to prune such justifications.

An Algorithm for Automated Pruning. Once the system has computed the impact of removal $P(\alpha)$ for each axiom $\alpha \in \mathcal{J}$, it sorts the axioms in descending order of $P(\alpha)$. After that, the algorithm should remove from $\mathcal{J}$ the most prolific axioms in a quantity determined by a parameter λ as explained next.

Given a justification $\mathcal{J} = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ for an error α, let $P(\mathcal{J}) = \sum_{i=1}^{n} P(\alpha_i)$ denote the total impact of removal of all axioms in $\mathcal{J}$. The algorithm then identifies the smallest subset $\mathcal{J}' \subseteq \mathcal{J}$, taken in order of decreasing $P(\alpha)$, such that:

$$\sum_{\alpha \in \mathcal{J}'} P(\alpha) \geq \lambda \cdot P(\mathcal{J})$$

where λ is a threshold parameter in the interval $[0, 1]$. The system protects the axioms in this subset $\mathcal{J}'$ from being removed in the current pruning step, since they are the most prolific ones, accounting for at least a fraction λ of the justification's inferential power. All the remaining axioms $\mathcal{J} \setminus \mathcal{J}'$ are kept in $\mathcal{U}$ and can be safely considered for possible removal in the subsequent interactive step.

Note that in Example 9 we said that, when computing the prolificacy of an axiom, we should not count the error as a consequence. However, note that there is no need to modify $P(\alpha)$, since we are now calculating the prolificacy just for the axioms in the justifications that do not have axioms in common with others. So, in the case where different justifications for the error exists, the unwanted consequence will not be counted for any of the axioms in the justification; whereas, in the case where the justification for whose axioms we are calculating the prolificacy, the error will be counted as a consequence in $P(\alpha)$ of all the axioms.

This strategy is a trade off between simplification and preservation of relevant knowledge, making the pruning phase both flexible and adaptive to the structure of each justification. In fact, it ensures that longer justifications are pruned more that smaller ones, reducing the number of axioms in $\mathcal{U}$ and facilitating the user in the computation of a diagnosis.

Again, note that automating the process of finding the prolific axioms is better than asking to the user to autonomously identify all the axioms that are important. This is due to the fact that the knowledge engineer may not be completely aware of which axioms of the knowledge base are important, in other

words they may not see the consequences of keeping or removing a given axiom. The error of keeping a redundant axiom, which may occur if the knowledge engineer is free to choose, is avoided using the impact of removal.

4.4. The Tree: Selection of the Diagnosis

So far, we have discussed strategies to prune the search space to find the "true" repair in the sense of restricting the number of axioms (and questions) to present to the user. This phase has ensured that all the remaining repairs verify the desired consequence of the user. Moreover, thanks to the identification of prolific axioms, we have an approximation of the fact that these repairs will be maximal (according to the desiderata of the user). At this point, most of the issues of the naïve approach have been overcome, and, as explained, we can continue either in an automatic way, or by presenting questions to the user. Next, the two options are presented.

For the manual one, we still need to think of an effective way to present justifications and axioms to the user. To this end, we will use some heuristics. For the automatic alternative, we will base the computation of the diagnosis on assumption that we prefer to apply minimal changes to the ontology, thus trying to remove the smallest set of axioms from $\mathcal{U}$. To this aim, the algorithm will present the smallest possible diagnosis, allowing the user to approve or modify it. Recall that the two options (manual and automatic) are not mutually exclusive nor fully intertwined, since the knowledge engineer is free to stop whenever they want, and let the process finish automatically.

If the knowledge engineer wants to control the construction of a diagnosis, the system should let them manually choose one axiom from each justification. Since they have already specified their preferences on consequences, we can safely present one justification at a time, and let the user choose from them. However, there are still multiple ways of presenting axioms and justifications to the user, each resulting in a different sequence of decisions. Some of these sequences are smaller than others; thus, presenting the justifications in a certain order may reduce the number of questions. Also, since after the pruning phase all the possible reachable repairs will verify the desired consequences, all possible diagnoses that can be built are considered to be already adequate from the user requirements point of view—that is, from the perspective of preserving the true knowledge from the domain. At this point of the procedure, since the knowledge engineer has already freely expressed their desiderata on consequences (and axioms), the user discovers their preferences in the interaction with the system. Thus, the very idea of presenting justifications in an "optimal" way is flawed. At this stage, what the system can do is to guide the user, and order justifications and axioms thanks to heuristics that help minimise the number of questions and maximise the utility of the answers.

In particular, the heuristics should act on the two dimensions that render the number of options to consider too big from the user's point of view: the length of each justification; and the number of justifications.

The issue of the length of each justification refers to the fact that a single justification may have a number of axioms too high to be manageable by the user. However, at this point of our procedure, the problem has mostly been solved thanks to both interactive and automated pruning. It may still be the case that long justifications have survived, but it is unlikely. In fact, if such justifications exist, it is because they did not have axioms in common with any of the justifications computed for the desired consequences. At the same time, they must also be justifications that share axioms with others. Otherwise, they would have been reduced during the automated pruning. As we will see in what follows, this observation provides us with the solution for solving the problem.

For what concerns the other dimension, the number of justifications may indeed be too long. However, we can try to overcome the problem by suggesting to the user to remove axioms that are shared by most of the justifications. When these axioms are removed from one justification, as an inherited answer they are also removed from all the justifications where they appear. Hence, we do not need to present those justifications to the user, and the number of questions is reduced. Moreover, consider that, in any case, if the amount of questions is too large for the user, they always have the option to let the system automatically compute the diagnosis.

In case of automatic computation, the heuristics help the user to orientate among the axioms of the justifications. Different heuristics may serve different goals, so, before implementing one heuristic, we have to figure out what each heuristic is useful for. If our scope is to apply minimal changes to the ontology, we should try to build the smallest possible diagnosis via shared axioms. If the aim is to preserve as much consequences as possible, we should start by presenting to the user the axioms less prolific for each justification; if our aim is to understand as quick as possible which axiom should be removed from each justification, we should present the smaller justifications first. These heuristics are presented more in detail in the next subsections.

In any case, it is important to remember (and to stress) that, in the end, the decisions of the user should be the primary consideration. The system defines an order just to help the user, but should not force them to choose one axiom over another. Also, we assume that the knowledge engineer knows how the system works, in the sense that they know that the machine orders axioms based on the estimation of which ones are preferable to remove first. Therefore, we stated that the approach is interactive, as it is based on an efficient Human-Computer collaboration.

4.5. Shared Axioms

As we have seen, axioms that are shared by different justifications may cause issues in the naïve approach, since they influence different justifications. We now show how they can be used to the user's advantage. The idea is that, in ordering the axioms in the justification that the system presents to the user, shared axiom should be presented first. In fact, these are the axiom that appear more frequently than others in the justifications for the error. If such axioms are removed, they are likely to lead to the smallest possible diagnosis.

Since the aim is to reduce as fast as possible the search space for repairs, the idea behind presenting shared axioms first, is that these allow to build a diagnosis in fewest step. The reason is that, once selected (and removed) from one justification, they are discarded also from all other justifications where they appear. Therefore, there is no need to select other axioms for such justifications, reducing the number of questions. The extreme case is that of a shared axiom appearing in all justifications, in such a case, the shared axiom constitutes a diagnosis by itself.

The idea of shared axioms can also be applied to, and is particularly useful in cases where there are more errors in an ontology, for instance when errors arise from merging knowledge bases. In this case we can call maximally shared axioms those that are the most common among the justifications of the two errors.

The benefits of this strategy are two, namely, that it reduces the number of questions the user has to answer, and that it leads to apply minimal changes to the knowledge base from the syntactic point of view. Of course, it only works if the shared axioms are actually removed by the user. On the other hand, if the user wants to keep the axioms, we do not reduce our search space by a lot, but we do not have to ask for such axioms again in the next justifications. Otherwise, if the system would ask for those axioms in following justification, and the user chooses to remove the shared axiom, the minimality of the diagnosis is lost. This is yet another example of the fact that a greedy approach leads to suboptimal results, as we have seen for the naïve approach.

It could also be useful for the user to see the number of justifications left to be answered. This number shows the worst case scenario, and should be updated after each axiom is chosen from a justification.

4.6. Revisiting Prolific Axioms

We have seen that within the justifications, it is a good idea to order axioms presenting those that are shared first. However, what is the system supposed to do with justifications that do not share axioms with others? Recall that in the automated pruning phase, we ordered such axioms according to their prolificacy. Hence, we can reuse this previous computation to order the axioms presenting them from the less to the most prolific one. Indeed, we want to suggest to the user to remove the less prolific axioms, so that the least consequences are removed from the final repair. Moreover, since we have

computed the entailments for each axiom, it is also possible for the user to see which are the effects of removing a certain axiom; i.e. which are the consequences that are lost. This is actually much more informative from the user's point of view than looking at the single axioms.

4.7. A Heuristic for Ordering Justifications

Recall that, within an interactive diagnosis process, the order in which justifications are presented to the user plays a key role in optimizing the interaction. In this section we explain that the simple heuristic of presenting the shortest justifications first is a good strategy to order them.

First note that shorter justifications contain fewer axioms, which means that the user can analyse them more quickly and with less effort. So, by prioritising shorter justifications, the system allows the user to tackle simpler cases first, reducing the risk of early disengagement. This aligns with the goal of reducing the cognitive load of the knowledge engineer.

Second, if the user chooses to stop interacting with the system before completing the entire process, they will have only dealt with simpler and smaller justifications, avoiding the burden of analysing more complex ones and possibly selecting axioms from more justifications. This ensures that even partial interaction can contribute meaningfully to refining the repair process. For example, if the user has energy to analyse just twenty axioms and we present a justification consisting of twenty axioms, they will ultimately analyse only this single justification. On the other hand, if we present four justifications consisting of five axioms each, the user will still consider twenty axioms, but will respond to four justifications.

In summary, presenting the shortest justifications first balances efficiency and user engagement, supporting a more user-friendly and practical interactive repair process.

5. Weakening the Diagnosis

In the previous sections, our focus was on finding ways to minimise the human effort in the computation of a repair, by protecting some axioms that are considered correct or at least wanted in the ontology. We should bear in mind that the user, when presented with the different questions, is iteratively selecting a diagnosis: the axioms signalled as erroneous.

From the point of view of classical repairs, the final step after computing a diagnosis is to remove it from the ontology, thus obtaining a repair *à la* Definition 3. However, removing full axioms from the ontology can be seen as an extreme solution: one should take into account that axioms are not added arbitrarily, but there is usually a motivation behind them. Moreover, some of the information provided by the axiom may be correct, even if not all of it. We partially considered all these issues by requiring the ontology to be in normal form. Yet, even in this simplified scenario, it is worth asking whether removing full axioms is the best solution.

A proposal to preserve more of the original knowledge is to *weaken* rather than delete, the elements of the diagnosis. This approach is typically known as *gentle* repair [10]. Weakening means to make the axiom less restrictive so that more interpretations satisfy it—which in turn removes some of the consequences. Obviously, the axioms should be weakened at least enough to make the error disappear.

A common approach for axiom weakening is to apply concept refinement operators to the elements of the axioms. Briefly, to weaken a GCI of the form $C \sqsubseteq D$, one may *specialise* C to a concept that is necessarily subsumed by C, or *generalise* D to a concept subsuming it. In both cases, the result is an axiom which has strictly more models than $C \sqsubseteq D$. Analogously, the assertion $C(a)$ can be weakened by generalising C. Note that if, as assumed, we have an $\mathcal{EL}^{\bot}$ ontology in normal form, then syntactic refinement operators are not only quite simple—in fact, concepts in the left-hand side of a normalised GCI can only specialise to $\bot$—but easily lead to tautologies, which is equivalent to removing the axiom. Hence, although gentle repairs can be very useful to preserve more consequences, on normalised ontologies their impact is limited. However, if the ontology is not originally in normal form or if the language under consideration is more expressive, they could provide better results.

6. Conclusions

It is well known that ontology engineering is an error-prone task, which has attracted quite some attention over the years in the development of methods which automatically correct a given ontology. Yet, there may be many different ways to correct an error. Many different ideas have been proposed for selecting the "best" among them like choosing largest correct sub-ontology or maximising the set of preserved consequences. However, these syntactic and semantic approaches ignore one fundamental issue: the *correctness* of an ontology—i.e., its expression of the knowledge domain under consideration—is a meta-logical property, which can only be verified by a domain expert. In other words, there is no way to guarantee that repairs with *minimum change* actually reflect the knowledge of the domain.

On the other hand, the size of the ontology and the potential number of justifications (and diagnoses) and axioms appearing in them makes it impossible for a knowledge engineer to select the most adequate repair from all the available options without the help of an automated tool. Moreover, human experts are costly and tire fast, so we should try to minimise their effort.

In this paper, we proposed an intermediate path, where the knowledge engineer is iteratively asked to choose one axiom from each justification to be removed as considered wrong, without the need to compute all justifications or diagnoses at any time. The main novelty of our work is to include first a pruning phase, which identifies axioms that are considered "correct" and hence safe from removal. This pruning takes input from the knowledge engineer, who specifies a set of consequences that need to be preserved, to choose minimal conditions that guarantee this request. It also applies an heuristic for minimal change, based on a new notion of *prolificacy*: the impact an axiom has on the number of consequences derived.

Our procedure computes a diagnosis, which refers to the class of axioms identified as wrong in the ontology. At this point, one could decide to simply remove these axioms (yielding a repair) or to weaken them towards a gentle repair, as suggested in previous work.

It should be clear that the choice of prolificacy as a way to find axioms to be preserved is a heuristic based on the centrality (or importance) of the axiom for the concept hierarchy. As mentioned already, no purely logical, syntactic, or semantic measure can fully characterise the meta-logical property of correctness for the representation domain in consideration. Thus, this one may also yield an incorrect repair. However, we argue that exactly their centrality makes these axioms to be subject to deeper scrutiny, and hence less likely to be wrong from the domain point of view.

One could imagine other measures for automated pruning like, e.g., a degree of trust associated to axioms, depending on their source and use in the ontology. In this way, one can measure trust on one or more consequences using the techniques from provenance [27, 28]. However, such approaches require further input from the knowledge engineer, who must specify those trust degrees, even for axioms which may seem incomparable. Another idea which was recently proposed was to learn the repair patterns from edit history of ontologies [29]; yet, this strategy requires data which is not always available.

For this work we focused on the light-weight DL $\mathcal{EL}^{\bot}$. The main reason for this is that its polynomial-time reasoning services allow us to provide polynomial-time algorithms for each of the steps of the iterative procedure, except for the initial one of computing the union of all justifications $\mathcal{U}$, which requires superpolynomial time (unless $P = NP$) [23]. However, the latter can be seen as a pre-processing step to be executed off-line when the error is detected. Importantly, as we never compute all diagnoses or justifications, we avoid a potentially exponential computation in the intermediate steps. Note that the same ideas can be applied to other, more expressive, logics. Indeed, our approach is designed to be modular, and thus applicable to other scenarios without much modification.

As part of our future work, we expect to build a better understanding about the properties of prolificacy and develop more efficient methods for computing them directly. Moreover, we will try to understand the guarantees that preserving axioms based on their prolificacy provides from the ontology repair and belief revision point of view. Note also that our approach is parameterised to maintain a proportion (λ) of the prolificacy of a justification, but we did not specify a precise value for this parameter. From a practical point of view, we want to study the impact of this parameter on the overall

effort and quality of the repair process.

Acknowledgments

Part of this work was developed during a research visit by the first author to TU Dresden, funded by the University of Milan under the programme "Tesi all'estero." We are thankful to Prof. Franz Baader for his advise and fruitful discussions during this visit and to Christian Alrabbaa for clarifications.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider, The Description Logic Handbook, Cambridge University Press, 2007.

[2] F. Baader, S. Brandt, C. Lutz, Pushing the EL envelope, in: L. P. Kaelbling, A. Saffiotti (Eds.), Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05), Professional Book Center, 2005, pp. 364–369. URL: http://ijcai.org/Proceedings/05/Papers/0372.pdf.

[3] B. Suntisrivaraporn, F. Baader, S. Schulz, K. A. Spackman, Replacing SEP-Triplets in SNOMED CT using tractable description logic operators, in: R. Bellazzi, A. Abu-Hanna, J. Hunter (Eds.), Proceegins of the 11th Conference on Artificial Intelligence in Medicine, AIME 2007, volume 4594 of *Lecture Notes in Computer Science*, Springer, 2007, pp. 287–291. doi:10.1007/978-3-540-73599-1_38.

[4] S. Schlobach, R. Cornet, Non-standard reasoning services for the debugging of description logic terminologies, in: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003), Morgan Kaufmann Publishers Inc., 2003, pp. 355–360.

[5] B. Parsia, E. Sirin, A. Kalyanpur, Debugging OWL ontologies, in: Proceedings of the 14th International Conference on World Wide Web, WWW '05, Association for Computing Machinery, New York, NY, USA, 2005, pp. 633–640. URL: https://doi.org/10.1145/1060745.1060837. doi:10.1145/1060745.1060837.

[6] F. Baader, B. Suntisrivaraporn, Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}$, KR-MED 2008 (2008) 1.

[7] R. Peñaloza, Axiom pinpointing, in: G. Cota, M. Daquino, G. L. Pozzato (Eds.), Applications and Practices in Ontology Design, Extraction, and Reasoning, volume 49 of *Studies on the Semantic Web*, IOS Press, 2020, pp. 162–177. URL: https://doi.org/10.3233/SSW200042. doi:10.3233/SSW200042.

[8] F. Baader, R. Peñaloza, B. Suntisrivaraporn, Pinpointing in the description logic $\mathcal{EL}+$, in: J. Hertzberg, M. Beetz, R. Englert (Eds.), Proceeding of the 30th Annual German Conference on AI, KI 2007, volume 4667 of *Lecture Notes in Computer Science*, Springer, 2007, pp. 52–67. doi:10.1007/978-3-540-74565-5_7.

[9] N. Troquard, R. Confalonieri, P. Galliani, R. Peñaloza, D. Porello, O. Kutz, Repairing ontologies via axiom weakening, in: S. A. McIlraith, K. Q. Weinberger (Eds.), 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, AAAI Press, 2018, pp. 1981–1988. URL: https://doi.org/10.1609/aaai.v32i1.11567. doi:10.1609/AAAI.V32I1.11567.

[10] F. Baader, F. Kriegel, A. Nuradiansyah, R. Peñaloza, Making repairs in description logics more gentle, in: M. Thielscher, F. Toni, F. Wolter (Eds.), Proceedings of the Sixteenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2018), AAAI Press, 2018, pp. 319–328. URL: https://aaai.org/ocs/index.php/KR/KR18/paper/view/18056.

[11] U. Furbach, C. Schon, Semantically guided evolution of SHI ABoxes, in: D. Galmiche, D. Larchey-Wendling (Eds.), Automated Reasoning with Analytic Tableaux and Related Methods, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 134–148.

[12] M. Ludwig, R. Peñaloza, Error-tolerant reasoning in the description logic $\mathcal{EL}$, in: E. Fermé, J. Leite (Eds.), Proceedings of the 14th European Conference on Logics in Artificial Intelligence (JELIA 2014), volume 8761 of *Lecture Notes in Computer Science*, Springer, 2014, pp. 107–121. URL: https://doi.org/10.1007/978-3-319-11558-0_8. doi:10.1007/978-3-319-11558-0_8.

[13] M. Bienvenu, On the complexity of consistent query answering in the presence of simple ontologies, in: J. Hoffmann, B. Selman (Eds.), Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, AAAI Press, 2012, pp. 705–711. doi:10.1609/AAAI.V26I1.8218.

[14] F. Baader, F. Kriegel, A. Nuradiansyah, Error-tolerant reasoning in the description logic $\mathcal{EL}$ based on optimal repairs, in: G. Governatori, A. Turhan (Eds.), Proceedings of the 6th International Joint Conference on Rules and Reasoning (RuleML+RR 2022), volume 13752 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 227–243. URL: https://doi.org/10.1007/978-3-031-21541-4_15. doi:10.1007/978-3-031-21541-4_15.

[15] E. Fermé, S. O. Hansson, Belief Change: Introduction and Overview, Springer, 2018.

[16] J. Lehmann, P. Hitzler, Foundations of refinement operators for description logics, in: H. Blockeel, J. Ramon, J. Shavlik, P. Tadepalli (Eds.), Inductive Logic Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 161–174.

[17] F. Baader, P. Koopmann, F. Kriegel, Optimal repairs in the description logic $\mathcal{EL}$ revisited, in: Proceedings of the 18th European Conference on Logics in Artificial Intelligence (JELIA 2023), September 20–22, 2023, Dresden, Germany, volume 14281 of *Lecture Notes in Computer Science*, Springer, 2023, pp. 11–34. doi:https://doi.org/10.1007/978-3-031-43619-2_2.

[18] F. Kriegel, Beyond optimal: Interactive identification of better-than-optimal repairs, in: Proceedings of the 40th ACM/SIGAPP Symposium On Applied Computing (SAC 2025), Association for Computing Machinery, 2025. doi:https://doi.org/10.1145/3672608.3707750.

[19] F. Baader, B. Ganter, B. Sertkaya, U. Sattler, Completing description logic knowledge bases using formal concept analysis, in: M. M. Veloso (Ed.), Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), 2007, pp. 230–235. URL: http://ijcai.org/Proceedings/07/Papers/035.pdf.

[20] B. Sertkaya, OntoComP: A Protégé plugin for completing OWL ontologies, in: L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, R. Mizoguchi, E. Oren, M. Sabou, E. Simperl (Eds.), Proceedings of the 6th European Semantic Web Conference, ESWC 2009, volume 5554 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 898–902. doi:10.1007/978-3-642-02121-3_78.

[21] Y. Kazakov, M. Krötzsch, F. Simancik, The incredible ELK - from polynomial procedures to efficient reasoning with EL ontologies, J. Autom. Reason. 53 (2014) 1–61. doi:10.1007/S10817-013-9296-3.

[22] A. Kalyanpur, B. Parsia, M. Horridge, E. Sirin, Finding all justifications of OWL DL entailments, in: K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, L. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, P. Cudré-Mauroux (Eds.), The Semantic Web, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 267–280.

[23] R. Peñaloza, B. Sertkaya, Understanding the complexity of axiom pinpointing in lightweight description logics, Artif. Intell. 250 (2017) 80–104. doi:10.1016/J.ARTINT.2017.06.002.

[24] R. Peñaloza, Axiom pinpointing in description logics and beyond, Ph.D. thesis, Dresden University of Technology, 2009.

[25] F. Baader, I. Horrocks, C. Lutz, U. Sattler, An Introduction to Description Logic, Cambridge University Press, 2017.

[26] C. Del Vescovo, B. Parsia, U. Sattler, T. Schneider, The modular structure of an ontology: Atomic decomposition, in: T. Walsh (Ed.), IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011, IJCAI/AAAI, 2011, pp. 2232–2237. URL: https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-372. doi:10.5591/978-1-57735-516-8/IJCAI11-372.

[27] C. Bourgaux, P. Bourhis, L. Peterfreund, M. Thomazo, Revisiting semiring provenance for datalog, in: G. Kern-Isberner, G. Lakemeyer, T. Meyer (Eds.), Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022, 2022. URL: https://proceedings.kr.org/2022/10/.

[28] C. Bourgaux, A. Ozaki, R. Peñaloza, L. Predoiu, Provenance for the description logic ELHr, in: C. Bessiere (Ed.), Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, ijcai.org, 2020, pp. 1862–1869. doi:10.24963/IJCAI.2020/258.

[29] T. P. Tanon, C. Bourgaux, F. M. Suchanek, Learning how to correct a knowledge base from the edit history, in: L. Liu, R. W. White, A. Mantrach, F. Silvestri, J. J. McAuley, R. Baeza-Yates, L. Zia (Eds.), Proceedings of The World Wide Web Conference, WWW 2019, ACM, 2019, pp. 1465–1475. doi:10.1145/3308558.3313584.