Making Decisions with Knowledge Base Repairs

Rafael Peñaloza

University of Milano-Bicocca, Milano, Italy
`rafael.penaloza@unimib.it`

Abstract. Building large knowledge bases (KBs) is a fundamental task for automated reasoning and intelligent applications. Needing the interaction between domain and modeling knowledge, it is also error-prone. In fact, even well-maintained KBs are often found to lead to unwanted conclusions. We deal with two kinds of decisions associated with faulty KBs. First, which portions of the KB (and their conclusions) can still be trusted? Second, which is the correct way to repair the KB? Our solution to both problems is based on storing all the information about repairs in a compact data structure.

1 Introduction

In logic-based knowledge representation, the goal is to encode the relevant knowledge of an application domain through a collection of axioms, which intuitively restrict the way in which the symbols used may be interpreted, so as to provide them with an unambiguous and clear meaning. Such a collection of axioms is known as a knowledge base (KB). Historically, many knowledge representation languages have been proposed; most notably, perhaps, are propositional logic [6], constraint systems [1], and description logics (DLs) [2]. Their success has led to the creation of more and larger KBs.

It should come as no surprise that constructing a KB, which requires a combination of domain and modeling knowledge, is an arduous and error-prone task. As a consequence, it is not uncommon to detect errors even in well-maintained KBs. Unfortunately, after an error has been detected, it is also quite difficult to identify the main sources, and select the adequate correction of the fault. Moreover, KB updates are typically subject to a production cycle that prevents them from publishing corrected versions on-demand. For example, Snomed CT [15]—a very large KB about medical terms—publishes two updated versions every year. Thus, even when a KB is known to be faulty, it may be necessary to wait for several months before a corrected version is available.

Since KBs are not static entities to be observed, but rather tools necessary for automated reasoning in practical scenarios, users cannot simply drop a faulty KB and wait for the next version to appear. At the same time, they cannot ignore the fact that some of the knowledge that they contain is wrong. Hence, they need to be able to use the KB, but in a way that preserves some guarantees of correctness. In this paper, we propose a method for reasoning about consequences that takes into account the *repairs* of the KB. In a nutshell, the repairs correspond to the ways in which all errors may be removed, by deleting a minimal amount of axioms from the KB. Our approach consists in encoding all repairs through a Boolean function over the axioms in the KB. This function can be easily updated if new errors are encountered, and can then be used to decide whether a consequence follows from one or all repairs. Thus, our method can be used throughout deployment time, collecting all known errors as they are detected, and providing guarantees over all reasoning results.

At some point, the knowledge engineer will take control of the KB, and will need to decide which of the many potential repairs is the right one; that is, which axioms are indeed wrong. We emphasise that this task cannot be automated because it needs the expert human knowledge to discern correctness. Rather than making them verify each axiom, we devise a method that suggest to them which axioms to check first, in order to reduce the search space efficiently. With the help of this method, finding the right repair requires minimal human effort, which is the most expensive resource in the repairing scenario.

2 Preliminaries

We consider an abstract logic-based knowledge representation language, where explicit knowledge is expressed via a set of constraints (or axioms), and logical entailments derive other implicitly encoded knowledge. For simplicity, we consider that axioms and implicit consequences have the same shape, although the results can be easily generalised to avoid this restriction (see e.g. [4, 11]).

Formally, a *knowledge representation language* is defined by an infinite set $\mathfrak{A}$ of (well-formed) *axioms* and an *entailment relation* $\models \subseteq \mathscr{P}_{\mathsf{fin}}(\mathfrak{A}) \times \mathfrak{A}$, where $\mathscr{P}_{\mathsf{fin}}(\mathfrak{A})$ denotes the class of all finite subsets of $\mathfrak{A}$. In general, we call any finite subset of $\mathfrak{A}$ a *knowledge base* (KB). Thus, a knowledge representation language provides the syntax of the axioms that form a knowledge base, and the semantics are given through the entailment relation expressing what consequences can be derived from which KBs. We use infix notation for the entailment relation; i.e., $\mathcal{K} \models \alpha$ expresses that the KB $\mathcal{K}$ entails the axiom α. In this work, we are only interested in *monotonic* knowledge representation languages; these are those where the entailment relation is monotonic in the sense that for every two KBs $\mathcal{K}$, $\mathcal{K}'$, and axiom α, if $\mathcal{K} \models \alpha$ and $\mathcal{K} \subseteq \mathcal{K}'$, then $\mathcal{K}' \models \alpha$.

For the sake of building an example and improving understanding of the results presented here, we consider a very simple knowledge representation language consisting of directed graphs and reachability between nodes. In this case, given an infinite set $\mathcal{V}$ of *nodes*, an axiom is of the form $v \to w$, where $v, w \in \mathcal{V}$. Intuitively, this axiom expresses that w is reachable from v. A KB, that is, a finite set of axioms, can be represented as a finite graph. The graph $\mathcal{K}$ entails the axiom $v \to w$ iff the node w is reachable from v in $\mathcal{K}$ (see Figure 1).

An important operation for KB update and belief revision is known as *contraction*. The goal of this operation is to remove a consequence from a KB; in other words, given the KB $\mathcal{K}$ and the axiom α, the contraction $\mathcal{K} - \alpha$ should yield a KB $\mathcal{K}'$ such that $\mathcal{K}' \not\models \alpha$. While it is possible to define many different kinds of contractions, we focus on one that provides minimal syntactic changes to the original KB, which is based on the notion of a repair.

Definition 1 (repair). *A* repair *of $\mathcal{K}$ w.r.t. α is a sub-KB $\mathcal{K}' \subseteq \mathcal{K}$ such that (i) $\mathcal{K}' \not\models \alpha$ and (ii) for all $\mathcal{K}' \subset \mathcal{L} \subseteq \mathcal{K}$, $\mathcal{L} \models \alpha$.*

In words, a repair is a maximal sub-KB of $\mathcal{K}$ that does not entail α. It is important to notice that repairs are not unique. In fact, removing a single consequence from a KB can produce exponentially many such repairs [10]. To try to reduce the number of options, some heuristics can be proposed; for example, to consider only the repairs with the largest cardinality. However, in general, this is still insufficient to identify one single solution. In order to find only one (adequate) solution, human intervention is needed to provide the expert domain knowledge that distinguishes the axioms that are in fact incorrect w.r.t. the current knowledge and should hence be removed.

As it is well known in the area of belief revision, the problem of choosing the right repair becomes even more crucial in the presence of an iterated contraction [8]; that is, when more than one consequence is to be removed in successive steps. Obviously, making a wrong choice of repair in any given step will negatively affect the following contraction steps. For example, suppose that we are only interested in finding a repair of maximum cardinality that removes a set of consequences from $\mathcal{K}$. If we simply choose one maximum cardinality repair at each step, we are not guaranteed to end up with a solution of maximum cardinality. If human intervention is necessary, then asking an expert to choose the right repair at every single contraction step becomes excessively expensive in terms of human expert resources.

Henceforth, it is useful to consider the dual notion of a repair, called a justification, that corresponds to a minimal sub-KB entailing a consequence.

Definition 2 (justification). *A* justification *of a consequence α w.r.t. the KB $\mathcal{K}$ is a set $\mathcal{K}' \subseteq \mathcal{K}$ such that $\mathcal{K}' \models \alpha$ and for all $\mathcal{L} \subset \mathcal{K}'$ it holds that $\mathcal{L} \not\models \alpha$.*

It is well known (see [16]) that repairs and justification are dual in the sense that are repair can be obtained by removing at least one axiom from every justification, and justifications are obtained similarly from the axioms that are removed to form repairs. This duality will allow us to exploit efficient methods developed for finding justifications to deal with repairs as well.

In the following sections, we first show how to deal with iterative contractions automatically without losing any valuable information, and then provide a method for helping a human expert to choose the right solution among the potentially exponentially many available, minimising the need of human effort.

3 Iterative Contractions

We are interested in an iterative process for repairing a KB. Starting from a given KB $\mathcal{K}$, we assume that a user is detecting a sequence of erroneous consequences that they try to bypass, while waiting for an official correction by the knowledge

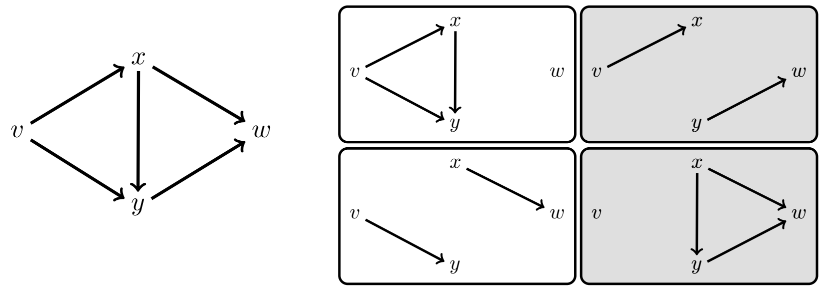

Fig. 1. A KB, and its repairs w.r.t. $v \to w$ and $v \to y$.

experts. In the meantime, the KB is still operational and new errors may be derived. At any point in this process, the knowledge expert may attempt to find the *correct* repair, for which automated support should be given.

In order to preserve all the information needed for computing the correct repair, *all* possible solutions should be stored; otherwise, we risk removing the best option from the search space. As mentioned already, even at the first contraction, the number of repairs may become exponential on the size of the KB. Moreover, suppose that we have already a set of repairs obtained after a sequence of contractions. When the following erroneous consequence is detected, it is necessary to contract each of these repairs. In practice, this means potentially computing an exponential number of new KBs for each previously known repair, with the additional cost of verifying that no set appears twice, and that they are all actual repairs. For example, Figure 1 depicts a simple KB and its repairs w.r.t. the consequence $v \to w$ (all the squared KBs), and w.r.t. the two errors $v \to w, v \to y$ (with a grey background). Notice that the set of axioms obtained by removing $v \to y$ from the lower-left repair is not a repair, because it is strictly contained in the lower-right repair.

To solve this issue, it was proposed in [14] to succinctly encode the class of all repairs by associating with each axiom in the KB a propositional formula expressing to which repairs this axiom belongs, and to which it does not. Although effective, this approach has several issues; most importantly, there is a trade-off between the succinctness of the representation, and the readability of the repairs. That is, reducing the size of the representation comes at the cost of making it harder to know the axioms that belong to each repair.

We propose an alternative approach, encoding the class of all repairs by a Boolean function over the original KB, called a repair function.

Definition 3 (repair function). *Let $\mathcal{K}$ be a KB and C a set of consequences. A function $\mathsf{rep}_C : \mathscr{P}(\mathcal{K}) \to \{0,1\}$ is a* repair function *of $\mathcal{K}$ w.r.t. C iff for every $\mathcal{K}' \subseteq \mathcal{K}$ it holds that $\mathsf{rep}_C(\mathcal{K}') = 0$ iff there is an $\alpha \in C$ such that $\mathcal{K}' \models \alpha$.*

Intuitively, a repair function takes as input a sub-KB $\mathcal{K}'$ of $\mathcal{K}$, and returns 1, if $\mathcal{K}'$ does not entail any of the (erroneous) consequences in C (that is, if $\mathcal{K}'$ gets rid of all errors), and 0 otherwise. Notice, however, that $\mathsf{rep}_C(\mathcal{K}') = 1$ does not mean that $\mathcal{K}'$ is a repair, since the maximality criterion is not being considered in the definition of this function, but it is easy to detect the repairs from it. When clear by the context, we remove the subscript C from the name rep.

If we see every axiom as a propositional variable, then rep is simply a propositional formula over the variables in $\mathcal{K}$. Hence, we can encode this function using any of the existing datatypes for compact representation of formulas, such as circuits [17], binary decision diagrams (BDDs) [9], or the more recent sentential decision diagrams (SDDs) [7]. For the sake of an example, and to be consistent with previous work (as should become clear later in this paper), we consider BDDs. In a nutshell, a BDD is a directed acyclic graph with one root node and two terminal nodes (called 0 and 1) such that every non-terminal node node is labelled with a propositional variable and has exactly two successors called the *low* and the *high* branch, and in every path from the root to a terminal node the same variable can appear at most once. A valuation is checked by the BDD by traversing the graph starting from the root and following the low branch if the variable is set of false, and the high branch otherwise. The valuation is a model of the BDD iff this traversal leads to the terminal 1.

To achieve our goal, we build a sequence of repair functions, updating the last one whenever a new fault is found. At the beginning, we have only a KB $\mathcal{K}$, and no errors; that is, $C = \emptyset$. Hence, we have a trivial repair function that maps every subset of $\mathcal{K}$ to 1. In terms of propositional formulas, we represent it as a tautology, which corresponds to the BDD having only one node 1.

Suppose now that we have already detected a set of faults C, and that we have computed a repair function rep for it. For simplicity, we observe this function as a propositional formula. Our goal now is, given a new unwanted consequence α, to compute a repair function rep' w.r.t. $C \cup \{\alpha\}$. One could, of course, compute this new function from scratch, ignoring the properties of rep. We instead opt to update rep to exclude those sub-KBs of $\mathcal{K}$ that entail α from being accepted.

Recall that a justification is a minimal sub-KB that entails α. Intuitively, if $\mathcal{M}$ is a justification, then any set containing $\mathcal{M}$ will entail α. If $\mathcal{M}$ was the *only* justification, then minimality would imply that excluding at least one axiom from $\mathcal{M}$ would also get rid of the consequence α; i.e., if $\mathcal{M} \setminus \mathcal{K}' \neq \emptyset$, then $\mathcal{K}' \not\models \alpha$ holds for all $\mathcal{K}' \subseteq \mathcal{K}$. Since there may exist more than one justification, we need to ensure that none of them is contained in a set to guarantee that α is not entailed. We do this with the help of a *pinpointing formula* [3].

Definition 4 (pinpointing formula). *Let $\mathcal{K}$ be a KB, and α a consequence. A* pinpointing formula *for α w.r.t. $\mathcal{K}$ is a Boolean function $\mathsf{pin}_\alpha : \mathscr{P}(\mathcal{K}) \to \{0, 1\}$ such that, for every $\mathcal{K}' \subseteq \mathcal{K}$ it holds that $\mathsf{pin}_\alpha(\mathcal{K}') = 1$ iff $\mathcal{K}' \models \alpha$.*

Notice that the pinpointing formula is the analogous notion to the repair function, when speaking about justifications instead of repairs. The name is chosen to preserve consistency with existing terminology. *Pinpointing* refers to the task of computing all justifications, and the notion of *formula* arises from considering all axioms as propositional variables inside the Boolean function pin, as we did before for the repair function. It is perhaps worth noting that Definition 4 is

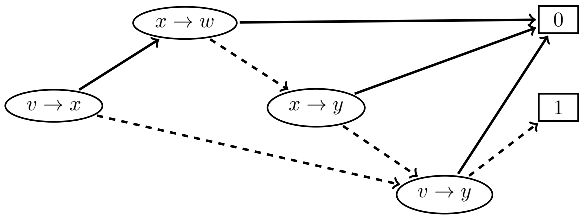

Fig. 2. A BDD for the repairs of Figure 1.

simpler than Definition 3 mainly because the former refers to only one consequence, while the later must consider a whole set of them. It is easy to see that for all α and all C, $\mathsf{rep}_{\{\alpha\}} = \neg\mathsf{pin}_\alpha$ and $\mathsf{rep}_C = \bigwedge_{\alpha\in C} \neg\mathsf{pin}_\alpha$.

Effective methods for constructing and dealing with pinpointing formulas have been studied and implemented for different representation languages; most notably for DLs [18]. The bottom line is that it is possible to efficiently construct a pinpointing formula for many different cases. In particular, one can even build a BDD encoding this formula for expressive representation languages.

As mentioned already, the pinpointing formula can be used to update the repair function to exclude one more consequence. Remember that, from each repair, we need to exclude every justification. That is, the new repair function should map a sub-KB $\mathcal{K}'$ to 1 if the original rep did so (i.e., $\mathsf{rep}(\mathcal{K}') = 1$), but the pinpointing formula did not ($\mathsf{pin}(\mathcal{K}') = 0$). If we see these functions as formulas, this means that rep' should be $\mathsf{rep} \wedge \neg\mathsf{pin}$.

Theorem 5. *Let $\mathcal{K}$ be a KB, rep a repair function w.r.t. a set of consequence C, and pin a pinpointing formula w.r.t. the consequence α. Then $\mathsf{rep}' := \mathsf{rep} \wedge \neg\mathsf{pin}$ is a repair function w.r.t. $C \cup \{\alpha\}$.*

Proof. Let $\mathcal{K}' \subseteq \mathcal{K}$. We need to show that $\mathsf{rep}'(\mathcal{K}') = 0$ iff there is a $\beta \in C \cup \{\alpha\}$ such that $\mathcal{K}' \models \beta$. We see that $\mathsf{rep}'(\mathcal{K}') = 0$ iff $\mathsf{rep}(\mathcal{K}') = 0$ or $\mathsf{pin}(\mathcal{K}') = 1$. The former holds iff there is a $\beta \in C$ such that $\mathcal{K}' \models \beta$, while the latter holds iff $\mathcal{K}' \models \alpha$. Hence, overall, we get the desired result. $\square$

This means that the repair functions can be iteratively constructed by conjoining the negations of the pinpointing formulas of the unwanted consequences that are encountered during the use of the KB. At any moment, we can try to extract some meaningful information out of this repair function. Before we describe the kind of information that can be extracted, and the methods for doing so, we note that the correctness of Theorem 5 depends fundamentally on the fact that the repair function captures all sub-KBs that do not entail the consequences in C, and not only the repairs (i.e., not only the subset-maximal ones). Figure 2 depicts a BDD for the repairs w.r.t. $\{v \to w, v \to y\}$ constructed this way.

Recall now that we are using, during production, a KB that is known to contain some errors. Still, until it is completely repaired, we want to be able to extract some meaningful information out of this KB. In particular, we would like to be able to derive consequences with a guarantee, or at least a hint, of correctness. This motivates the use of cautious and brave consequences.

Definition 6 (cautious, brave consequences). *Let $\mathcal{K}$ be a KB, C a set of consequences, and α a consequence. $\mathcal{K}$* cautiously entails *α w.r.t. C ($\mathcal{K} \models_c \alpha$) iff for every repair $\mathcal{K}'$ of $\mathcal{K}$ w.r.t. C it holds that $\mathcal{K}' \models \alpha$. $\mathcal{K}$* bravely entails *α w.r.t. C ($\mathcal{K} \models_b \alpha$) iff there exists a repair $\mathcal{K}'$ of $\mathcal{K}$ w.r.t. C such that $\mathcal{K}' \models \alpha$.*

Intuitively, a cautious consequence is guaranteed to hold regardless of which repair we choose, and hence we are certain that, after the errors have been fixed, this consequence will still hold. Thus, a user can safely use this consequence, which circumvents all the known errors. On the other hand, brave consequences only need to hold in at least one repair. Note that this could, in fact, be the correct repair, in which case this consequence would still hold after the KB is fixed. However, there is no guarantee that this will be the case. Although brave consequences are relatively weak, and do not preserve some basic logical closure properties (e.g., $\mathcal{K} \models_b a \rightarrow b$ and $\mathcal{K} \models_b b \rightarrow c$ does not imply that $\mathcal{K} \models_b a \rightarrow c$), they can be useful to know that the consequence is still possible after repairing. If this is a wanted consequence, we can use this information to guide the search for the right repair. Or, if it is an unwanted consequence, knowing that it is not bravely entailed allows us to avoid making an additional (but irrelevant) contraction step. We first see that brave consequences can be decided by operating over the repair function and the pinpointing formula.

Proposition 7. *Let $\mathcal{K}$ be a KB, rep a repair function w.r.t. C, and pin a pinpointing formula for α. $\mathcal{K}$ bravely entails α w.r.t. C iff rep does not entail $\neg\mathsf{pin}$.*

Proof. If $\mathcal{K} \models_b \alpha$, then there exists a repair $\mathcal{K}'$ w.r.t. C such that $\mathcal{K}' \models \alpha$. By definition, this means that $\mathsf{rep}(\mathcal{K}') = 1 = \mathsf{pin}(\mathcal{K}')$. So, it cannot hold that $\mathsf{rep} \Rightarrow \neg\mathsf{pin}$. Conversely, if $\mathcal{K} \not\models_b \alpha$, then for every repair $\mathcal{K}'$ we know that $\mathcal{K}' \not\models \alpha$. By monotonicity of the consequences, the same is true for all subsets of a repair. That is, for every $\mathcal{L} \subseteq \mathcal{K}$, if $\mathsf{rep}(\mathcal{L}) = 1$ then $\mathcal{L} \not\models \alpha$. In terms of the pinpointing formula, this means that if $\mathsf{rep}(\mathcal{L}) = 1$, then $\mathsf{pin}(\mathcal{L}) = 0$, or alternatively, $\neg\mathsf{pin}(\mathcal{L}) = 1$. Hence $\mathsf{rep} \Rightarrow \neg\mathsf{pin}$. □

Dealing with cautious consequences requires a slightly more complex analysis. To verify that a consequence is not cautiously entailed, it does not suffice to simply check whether there is a set accepted by rep that is rejected by pin (i.e., a set that avoids C and also rejects α). This is because rep also accepts all subsets of the repairs. In particular, the empty KB is such that $\mathsf{rep}(\emptyset) = 1$ and $\mathsf{pin}(\emptyset) = 0$, assuming that α and the consequences in C are not tautologies. Thus, we really need to be careful that the maximality of the repairs is taken into consideration.

Proposition 8. *Let $\mathcal{K}$ be a KB, rep_C a repair function w.r.t. C, and pin_α a pinpointing formula for α. $\mathcal{K}$ cautiously entails α w.r.t. C iff for every set $\mathcal{K}'$ satisfying $\mathsf{rep}_C \wedge \neg\mathsf{pin}_\alpha$ there exists some $\mathcal{L} \supset \mathcal{K}'$ that satisfies $\mathsf{rep}_C \wedge \mathsf{pin}_\alpha$.*

Proof. Suppose that $\mathcal{K} \models_c \alpha$, and let $\mathcal{K}'$ be such that satisfies $\mathsf{rep}_C \wedge \neg\mathsf{pin}_\alpha$. By definition of pin, $\mathcal{K}' \not\models \alpha$. Since α is cautiously entailed, $\mathcal{K}'$ cannot be a repair, but since $\mathsf{rep}_C(\mathcal{K}') = 1$, it should be a subset of a repair. Let $\mathcal{L}$ be any repair containing $\mathcal{K}'$. By construction, $\mathcal{L}$ satisfies $\mathsf{rep}_C \wedge \mathsf{pin}_\alpha$. Conversely, if $\mathcal{K} \not\models_c \alpha$, then there exists some repair $\mathcal{K}'$ such that $\mathcal{K}' \not\models \alpha$. Again, by construction $\mathcal{K}'$ satisfies $\mathsf{rep}_C \wedge \neg\mathsf{pin}_\alpha$. But since $\mathcal{K}'$ is a repair, for every $\mathcal{L} \supset \mathcal{K}'$ we know that $\mathsf{rep}_C(\mathcal{L}) = 0$. Hence, there exists no $\mathcal{L} \supset \mathcal{K}'$ that satisfies $\mathsf{rep} \wedge \mathsf{pin}$. □

For the sake of completeness, we consider a third kind of entailment that has been studied in the literature. A KB *IAR entails* the consequence α w.r.t. a set C iff α follows from the intersection of all repairs of $\mathcal{K}$ w.r.t. C [5]. In this case, the name IAR is an acronym for *intersection of all repairs*. Note that when dealing with IAR entailments, the structure of the repairs becomes irrelevant, and the only important information is which axioms appear in all of them. In fact, if we know the intersection of all the repairs, then IAR entailment becomes just a standard entailment over this sub-KB.

To solve IAR entailments, it suffices to identify the axioms that appear in the intersection. This can be done in two ways: either, on demand, whenever an IAR entailment test is required, or during the construction of the repair function, while preserving an additional data structure. We can see that the intersection of all repairs is the complement of the union of all justifications. Hence, as we construct rep, every time that we compute the set of all justifications, we can simply remove their union from the set of still active axioms. This approach becomes more efficient if many IAR entailment tests are expected, since the intersection is computed only once, and used for all tests.

We have so far focused on the problem of computing the repair function, which encodes all the repairs for a sequence of unwanted consequences from a KB, and how to use this function, together with the pinpointing formula, to perform meaningful reasoning tasks that avoid the known errors. In the next section, we consider a different problem; namely, helping a knowledge engineer to find the right repair (that is, identify which are the truly faulty axioms) from a potentially exponential class of options without having to analyse all of them independently, and trying to minimise their effort.

4 Choosing the Right Repair

So far we have worked under the assumption that a user of the KB, while capable of detecting errors in the consequences observed, is not knowledgeable enough—or is not authorised—to definitely fix the KB. As we have seen, there may exist many different repairs, and each of them is a potential solution to get rid of all the errors. However, not all of them are equally valid. Suppose for example that our KB refers to a concept hierarchy, where $a \to b$ means that every a is a b. In this setting, we can consider the KB that has two axioms $\mathsf{human} \to \mathsf{mammal}$ and $\mathsf{mammal} \to \mathsf{plant}$. A consequence of this KB is that every human is a plant, which is obviously unwanted. Each of the two axioms $\mathsf{human} \to \mathsf{mammal}$ and $\mathsf{mammal} \to \mathsf{plant}$ is a repair, but clearly, only the former one is a valid repair w.r.t. the domain of the KB. The task of the knowledge engineer is to identify this right repair from the sea of all potential repairs.

It should be clear that it is completely infeasible for the knowledge engineer to explore all the repairs one by one, and decide which is the correct one. First, there are many such repairs, making this an overwhelming task for a human resource. Moreover, each repair is expected to be large and, as in the original KB, it may be very hard to spot an error in an axiom hidden within thousands of other axioms. Our proposal is to make the knowledge engineer verify only one axiom at a time, rather than a full repair. Once again, one could think of checking each axiom in the KB, which is clearly infeasible for very large KBs. Instead, we select first those axioms that are more likely to lead to the correct repair in the least amount of steps, regardless of the answer provided by the knowledge engineer. More precisely, we try to find an axiom that belongs to half of all repairs, or as close as possible to that.

Definition 9 (cut axiom). *Let $\mathcal{K}$ be a KB, and $\mathcal{R}$ the set of all repairs of $\mathcal{K}$ w.r.t. a set of consequences C. Given an axiom $\alpha \in \mathcal{K}$, we define the sets*

$$\mathcal{R}_\alpha^+ := \{\mathcal{L} \in \mathcal{R} \mid \alpha \in \mathcal{L}\},$$
$$\mathcal{R}_\alpha^- := \{\mathcal{L} \in \mathcal{R} \mid \alpha \notin \mathcal{L}\}.$$

The axiom α is called a cut axiom *iff for every $\beta \in \mathcal{K}$ it holds that*

$$|\mathcal{R}_\alpha^+| - |\mathcal{R}_\alpha^-| \leq |\mathcal{R}_\beta^+| - |\mathcal{R}_\beta^-|.$$

The idea behind the cut axiom is that, by verifying its correctness, we can immediately cut the search space (almost) in half. Specifically, if α is correct, then we know that the right repair is among $\mathcal{R}_\alpha^+$, and if it is wrong, we should focus only on $\mathcal{R}_\alpha^-$. Hence, the first question is how to compute such an axiom. Unfortunately, it turns out that deciding whether an axiom is a cut axiom is coNP-hard already for the very simple representation language that we are using as an example, and only one unwanted consequence is known.

Theorem 10. *Given a graph $\mathcal{K}$, an edge $v \to w \in \mathcal{K}$, and an unwanted reachability entailment $x \to y$, deciding whether $v \to w$ is a cut axiom is co*NP*-hard.*

Proof. We prove this by a reduction from the following coNP-hard problem: is there a repair $\mathcal{L}$ for $\mathcal{K}$ w.r.t. $x \to y$ such that $v \to w \notin \mathcal{L}$? [12]. Let $n := |\mathcal{K}|$. We notice that there can exist at most 2^{n-1} repairs that contain $v \to w$ and at most 2^{n-1} that do not contain this axiom. Assuming w.l.o.g. that there are at least two repairs, we construct the new KB $\mathcal{M}$, which extends $\mathcal{K}$ by adding $2n$ new vertices $z_1, \ldots, z_{2n}$, and the edges

$$E' := \{v \to z_i, z_i \to w \mid 1 \leq i \leq 2n\}.$$

Clearly, the size of $\mathcal{M}$ is linear on the size of $\mathcal{K}$.

This new KB has the following property. For every repair $\mathcal{K}'$ of $\mathcal{K}$ w.r.t. $x \to y$, (i) if $v \to w \in \mathcal{K}'$ then $\mathcal{K} \cup E'$ is the only repair of $\mathcal{M}$ w.r.t. $x \to y$ that contains $\mathcal{K}'$ (that is, there is a one-to-one correspondence between the repairs of $\mathcal{K}$ and of $\mathcal{M}$ that contain $x \to y$); and (ii) if $v \to w \notin \mathcal{K}'$, then there exist 2^{2n}

different repairs of $\mathcal{M}$ w.r.t. $x \to y$ that contain $\mathcal{K}'$; in particular, exactly 2^{2n-1} of those repairs contain the axiom $v \to z_1$.

In particular, if $\mathcal{K}$ has m repairs containing $v \to w$ and ℓ not containing this axiom, then $\mathcal{M}$ will have $m + \ell \cdot 2^{2n-1}$ repairs containing $v \to z_1$ and $\ell \cdot 2^{2n-1}$ repairs not containing the axiom. Moreover, every other axiom of $\mathcal{K}$ will appear in at most 2^{n-1} repairs of $\mathcal{M}$, and all the axioms in E' will be in exactly the same number of repairs as $v \to z_1$. Thus, $x \to z_1$ is a cut axiom w.r.t. $\mathcal{M}$ iff $\ell \geq 1$; that is, iff there is at least one repair of $\mathcal{K}$ that does not contain $v \to w$. $\square$

What this theorem shows is that it is in general hard to detect which axiom is the best option to verify first in order to guarantee a reduction of the search space. In fact, it is very unlikely that the exact complexity coincides with this lower bound. It is known that counting the number of justifications, and the number of justifications to which an axiom belongs are #P-complete problems [13]. Although such hardness results have not been shown, to the best of our knowledge, also for the problem of counting repairs, the duality between both problems strongly suggests that this is the case as well.

On the other hand, notice that not all knowledge representation languages are as inexpressive as our example graph language. Indeed, even if we restrict to decidable cases, there are some mainstream languages where reasoning is at least NExpTime-hard [2]. In such cases (and indeed in any language where reasoning is PSpace-hard), finding a cut axiom is as expensive, in terms of computational complexity, as merely deciding whether a consequence follows.

Suppose that we have found a cut axiom α. The next step is to use it to prune the search space in a way that the process can be iterated until the right repair is found. After we propose this axiom to the knowledge engineer, they will respond either that the axiom is (i) correct, or that it is (ii) incorrect. In the second case, we know that α cannot appear in the right repair. Thus, we simply eliminate it from the KB $\mathcal{K}$. In practical terms, this means updating the repair function to ignore α, considering it as being always false. If the repair function is expressed as a BDD, this operation corresponds simply to removing the nodes representing α from the diagram, and updating every edge pointing to those nodes to now point to their lower children—that is, to the node reached when α is evaluated to false at that node.

In the case (i)—i.e., when α is marked as correct—then we should ignore all repairs that do not contain α, and focus only on those that include it. In principle, we could do as before and remove all α nodes, but now assuming that it is always true within the repair function. However, we want to preserve the knowledge that α must appear in all repairs. Hence, rather than removing the variable from the formula, we enforce the repair function to exclude any set that does not contain α. In a BDD, this is achieved by substituting the lower branch of every α node with the 0-terminal.

Importantly, after these operations the resulting structure is still a repair function, but which now accepts only those repairs that comply with the information provided by the knowledge engineer. Hence, we can repeat the process, reducing at each step the total number of repairs remaining. Note that once that an axiom has been analysed by the knowledge engineer, it will never be proposed by the system again. Hence, in the worst case, it will need as many tests as there are axioms in the KB; i.e., it is not worse than the naïve approach of testing all axioms in order. In reality, far less axioms will be tested. First, notice that all axioms that appear either in all repairs, or in none of them will never be proposed by the method: the only way they become cut axioms is if there is only one repair left, in which case the process has already finished. Second, every time that we test an axiom, we reduce the class of remaining repairs, which in turn increases both, the class of axioms that belong to all repairs, and the class of axioms that belong to none. Thus, more axioms will be excluded from testing.

5 Conclusions

We have proposed a methodology for helping knowledge engineers to make decisions in relation to faulty knowledge bases. The premise of this work is based on the idea of preserving all the information regarding all the possible ways in which errors, which may be detected while a KB is in use, may be avoided. For this purpose, we propose to store a Boolean function, which accepts only those sub-KBs that are free of all known errors. If new faults are encountered, this Boolean function—called the repair function—can be updated using known techniques from the area of consequence explanation (or axiom pinpointing).

The importance of the repair function is that it can be used to decide different properties of the consequences of the original KB. Specifically, one can verify whether a consequence can still be derived from any possible repair (cautious reasoning), or from at least one of them (brave reasoning). As we have no precise notion *a priori* of the actual repair that will be obtained after the knowledge engineer has verified the correctness of the axioms, brave consequences have also a place as ones that can potentially remain.

In addition, we introduced the notion of a cut axiom, which is the one that appears in half the repairs, or as close to that as possible. Although finding a cut axiom is computational hard, it is an effective tool for guiding the search for the right repair that fixes all the faults encountered. The computational effort needed to find the cut axiom is a good investment in reducing the human cost of verifying the correctness of a repair. We notice that a full repair plan (e.g. a decision tree) can be computed offline before any intervention by the human expert, so that they are given a sequence of questions without delay.

In summary, our proposal is to preserve all the information of the potential repairs during deployment to allow users to make informed decisions that sidestep known errors, and to use this information to minimise the number of queries made to a human expert to identify the actual correct knowledge base that should be used in the following publishing cycle.

One line of future research is to try to extend the notion of a cut axiom into a *cut consequence*; that is, using potentially complex consequences to better separate the space of repairs. The challenge in this direction is to adequately restrict the class of consequences that may be used, and to develop an effective method that does not need to enumerate them all. Another goal is to improve the notion of a cut axiom so that it also takes into account the improvements on the following steps; for instance, to consider those that minimise the average number of questions that need to be asked overall. Finally, we will implement a prototypical system for a well-known knowledge representation language, and test the effectiveness of our methods on realistic KBs.

References

1. Apt, K.: Principles of Constraint Programming. Cambridge University Press, New York, NY, USA (2003)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, second edn. (2007)
3. Baader, F., Peñaloza, R.: Automata-based axiom pinpointing. In: Proc. IJCAR 2008. LNCS, vol. 5195, pp. 226–241. Springer (2008)
4. Baader, F., Peñaloza, R.: Axiom pinpointing in general tableaux. Journal of Logic and Computation **20**(1), 5–34 (2010). https://doi.org/10.1093/logcom/exn058
5. Bienvenu, M., Rosati, R.: Tractable approximations of consistent query answering for robust ontology-based data access. In: Proc. IJCAI'13. pp. 775–781. AAAI Press (2013)
6. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press (2009)
7. Darwiche, A.: SDD: A new canonical representation of propositional knowledge bases. In: Proc. IJCAI'11. pp. 819–826. IJCAI/AAAI (2011)
8. Darwiche, A., Pearl, J.: On the logic of iterated belief revision. In: Proc. TARK '94. pp. 5–23 (1994)
9. Drechsler, R., Becker, B.: Binary Decision Diagrams - Theory and Implementation. Springer (1998)
10. Ludwig, M., Peñaloza, R.: Error-tolerant reasoning in the description logic EL. In: Proc. JELIA 2014. LNCS, vol. 8761, pp. 107–121. Springer (2014)
11. Peñaloza Nyssen, R.: Axiom pinpointing in description logics and beyond. Ph.D. thesis, Technische Universität Dresden, Germany (2009)
12. Peñaloza, R.: Inconsistency-tolerant instance checking in tractable description logics. In: Proc. RuleML+RR 2017. LNCS, vol. 10364, pp. 215–229. Springer (2017)
13. Peñaloza, R., Sertkaya, B.: Understanding the complexity of axiom pinpointing in lightweight description logics. Artificial Intelligence **250**, 80–104 (2017)
14. Peñaloza, R., Thuluva, A.S.: Iterative ontology updates using context labels. In: Proc. JOWO 2015. CEUR Workshop Proceedings, vol. 1517. CEUR-WS.org (2015)
15. Price, C., Spackman, K.: Snomed clinical terms. British Journal of Healthcare Computing and Information Management **17**(3), 27–31 (2000)
16. Reiter, R.: A theory of diagnosis from first principles. AIJ **32**(1), 57–95 (1987)
17. Shannon, C.E.: The synthesis of two-terminal switching circuits. Bell System Technical Journal **28**(1), 59–98 (1949)
18. Zese, R., Bellodi, E., Riguzzi, F., Cota, G., Lamma, E.: Tableau reasoning for description logics and its extension to probabilities. AMAI **82**(1-3), 101–130 (2018)