ABC Repair System for Datalog-like Theories

ABC Repair System for Datalog-like Theories

Xue Li, Alan Bundy and Alan Smaill
CISA, School of Informatics, University of Edinburgh, 10 Crichton St, Edinburgh, UK
Xue.Lee@ed.ac.uk, A.Bundy@ed.ac.uk, A.Smaill@ed.ac.uk

Keywords: Theory repair, Automated reasoning, Knowledge representation, Abduction, Belief revision, Conceptual change, Datalog.

Abstract: This paper aims to develop a domain-independent system for repairing faulty Datalog-like theories by combining three existing techniques: abduction, belief revision and conceptual change. Accordingly, the proposed system is named the ABC repair system. Given an observed assertion and a current theory, abduction adds axioms which represent the simplest and most likely explanation. Belief revision incorporates a new piece of information which conflicts with the input theory by deleting axioms. Conceptual change uses the reformation algorithm for blocking unwanted proofs or unblocking wanted proofs. The former two techniques change an axiom as a whole, while reformation changes the language in which the theory is written. These three techniques are complementary: abduction adds new axioms, belief revision deletes conflicting axioms, while reformation changes the language of the theory. But they have not previously been combined into one system. We are working on aligning these three techniques in the ABC repair system, which is capable of repairing logical theories with better quality than individual techniques. Datalog is used as the underlying logic of theories in this paper, but the proposed system has the potential to be adapted to theories in other logics.

PRELIMINARY

In this paper, a Datalog-like theory is represented as $\mathbb{T}$, and $\mathbb{T}'$ is a corresponding repaired theory. α and β are Datalog-like axioms written in the language of $\mathbb{T}$. All constants and predicates start with an upper-case letter, while variables start with a lower-case one. The unary function $\mathcal{N}$ calculates the number of the elements in its argument. The argument of $\mathcal{N}$ is a set, e.g. $\mathcal{N}(\mathbb{T})$ is the number of the axioms in $\mathbb{T}$.

1 INTRODUCTION

In a knowledge base, logical theories need to evolve to keep up with a changing environment and to correct faulty representations. The need for evolution could be formalised as the conflicts between what the user knows and what the logical theories derive. By repairing logical theories, these conflicts can be fixed, and then the logical theories evolve for absorbing users' knowledge. A user could be an automated agent or a human.

Datalog is a declarative logic programming language in FOL, which has resurged in the database community in recent years (Bellomarini et al., 2018). In comparison with first-order logic (FOL), Datalog excludes negations, function symbols, and existential quantification. Assertions and rules in Datalog are formalised as Horn Clauses, which should satisfy conditions that each assertion does not contain any variables, and each variable which occurs in the head of a rule also occurs in the body of the same rule (Ceri et al., 1989). There are different variants of Datalog. This paper restricts to the basic Datalog introduced above. For a uniform representation in this paper, we use the implication symbol $\Longrightarrow$ rather than the traditional symbol, :- in a Datalog-like theory.

Definition 1 (Datalog Language)**.**

$$
\begin{aligned}
Term &:= Constant \mid Variable \\
Propositions &:= Predicate(Term_1, \ldots, Term_n) \\
Rule &:= Proposition_1 \wedge \ldots \wedge Proposition_n \Longrightarrow Proposition \\
Goal &:= Proposition_1 \wedge \ldots \wedge Proposition_n \Longrightarrow
\end{aligned}
$$

where $n \in \mathbb{N}$*, i.e., it might be* 0*.*

Although the restriction of Datalog reduces the expressive power of the logic, it brings significant advantages including that deduction is decidable (Pfenning, 2006); Prolog unification is sufficient, and the reformation algorithm is greatly simplified, as will be introduced in §2.2.1.

An example of a Datalog-like theory is the Swan theory as below. It has four axioms saying that German is part of Europe, all European swans are white,

and the swan named Bruce is a German swan. One theorem is that Bruce is white. Imagine the user observes the fact that Bruce is black. In this scenario, the theory is faulty so that it needs to be repaired.

```
A1. German(x) ⇒ European(x).
A2. European(x) ∧ Swan(x) ⇒ White(x).
A3. German(Bruce).
A4. Swan(Bruce).
```

2 ABC REPAIR SYSTEM

The ABC system repairs faulty Datalog-like theories automatically or partially automatically. Figure 1 gives the main components of the system. The inputs given by the user are a Datalog-like theory ($\mathbb{T}$) and a *preferred structure* ($\mathbb{PS}$) which will be defined in §2.1.1. The output is a set of repaired theories. Because each original repair technique could have multiple repairs to one fault, it is also normal that the combination repair mechanism generates more than one repair solution and then outputs a set of repaired theories rather than single repaired theory.

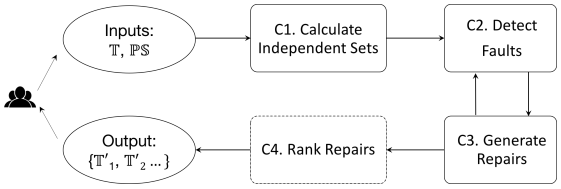

Figure 1: The components (C1-C4) of the ABC repair system: $\mathbb{T}$ is a Datalog-like theory; $\mathbb{PS}$ is a preferred structure defined in §2.1.1; The output is a set of repaired theories.

C1 calculates a minimal set of axioms by pruning redundancy, which reduces the search space of both fault detection and repair generation. C2 and C3 are central parts, which will be discussed in details in §2.1 and §2.2. Generally speaking, when a fault is detected by C2, its proof/proofs will be provided to C3, based on which, C3 generates repairs by combining abduction, belief revision and reformation. This process will repeat until there are no faults left. In the end, all repaired theories will be ranked by C4. Ideally, best-repaired theories could be highlighted for users. As a part of future work, C4 is denoted in the imaginary line in Figure 1.

2.1 Fault to be Repaired

This section discusses what faults are to be repaired and how to detect them based on automated reasoning.

2.1.1 Fault Definition

Common faults of a logical theory are inconsistency and incompleteness. Although a Datalog-like theory is guaranteed to be consistent by never proving a negative statement, consistency is an underlying requirement of a repaired theory in our system. However, completeness, which requires a theory to prove all the possible sentences or their negations in the signature, may be too strong a requirement. Instead of incompleteness, we are going to focus on the propositions which the user insists an object theory to prove or not to prove. The *preferred structure* is defined as below.

Definition 2 (Preferred Structure)**.** *A preferred structure ($\mathbb{PS}$) is a structure over the language of a logical theory ($\mathbb{T}$); $\mathbb{PS}$ describes the user's intention that a* faithful $\mathbb{T}$ *should follow by giving preferred base sets:*

True Set ($\mathcal{T}(\mathbb{PS})$): *Set of the ground propositions which should be provable by $\mathbb{T}$. These propositions are called the preferred propositions.*

False Set ($\mathcal{F}(\mathbb{PS})$): *Set of the ground propositions which should not be proved by $\mathbb{T}$. These propositions are called the violative propositions*

Obscure Set ($O(\mathbb{PS})$): *Set of the ground propositions which are neither in $\mathcal{T}(\mathbb{PS})$ nor $\mathcal{F}(\mathbb{PS})$.*

Contradictory Set ($\mathcal{C}(\mathbb{PS})$): *Set of the ground propositions which are both in $\mathcal{T}(\mathbb{PS})$ and $\mathcal{F}(\mathbb{PS})$.*

$\mathbb{PS}$ is a guideline for determining the faithfulness of a theory $\mathbb{T}$ according to user's view. As shown in Figure 1, $\mathbb{PS}$ is a required input for the ABC system.

Based on the definition, the last two sets are decided by the first two. Therefore, in the implementation, only $\mathcal{T}(\mathbb{PS})$ and $\mathcal{F}(\mathbb{PS})$ need to be given explicitly. In the swan example, the $\mathbb{PS}$ for formalising the user's knowledge that Bruce is black so that it cannot also be white is:

```
The true set: {Black(Bruce)}
The false set: {White(Bruce)}
```

An object theory should follow its $\mathbb{PS}$ by proving the preferred propositions, while not proving the violative propositions. This makes the theory correct and useful to the user.

$$\forall \alpha,\ \alpha \in \mathcal{T}(\mathbb{PS}),\ \mathbb{T} \vdash \alpha \tag{1}$$

$$\forall \beta,\ \beta \in \mathcal{F}(\mathbb{PS}),\ \mathbb{T} \nvdash \beta \tag{2}$$

Notice that $\mathcal{C}(\mathbb{PS})$ should be empty. Otherwise, the preferred structure is self-contradictory, to which no consistent theory could be faithful. If there is a proposition in $\mathcal{C}(\mathbb{PS})$, a warning would be given and the

process of our system has to stop. In this case, the user has to empty $\mathcal{C}(\mathbb{PS})$ before giving the $\mathbb{PS}$ as the input of the ABC system.

Given a preferred structure with empty $\mathcal{C}(\mathbb{PS})$, the possible faults of a logical theory are incompatibility and insufficiency.

Definition 3 (Incompatible and Insufficient)**.**

- *An incompatible theory with respect to the preferred structure is one which has a theorem that is in the false set:* $\exists\beta,\ \beta \in \mathcal{F}(\mathbb{PS}),\ \mathbb{T} \vdash \beta$.
- *An insufficient theory with respect to the preferred structure is one in which a proposition that is in the true set, but is not a theorem of the theory:* $\exists\alpha,\ \alpha \in \mathcal{T}(\mathbb{PS}),\ \mathbb{T} \nvdash \alpha$.

The defined faults represent the conflicts between the input theory and the user's intention. One of the applications is when the user verifies a theory by experiments. For example, a professor of medicine is studying how a medicine affects a human. At first, he investigated relevant materials, so he had built a set of axioms in his theory. Then he ran a series of experiments to evaluate his theory. In this scenario, the result of his experiments could be formalised as a preferred structure, based on which the conflicts between his theory and his experimental results correspond to incompatibility and insufficiency as defined above.

In the swan example, the original theory proves that Bruce is white, which is an incompatibility, and not prove that Bruce is black, which is an insufficiency. The definition of the preferred structure and the faults defined based on it can reflect the conflicts between a theory and empirical evidence correctly.

In conclusion, the faults of an object theory to be repaired in this paper are insufficiency and incompatibility, defined based on a preferred structure representing the user's opinion.

2.1.2 Fault Detection

This section shows how to detect the faults of a Datalog-like theory by employing selected literal (SL) resolution.

Detecting insufficiency or incompatibility is to check the provability of a target proposition from a Datalog-like theory. The inference rule, SL resolution (Kowalski and Kuehner, 1971) can do this detection, which follows the principles:

- Select one of the most recently introduced literals to be resolved upon.
- Resolve the selected literal with an input clause.

SL resolution is not only sound and complete (Gallier, 2003), but also decidable for Datalog-like theories (Pfenning, 2006) so that proofs can always be detected if there are any.

For reducing the search space, proof discovery process starts with resolving a goal, with an input clause whose head is complementary to the goal. As an axiom in Datalog-like theory is a Horn clause, which has at most one positive literal, the most recently introduced literals of a nonempty resolvent could only be a disjunction of negative literals, called sub-goals here. Then we continually resolve the first sub-goal, with an input clause. Again, either an empty clause or a clause of a disjunction of negative literals would be the result. Repeat this process until no resolution step is available. If it ends up with an empty clause, refutation occurs, which means that the input theory proves the goal clause.

For detecting incompatibility, the negation of a violative proposition in $\mathcal{F}(\mathbb{PS})$ is set as the initial goal for refutation based on the input theory. Every violative proposition, in turn, is checked in the same way. Incompatibility is detected if the input theory proves a violative proposition. $\perp$ will represent false.

$$\mathbb{T} \text{ is incompatible wrt } \beta, \text{ iff } \mathbb{T}' \vdash \perp, \quad \text{where } \mathbb{T}' = \mathbb{T} \cup \{\neg\beta\},\ \beta \in \mathcal{F}(\mathbb{PS}). \quad (3)$$

For example, Figure 2 shows how the Swan theory proves the violative proposition of the $\mathbb{PS}$, *White(Bruce)*. There are four proof steps in total. In proof step 1, the initial goal is the negation of the violative proposition, *White(Bruce)*$\Rightarrow$, which is resolved with the head of axiom A2, *White(x)*, by substituting *Bruce* for *x*. In the same way, *European* and *German* are resolved in proof step 2 and 3 respectively. In proof step 4, the empty clause is derived, which means that the input Swan theory proves *White(Bruce)*, so the Swan theory is incompatible.

As for a sufficient theory, all the preferred propositions should be logical consequences of the object theory. Similar to detecting incompatibility, preferred propositions need to be checked one by one. By negating a preferred proposition into a goal, the theory can be concluded to be sufficient concerning the preferred proposition if and only if resolving the goal with the input theory results in an empty clause. Otherwise, the object theory is insufficient. Repeat the process until all the preferred propositions are checked.

$$\mathbb{T} \text{ is insufficient wrt } \alpha, \text{ iff } \mathbb{T}' \nvdash \perp, \quad \text{where } \mathbb{T}' = \mathbb{T} \cup \{\neg\alpha\},\ \alpha \in \mathcal{T}(\mathbb{PS}) \quad (4)$$

At each resolution in the proof, there is always one parent being an input clause. When generating repairs, we can directly change the parent which is an input clause with no traceback needed.

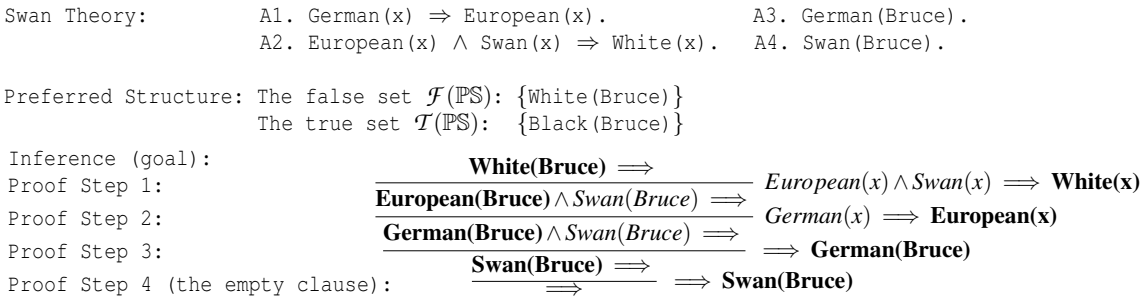

Figure 2: Proving White(Bruce): proof steps of the incompatibility of the Swan theory.

2.2 Repair Generation

In the ABC repair system, repairs are generated based on the proofs of faults: incompatibility and insufficiency. The desired repairs for each fault are shown in Table 1: incompatibility could be repaired by blocking all unwanted proofs of each violative proposition, and insufficiency could be repaired by unblocking a wanted proof of each preferred proposition. Blocking a proof could be done by breaking any proof step of it, while unblocking a proof requires to build all necessary proof steps. As the decision of which axiom(s) or predicate(s) to be changed is not unique, usually, the core task here is to avoid unnecessary information loss by making minimal changes (Gärdenfors, 1992).

Table 1: The target of repairing a faulty theory $\mathbb{T}$.

	Incompatibility	Insufficiency
Fault	$\exists\beta, \beta \in \mathcal{F}(\mathbb{PS}), \mathbb{T} \vdash \beta$	$\exists\alpha, \alpha \in \mathcal{T}(\mathbb{PS}), \mathbb{T} \nvdash \alpha$
Target	$\mathbb{T}' \nvdash \beta$	$\mathbb{T}' \vdash \alpha$
Core Task	Blocking all proofs of β.	Unblocking one proof of α.
Method	Break one proof step in each proof.	Build all necessary proof steps.

Abduction, belief revision and reformation are three technical candidates for generating repairs. Different techniques work in different scenarios.

2.2.1 Repair Techniques in the ABC System

Original Abduction. The underlying reasoning of abduction is the inference to the best explanation. Given an observation, abduction seeks for its explanation (Cox and Pietrzykowski, 1986). If the observation is not a logical consequence of the original theory, abduction will add the explanation as an axiom, which could be an assertion or a rule. With no extra information, the solution of abduction is not unique because there could be different ways of proving the observation.

Abduction in the ABC System. Regarding a preferred proposition in $\mathcal{T}(\mathbb{PS})$ as the observation, abduction repairs insufficiencies by adding axioms. Abduction could directly add a preferred proposition. However, directly adding everything that is true would result in a clumsy theory with lower quality, e.g., a theory of gases that just added all observations of gas pressure and volume is inferior to one that includes Boyle's law. Therefore, it is better to add a rule which proves not only the targeted preferred proposition but also several other axioms and/or other preferred proposition, while not proving any proposition in $\mathcal{F}(\mathbb{PS})$.

Original Belief Revision. The underlying reasoning of belief revision is deduction. Belief revision works on incorporating new information which is inconsistent with the original knowledge base (Gärdenfors, 1992). The inconsistent new information is usually represented as a new belief, e.g. β. Then revision function adds the new belief and deletes old ones to make the revised theory consistent, which is blocking all proofs of $\neg\beta$. From the view of proofs, the task of a revision function is to break the unwanted proofs with minimal changes.

Belief Revision in the ABC System. To repair the incompatibility, the ABC system uses belief revision to block the proofs of the propositions in $\mathcal{F}(\mathbb{PS})$. For blocking these unwanted proofs, belief revision deletes axioms while making minimal changes.

Original Conceptual Change. Conceptual change uses the reformation algorithm, which is triggered by the reasoning failure that $\mathbb{T}$ proves an unwanted assertion or fails in proving a wanted one (Bundy and Mitrovic, 2016). The underlying reasoning of reformation is deduction, and originally in FOL. By changing the language of a theory, reformation inverts the outcome of the targeted unification, which contributes to blocking or unblocking proofs. The repair types of reformation include changing the name or the arity of a predicate, changing the name of a constant or chang-

ing a constant into a term containing a variable.

Conceptual Change in the ABC System. Because the ABC system is based on Datalog, reformation is much simpler than that for FOL. For example, differing arity of a predicate overloads the predicate, and changing a constant into a term containing a variable would introduce a function. These two kinds of repairs are illegal in the circumstance of Datalog so that they are abandoned in the ABC system. In this paper, we use the left-hand side term of each resolution to represent a goal/sub-goal, while the right is the input clause side. Table 2 gives reformation repairs on different types of resolution problems in the ABC system, in which the changes are underlined. R1 replaces $P_2(\overrightarrow{u}^m)$ by $P_1(\overrightarrow{t}^n)$ for making the originally failed proof step between $P_1(\overrightarrow{t}^n)$ and $P_2(\overrightarrow{u}^m)$ successful. The remaining repairs are about making the originally successful proof steps fail: R2 renames a predicate on the input clause side, R3 changes one argument of the predicate from the input clause side, and R4 increases arity by 1, and then adds distinguished constants as arguments to predicates on both sides. Because R4 changes the arity of a predicate, it needs to be propagated to all instances of the modified predicate. When the resolution is between a constant goal and a variable from input clause side, it can always succeed unless the variable is instantiated by another constant as in R5.

Considering the definition of the preferred structure, reformation is capable of repairing both insufficiency (R1) and incompatibility (R2-R5) by changing the language in which the theory is written.

Table 2: Repairs of reformation for inverting the outcome of proof step in the ABC system. P is a predicate; t and u are either constants or a variables; $n \geq 0, m \geq 0$; $1 \leq i \leq n$; P refers to a constant when its arity is 0; '=' represents unifying, of which goal/sub-goal is on the left-hand side, and input clause on the right.

Resolution Problem	Reformation Repair
$P_1(\overrightarrow{t}^n) \neq P_2(\overrightarrow{u}^m)$	R1. Replace input side by the other: $P_1(\overrightarrow{t}^n) = \underline{P_1(\overrightarrow{t}^n)}$.
$P(\overrightarrow{t}^n) = P(\overrightarrow{u}^n)$	R2. Rename input side P: $P(\overrightarrow{t}^n) \neq \underline{P'}(\overrightarrow{u}^n)$.
	R3. Rename one argument in $\overrightarrow{u}^n$, e.g., let $t_i \neq \underline{u'_i}$: $P(\overrightarrow{t}^n) \neq P(\underline{\overrightarrow{u'}^n})$.
	R4. Add distinguished arguments C, C': $P(\overrightarrow{t}^n, \underline{C}) \neq P(\overrightarrow{u}^n, \underline{C'})$.
$C = x$	R5. Instantiate x to C': $C \neq \underline{C'}$.

All Repair Candidates in the ABC System. Apart from changing language by reformation, or adding or deleting an axiom as a whole by abduction and belief revision respectively, a rule can be changed by adding or deleting a precondition (McNeill and Bundy, 2007). A precondition needs to be deleted when a preferred proposition should be derived based on a rule, but it is not because of the improperly strict precondition. In this case, the repair of deleting the precondition could be classified as a variant abduction, which unblocks a wanted proof. Similarly, the precondition which can block a rule from proving a violative proposition could be added to the rule, which could be classified as a variant of belief revision that blocks unwanted proofs. In conclusion, all repair candidates in the ABC system are summarised in Table 3.

Table 3: Technique candidates for repairing faults.

	Incompatibility	Insufficiency
Tech.	Belief revision: deletes axioms; adds unprovable preconditions to a rule.	Abduction: adds an assertion/ rule; deletes unprovable preconditions from a rule.
	Reformation: changes the language of $\mathbb{T}$.	

2.2.2 Combination of Techniques

The candidate techniques discussed in the last section are complementary. Combining them allows us to fix faults with new repairs. Due to the minimal change principle, we do not combine techniques in repairing a single proof step, because each original technique alone is sufficient already. Now, we can list all possible repairs for one proof step when tackling incompatibility in Table 4, and Table 5 gives all possible repairs for insufficiency.

It is fairly common that several faults are involved in one theory (McNeill and Bundy, 2007). The ABC system combines different techniques when there are multiple problematic proof steps. These problematic proof steps are detected in turn as errors: after repairing one error, a new one is detected. Here errors could be the faults: insufficiency and incompatibility based on a preferred structure; or the issues against the restriction of Datalog, or some heuristics, e.g., prefer not to change the equality predicate. Except for the original existing errors, there could be two kinds of introduced errors during a repair process:

1. New introduced errors. A repair may introduce a error which did not exist previously. For example, a

Table 4: Incompatibility repairs: block a proof by breaking one proof step of it in the ABC system. All symbols mean the same as in Table 2.

Resolution Problem	(Technique Abbr.) Repair
$P(\overrightarrow{t}^n) = P(\overrightarrow{u}^n)$	(R) $P(\overrightarrow{t}^n) \neq \underline{P'}(\overrightarrow{u}^n)$.
	(R) $P(\overrightarrow{t}^n) \neq P(\underline{\overrightarrow{u}^n})$.
	(R) $P(\overrightarrow{t}^n, \underline{C}) \neq P(\overrightarrow{u}^n, \underline{C'})$.
	(B) Delete the axiom where $+P(\overrightarrow{u}^n)$ came from.
	(B) Add an unprovable precondition $-Q(\overrightarrow{t}^n)$ to the axiom at either side.
$C = x$	(R) Instantiate x to C'.

Table 5: Insufficiency repairs: unblock a proof by building all necessary proof steps in the ABC system. All symbols mean the same as in Table 2.

Resolution Problem	(Technique Abbr.) Repair
$P_1(\overrightarrow{t}^n) \neq P_2(\overrightarrow{u}^m)$	(R) $P_1(\overrightarrow{t}^n) = \underline{P_1(\overrightarrow{t}^n)}$.
	(A) Delete $-P_1(\overrightarrow{t}^n)$, a precondition from its axiom.
	(A) Add a rule which proves $+P_1(\overrightarrow{t}^n)$.
	(A) Add the assertion: $P_1(\overrightarrow{t}^n)$.

predicate in an unwanted proof is changed, if it is also necessary for proving a preferred proposition, then this repair causes newly introduced insufficiency.

2. Recurred errors. A repair may affect a previous one, especially when these two repairs are provided by different algorithms, which could be difficult to avoid. For example, an axiom is added by abduction, and it could be changed by reformation later on. In this case, developing a set of heuristics is necessary, which is a part of future work.

We need to continue repairing until no error remains, and then a final solution could be generated completely. By applying three candidate techniques in parallel for each error, we can get all possible repair combinations, as shown in Figure 3. For example, if belief revision deletes an axiom, after which reformation changes the language of the theory to tackle another error, then the final repaired theory is generated by combining belief revision and reformation.

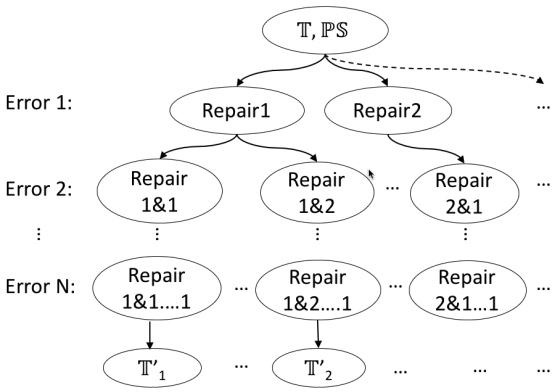

Figure 3: Combining repairs when repairing multiple errors. $\mathbb{T}'_1, \mathbb{T}'_2$ refer to repaired theories.

2.2.3 Repair Postulates

Resembling the idea of the AGM postulates for belief revision (Gärdenfors, 1992), several postulates are formulated in this section. The ABC repair system should generate repairs which satisfy these postulates. The underlying motivation is that we want to generate reasonable combination repairs and make a minimal change. All the information conveyed by axioms are seen as not gratuitous so that we do not want any unnecessary informational loss. On the other hand, we do not want to add extra axioms unless it is necessary.

1. $\exists \mathbb{T}'$, iff $\mathcal{T}(\mathbb{PS}) \cap \mathcal{F}(\mathbb{PS}) = \emptyset$
   An input theory can be repaired, if and only if the preferred structure is not self-contradictory. In other words, for achieving a faithful theory, the preferred structure must not be self-contradictory.
2. $\mathbb{T}' = \mathbb{T}$, iff
   $\forall \alpha,\ \alpha \in \mathcal{T}(\mathbb{PS}),\ \mathbb{T} \vdash \alpha; \forall \beta,\ \beta \in \mathcal{F}(\mathbb{PS}),\ \mathbb{T} \nvdash \beta$
   Do nothing if the input theory already satisfies preferred structure.
3. If $\mathbb{T}$ is a Datalog-like theory, then $\mathbb{T}'$ is also a Datalog-like theory.
   This postulate guarantees that the repaired theory also follows the restrictions of Datalog if the input theory is a Datalog-like theory. We assume that no matter working in which logic, repairs should not break the restriction of the logical format convention. Otherwise, it would come up with something that the system cannot understand, and cause unexpected problems.
4. $\mathcal{N}(\mathbb{T}') \leq \{\mathcal{N}(\mathbb{T}) + \mathcal{N}(\mathcal{T}(\mathbb{PS}))\}$
   $\mathcal{N}$ is the function giving the size of a set. This postulate means that the number of the axioms in a repaired theory should not be greater than the total number of the original theory and the true set of the preferred structuren ($\mathcal{T}(\mathbb{PS})$).

In table 4, no repairs of incompatibility would increase the size of the theory. In table 5, the first two repairs do not change $\mathcal{N}(\mathbb{T})$. As for adding a rule, $\mathcal{N}(\mathbb{T}')$ should not be bigger than $\mathcal{N}(\mathbb{T})$. Because if a rule can only prove the current preferred proposition, it is more reliable for us to add the proposition directly as an assertion. We assume a rule should be able to derive two assumptions at least; otherwise, it is not effective and should not be considered. The last repair in table 5, which directly adds a preferred proposition as an assertion, increases the size of the theory by one for each preferred proposition. It is not a wise decision obviously if you add more than one axiom for deriving a preferred proposition, in which case, why not just add the preferred one?

Therefore, when repairing a theory based on its preferred structure, we should not result in a new theory containing more axioms than the original theory and the true set of the preferred structure in total.

2.2.4 Repairs for Swan Theory by ABC System

This section will discuss the current repairs of the aforementioned Swan theory. The given theory and its preferred structure can be found in Figure 2. The problem could be caused by the fact that European swan is ambiguous: it may mean 'the European variety of swans' or 'swans resident in Europe'. In the scenario that all European variety swans are white, but Bruce is resident in Europe, the target theory is as below, where the desired repairs are underlined.

TA1. German(x, y) ⇒ European(x, y).
TA2. European(x, Variety) ∧ Swan(x) ⇒ White(x).
TA3. German(Bruce, Resident).
TA4. Swan(Bruce).
TA5. Black(Bruce).

In the rest of this section, we are going to discuss how the ABC system repairs the Swan theory and whether it can generate the targeted theory as we proposed. The order of solving incompatibility and insufficiency is immaterial. The ABC system tackles incompatibility firstly in multiple ways:

1. Delete A4: Swan(Bruce).
2. Rename European in A1:
   German(x) ⇒ Europeandash(x).
3. Add a constant argument to European in A2:
   European(x, Dummy1) ∧ Swan(x) ⇒ White(x).

The first repair, which deletes A1, blocks proof step 4 in Figure 2, which is proper in the scenario that Bruce is another kind of a bird rather than a swan. The second repair blocks proof step 2, which does not make sense in this scenario, although it solves the incompatibility. It is possible that some repairs are difficult to explain from the perspective of semantics in some scenarios. The last one is quite close to the desired repairs. As this repair changes the arity of a predicate, it has to be propagated to all instances of the predicate. Here *European* in A2 is seen as the trigger literal, which is assigned a unique constant *Dummy1*, while the other instances are assigned the common constant *DummyDefault* or a variable for rules. After propagation, the partially repaired theory is:

Repairs:
add_argument(European).
add_argument(German).

Repaired Theory:
A1'. German(x, y) ⇒ European(x, y).
A2'. European(x, Dummy1) ∧ Swan(x) ⇒ White(x).
A3'. German(Bruce, DummyDefault).
A4 . Swan(Bruce).

For aligning with changed *European* in rule A1', the arity of *German* in the rule is also changed from one to two by adding the new variable *y*, because Datalog requires that each variable in the head of a rule must also exist in its body. However, the changing of the arity is not triggered by *German*, so that the new arguments of all its instances are either variables for rules or *DummyDefault* for facts, e.g., *DummyDefault* in A3'.

For the current theory, the new proof step 3 failed because its sub-goal *German(Bruce, Dummy1)* cannot unify the input literal *German(Bruce, DummyDefaut)*, neither other input literals. Here *Dummy1* is inherited from *European(x, Dummy1)* in A2' during resolution. Now the incompatibility has been solved. The ABC system will continue to repair insufficiency. The best repair in our scenario is adding the preferred proposition as an axiom directly.

Repairs:
add_argument(European).
add_argument(German).
add_axiom: Black(Bruce).

Repaired Theory:
A1'. German(x, y) ⇒ European(x, y).
A2'. European(x, Dummy1) ∧ Swan(x) ⇒ White(x).
A3'. German(Bruce, DummyDefault).
A4 . Swan(Bruce).
A5 . Black(Bruce).

Now the theory is faithful concerning the preferred structure. The repair solution is generated by combining reformation and abduction, and the solution satisfies the postulates of the ABC system. By interpreting *Dummy1* as variety and *DummyDefault* as resident, the produced theory is the desired one given at the beginning of this section. The current ABC system does not deal with semantic interpretation, which is something worthwhile to research in the future.

In this section, it can be seen that a single technique is not enough for generating the best repairs for

a theory in some scenarios. Therefore, the ABC repair system works better than the individual techniques it combines.

3 FUTURE WORK

1. Give semantic meaning to dummy term introduced by reformation, which can make repaired theories more readable and understandable.
2. Investigate how to change sub-goals in a rule by reformation where trace-back will be needed, because it is possible that a condition in a rule is not represented properly, where reforming a sub-goal is necessary.
3. We are going to investigate more combination mechanisms for increasing the performance of the ABC repair system.
4. More heuristics are going to be built for guiding our system.
5. Epistemic entrenchment (Gärdenfors, 1988) will be evaluated for ranking all of the repair solutions for the user.
6. A method will be developed allowing a user to interact with our system to choose which repairs to prefer and to give meaning to new constants and predicates.
7. Evaluation will be based on comparing the repairs generated by the ABC system with the ones given by any of the techniques it combines, and see which can give the best solutions. The gold standard of best solutions could come from the repair history of an existing ontology, or be given by human users.

4 CONCLUSIONS

A preferred structure is defined for formalising the user's intention, and a faulty theory fails in following its preferred structure in the way of being insufficient or incompatible. For detecting these two faults, SL resolution is used as the inference rule in our system. Based on detected proofs, the ABC repair system can repair Datalog-like theories by combining abduction, belief revision and reformation, especially when tackling multiple faults. The combination technique operates at the levels of both axioms and the language of the input theory. For guiding the system generating reasonable repairs, a set of postulates are developed. We have successfully applied the current ABC system to six faulty Datalog-like theories, and our system can generate new repair solutions with better representations for these theories. In a longer journal paper under development, we are going to give a thorough treatment including the systematic introduction of motivation, comprehensive evaluation, extensive applications and so on. To ensure the termination of inference, this paper restricts the ABC repair system to Datalog. Our group's prior work was on full FOL, and with heuristic search limits on inference, ABC repair system could readily be extended to FOL. In conclusion, the ABC repair system can significantly improve the quality of repairing Datalog-like theories and fill the gap of the repair using just belief change, just abduction or only language change.

REFERENCES

Bellomarini, L., Sallinger, E., and Gottlob, G. (2018). The vadalog system: datalog-based reasoning for knowledge graphs. *Proceedings of the VLDB Endowment*, 11(9):975–987.

Bundy, A. and Mitrovic, B. (2016). Reformation: A domain-independent algorithm for theory repair. Working paper, university of edinburgh.

Ceri, S., Gottlob, G., and Tanca, L. (1989). What you always wanted to know about Datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, 1(1):146–166.

Cox, P. T. and Pietrzykowski, T. (1986). Causes for events: their computation and applications. In *International Conference on Automated Deduction*, pages 608–621. Springer.

Gallier, J. (2003). *SLD-Resolution and Logic Programming*. chapter 9 of Logic for Computer Science: Foundations of Automatic Theorem Proving. originally published by Wiley, 1986.

Gärdenfors, P. (1988). *Knowledge in flux: Modeling the dynamics of epistemic states.* The MIT press.

Gärdenfors, P. (1992). Belief revision: An introduction. In Gärdenfors, P., editor, *Belief Revision*, pages 1–28. Cambridge University Press. Cambridge Tracts in Theoretical Computer Science.

Kowalski, R. A. and Kuehner, D. (1971). Linear resolution with selection function. *Artificial Intelligence*, 2:227–60.

McNeill, F. and Bundy, A. (2007). Dynamic, automatic, first-order ontology repair by diagnosis of failed plan execution. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 3(3):1–35.

Pfenning, F. (2006). *Datalog*. Lecture 26. 15-819K: Logic Programming.