The Use of Max-Sat for Optimal Choice of Automated Theory Repairs

The Use of Max-Sat for Optimal Choice of Automated Theory Repairs (Long Version)[*]

Marius Urbonas Alan Bundy Juan Casanova Xue Li

University of Edinburgh, UK

Abstract

The ABC system repairs faulty Datalog theories using a combination of abduction, belief revision and conceptual change via reformation. Abduction and Belief Revision add/delete axioms or delete/add preconditions to rules, respectively. Reformation repairs them by changing the *language* of the faulty theory. Unfortunately, the ABC system overproduces repair suggestions. Our aim is to prune these suggestions to leave only a Pareto front of the optimal ones. We apply an algorithm for solving Max-Sat problems, which we call *the Partial Max-Sat algorithm*, to form this Pareto front.

Faulty logical theory repair Max-Sat Reformation Belief revision Abduction Datalog theories Automated theorem proving.

1 Introduction

We model the environment as a logical theory. Such a theory will need to be repaired when: errors are detected; the environment changes; or it needs to be re-tuned to cope with new tasks. The ABC system repairs faulty logical theories [Li *et al*, 2018]. It is given a theory, $\mathbb{T}$, as a set of axioms in the decidable logic Datalog [Ceri *et al*, 1990], and some observations $\mathbb{S}$, represented as a pair of sets of ground propositions. One set, $\mathcal{T}(\mathbb{S})$, is of propositions observed to be true of the environment and the other, $\mathcal{F}(\mathbb{S})$, of those observed to be false. $\mathbb{T}$ is used to make predictions about the environment. When these predictions conflict with the observations in $\mathbb{S}$, the ABC system applies a sequence of repairs to $\mathbb{T}$ until it is fault free. ABC is unique in repairing the *language* of $\mathbb{T}$ as well as the axioms.

$\mathbb{T}$'s predictions are wrong if it proves something in $\mathcal{F}(\mathbb{S})$ (incompatibility) or fails to prove something in $\mathcal{T}(\mathbb{S})$ (insufficiency). The ABC system then tries to repair $\mathbb{T}$ either by adding/deleting axioms, deleting/adding preconditions to rules or changing $\mathbb{T}$'s language. Language changes are implemented by reformation [Bundy & Mitrovic, 2016] and consist of splitting/merging predicates or constants, or adding/deleting arguments of predicates. Unfortunately, ABC produces too many repair options. In this paper, we describe and evaluate the use of Partial Max-Sat to detect and prune sub-optimal repairs. *Optimal repairs*, are those that minimise the number of any remaining or newly introduced faults. Our hypothesis then is:

> *Our Partial Max-Sat based algorithm prunes sub-optimal repairs from ABC's output. It usually terminates successfully with a significantly smaller set of fault-free, optimal repaired theories.*

The results supporting this claim are discussed in §5.

Note that a Pareto-front is required because there are conflicting requirements on the repair process. Repairing an incompatibility reduces the number of theorems in order to remove the false one. Repairing an insufficiency, on the other hand, increases the number of theorems to add the true but unprovable one. So, there may be multiple incomparable and conflicting optimal repairs.

[*]Marius Urbonas was funded by a studentship from the Student Awards Agency Scotland, Alan Bundy was funded by EPSRC grant F14R10199 and Juan Casanova by an EPSRC CDT in Data Science and a Brainnwave studentship. We are grateful to Joshua Knowles for suggesting this project, and to several anonymous reviewers for suggestions that improved the paper. This paper is a long version of the paper with the same title in the AI 2020 Conference.

Several of the algorithms we use have worst-case exponential complexity in time and/or space. These complexities do not compound and our scaling experiment at the end of §2.4 shows a quadratic time complexity.

The ABC system is not intended to be a stand-alone system, but to be a component of a larger system, for instance, with sensors, planners, actuators, etc. The wider context will help address some of the current gaps in ABC, e.g., Where do the observations come from? How to choose the best optimal, fault-free repairs? How to assign meaningful names to newly created concepts?

2 Background

We first describe the ABC system. We define: what we mean by a fault; the Datalog theories that ABC repairs; SL Resolution, which ABC uses for deduction; the repair operations it uses; and we illustrate the overproduction problem that this paper addresses.

2.1 Faults as Reasoning Failures

Both incompatibility and insufficiency arise from reasoning failures: mismatches between the theorems of a theory $\mathbb{T}$ and the observations of the environment $\langle \mathcal{T}(\mathbb{S}), \mathcal{F}(\mathbb{S})\rangle$. A *ground proposition* is a formula of the form $P(C_1, \ldots, C_n)$, where P is an n-ary predicate and the C_is are constants. So, ideally:

$$R \in \mathcal{T}(\mathbb{S}) \implies \mathbb{T} \vdash R \qquad\qquad R \in \mathcal{F}(\mathbb{S}) \implies \mathbb{T} \not\vdash R$$

That is, the true ground propositions are theorems of $\mathbb{T}$ and the false ones are not. The language of $\mathbb{T}$ is given in Definition 2 and the inference system in §2.3.

Definition 1 (Incompatible and Insufficient)

Incompatible: $\mathbb{T}$ *is* incompatible *with* $\mathbb{S}$ *iff* $\exists R.\ \mathbb{T} \vdash R \wedge R \in \mathcal{F}(\mathbb{S})$.

Insufficient: $\mathbb{T}$ *is* insufficient *for* $\mathbb{S}$ *iff* $\exists R.\ \mathbb{T} \not\vdash R \wedge R \in \mathcal{T}(\mathbb{S})$.

The ABC system detects and repairs both kinds of faults.

2.2 Datalog Theories

To ensure termination of proof search, it is convenient to limit the ABC System to a decidable logic. Datalog theories are not only decidable but are sufficiently expressive to admit a wide range of practical applications. Although reformation has been implemented for richer logics [Bundy & Mitrovic, 2016, Mitrovic, 2013]. Datalog is a logic programming language consisting of Horn clauses in which there are no functions except constants. We represent clauses in Kowalski normal form: an implication between a conjunction of the negated propositions and a disjunction of the positive propositions, i.e., in Kowalski normal form, a clause: $\neg Q_1 \vee \ldots \vee \neg Q_m \vee R_1 \vee \ldots \vee R_n$ is represented as:

$$Q_1 \wedge \ldots \wedge Q_m \quad \implies \quad R_1 \vee \ldots \vee R_n$$

In Horn clauses $n = 0$ or $n = 1$, so they fit one of the four forms in Definition 2.

Definition 2 (Datalog Formulae)

Let the language *of a Datalog theory* $\mathbb{T}$ *be a triple* $\langle \mathcal{P}, \mathcal{C}, \mathcal{V}\rangle$*, where* $\mathcal{P}$ *are the propositions, C are the constants and V are the variables. We will adopt the convention that variables are written in lower case, and constants and predicates start with a capital letter*[1].

A proposition is a formula of the form $P(t_1, \ldots, t_n)$*, where* $t_j \in \mathcal{C} \cup \mathcal{V}$ *for* $1 \leq j \leq n$*, i.e., there are no compound terms. Let* $R \in \mathcal{P}$ *and* $Q_i \in \mathcal{P}$ *for* $0 \leq i \leq m$ *in* $\mathbb{T}$*.* R *is called the* head *of the clause and the conjunction of the* Q_i*s forms the* body.

[1] The opposite of the Prolog convention.

Implication: $(Q_1 \wedge \ldots \wedge Q_m) \implies R$. *These usually represent the rules of* $\mathbb{T}$.

Assertion: $\implies R$. *These usually represent the facts of* $\mathbb{T}$.

Goals: $Q_1 \wedge \ldots \wedge Q_m \implies$. *These usually arise from the negation of the conjecture to be proved and from subsequent subgoals in a derivation.*

Empty Clause: $\implies$. *This represents false, which is the target of a refutation-style proof. Deriving it, therefore, represents success in proving a conjecture.*

Repairs operate on the language of $\mathbb{T}$ and on both its implications and assertions.

The Datalog *safety* condition requires that every variable that appears in the head of a clause also appears in the body. Variables in the head but not the body are called *orphans*[2]. There are other Datalog restrictions, but these are to make it behave efficiently as a programming language and we do not need to adopt them. As we will see, despite these restrictions, Datalog is sufficiently expressive for many practical applications.

A small Datalog theory is given in Example 1. The axioms assert that all birds can fly and are feathered, penguins are birds, and Tweety and Polly are both birds.

Example 1 (Tweety Theory) $\mathbb{T}_{Tw}$ *consists of the following set of axioms:*

$$\begin{aligned} Bird(x) &\implies Fly(x) && (1)\\ Bird(x) &\implies Feathered(x) \\ Penguin(y) &\implies Bird(y) && (2)\\ &\implies Penguin(Tweety) && (3)\\ &\implies Bird(Polly) \end{aligned}$$

2.3 Deduction by SL Resolution

Deduction in Datalog is decidable but exponential. So, if there is no proof of a conjecture, the search will eventually terminate without success, so we can be sure that the conjecture is not a theorem. Such finite failure is important for detecting insufficiencies, so was one of the technical reasons for choosing Datalog.

However, if the minimal proof is long then the search for it could exhaust the available resources. Fortunately, in many practical applications, the number of rules is small compared to the facts[3]. So proofs are quite short, even when the number of axioms is large.

Resolution proofs work by refutation: the conjecture to be proved is negated and added to the axioms. If the empty clause, $\implies$, is derived then the conjecture has been proved by *reductio ad absurdum*. In Horn clauses, the negated conjecture takes the form of a goal clause.

For deduction, we use SL Resolution [Kowalski & Kuehner, 1971], a deductive rule that is particularly well suited to fault diagnosis. A single SL Resolution step takes the following form:

$$\frac{\bigwedge_{k=1}^{i-1} R_k \wedge R_i \wedge \bigwedge_{k=i+1}^{n} R_k \implies \qquad \bigwedge_{k=1}^{m} Q_k \implies P}{(\bigwedge_{k=1}^{i-1} R_k \wedge \bigwedge_{k=1}^{m} Q_k \wedge \bigwedge_{k=i+1}^{n} R_k)\sigma \implies} \qquad (4)$$

where the highlighted R_i is the selected goal, the highlighted P is the rule head it is resolved with and σ is the most general substitution of terms for variables that will make P and R_i identical. Note that, to prevent the same variable appearing in both the selected proposition and the head of the axiom, the variables in the axiom should be renamed to new variables. To aid readability, we will do this conservatively.

[2]Although orphans cannot appear in a well-formed Datalog theory, we define them here because they may be created temporarily during the repair process, so must be identified and then eliminated by subsequent repairs.

[3]Personal communication from Frank van Harmelen. Based on the LOD-a-lot survey of the Linked Open Data cloud, he estimates that of 23.8 billion unique statements only 565 million could be classified as rules - the rest being facts, i.e., rules make up just under 2% of the total. For more detail, see `https://frankvanharmelen.home.blog/2020/07/13/2-makes-all-the-difference-on-the-lod-cloud/` accessed 14.7.20.

A SL Resolution refutation on Horn clauses takes the form of a linear sequence of SL Resolutions steps (4) in which a goal in each goal clause is resolved with either the head of an implication (rule) or an assertion (fact).

$$\begin{array}{c} \dfrac{Goal}{Goal_1 \wedge \ldots \wedge Goal_m}\ Axiom \\ \vdots \\ \dfrac{Goal'}{\implies}\ Assertion \end{array}$$

where the $Goal$s are all goal clauses and the $Axiom$s are either implication or assertion clauses.

This has the advantage that we can apply any repair directly to the axiom involved in either the current or an earlier SL Resolution step in the current branch, so we do not need to inherit the repair back up through derived clauses to an axiom. This advantage is inherited by restricting to Datalog, as all its formulae are Horn clauses, which is the second technical reason for choosing Datalog.

Example 2 uses SL Resolution to infer $Fly(Tweety)$. The highlighting is explained in §2.4.

Example 2 *We use $\mathbb{T}_{Tw}$ from Example 1.*

$$\cfrac{\cfrac{\cfrac{Fly(Tweety) \implies}{Bird(Tweety) \implies}\ Bird(x) \implies Fly(x)}{Penguin(Tweety) \implies}\ Penguin(y) \implies Bird(y)}{\implies}\ \implies Penguin(Tweety) \tag{5}$$

2.4 Repair Operations

Incompatibility and insufficiency faults are diagnosed and repaired in a dual way. $\mathcal{F}(\mathbb{S})$ and $\mathcal{T}(\mathbb{S})$ are both finite sets. The ABC system tries to prove each member of these sets. If a member of $\mathcal{F}(\mathbb{S})$ is proved then we have discovered an incompatibility. Similarly, if a member of $\mathcal{T}(\mathbb{S})$ is not proved then we have discovered an insufficiency. Incompatibilities can be repaired by blocking the unwanted proof. Insufficiencies can be repaired by unblocking a wanted, but failed proof.

The repair operations used by the ABC system are listed in Definitions 3 and 4. They are drawn from the literature on abduction and belief revision, plus our own work on reformation. Note that a single repair application may not produce a fault-free theory. Several applications may be required.

The reformation operations arose from an analysis of the first-order unification algorithm, by systematically considering how unification steps could be blocked or unblocked. Note that some first-order repair operations, such as blocking/unblocking an occurs check failure, do not apply to Datalog theories, because they do not admit function nesting. In the opposite direction, not all ABC repair operations apply to other logics, for instance, Reformation 2, which changes the arity of a predicate, does not apply to Description Logics, in which predicates are either unary (concepts or classes) or binary (roles or properties).

New applications, however, occasionally reveal the opportunity or necessity of new kinds of repair operations or the generalisation of existing operations. So, the space of repair operations seems open-ended and we make no claim to have exhausted the possibilities. In fact, given the unbounded nature of ingenuity, we doubt that an exhaustive classification of repair operations exists or, even if one did, that it could be *proved* to be exhaustive.

Definition 3 (Repair Operations for Incompatibility) *In the case of incompatibility, the unwanted proof can be blocked by causing any of the resolution steps to fail. Suppose the targeted resolution step is between a goal $P(s_1, \ldots, s_n)$ and an axiom $Body \implies P(t_1, \ldots, t_n)$, where each s_i and t_i pair can be unified. Possible repair operations are as follows:*

Belief Revision 1: *Delete the targeted axiom.*

Belief Revision 2: *Add an additional precondition to the body of an earlier rule axiom which will become an unprovable subgoal in the unwanted proof.*

Reformation 1: *Rename P in the targeted axiom to the new predicate P'.*

Reformation 2: *Increase the arity of all occurrences P in the axioms by one. Ensure, recursively, that the new arguments, s_{n+1} and t_{n+1}, in the targeted occurrence of P, are not unifiable.*

Reformation 3: *For some i, suppose s_i is C. Since s_i and t_i unify, t_i is either C or a variable. Change t_i to the new constant C'.*

Definition 4 (Repair Operations for Insufficiency) *In the case of insufficiency, the wanted but failed proof can be unblocked by causing a currently failing resolution step to succeed. Suppose the chosen resolution step is between a goal $P(s_1, \ldots, s_m)$ and an axiom $Body \implies P'(t_1, \ldots, t_n)$, where either $P \neq P'$ or, for some i, s_i and t_i cannot be unified. Possible repair operations are:*

Abduction 1: *Add a new axiom whose head unifies with the goal $P(s_1, \ldots, s_m)$.*

Abduction 2: *Locate the rule whose body proposition created this goal and delete this proposition from the rule.*

Reformation 4: *Replace $P'(t_1, \ldots, t_n)$ in the axiom with $P(s_1, \ldots, s_m)$.*

Reformation 5: *Suppose s_i and t_i are not unifiable. Remove the i^{th} argument from all occurrences of P'.*

Reformation 6: *If s_i and t_i are not unifiable, then they are unequal constants, say, C and C'. Either (a) rename all occurrences of C' in the axioms to C or (b) replace the offending occurrence of C' in the targeted axiom by a new variable.*

Note that we disallow repairs that would change $\mathbb{S}$. This is because $\mathbb{S}$ consists of *observations* of the environment. Our goal is to repair the theory $\mathbb{T}$ so that it predicts our observations $\mathbb{S}$ of the environment - not the other way around. There is also the practical consideration that if a predicate, say, $P(C) \in \mathcal{T}(\mathbb{S})$, were changed to, say, $P(C, Normal)$ and $P(C, Abnormal)$ then we would have no basis to say whether either of them belonged to $\mathcal{T}(\mathbb{S})$ or $\mathcal{F}(\mathbb{S})$. This would make it difficult to track the progress of a sequence of repairs. This restriction is implemented by a mechanism that protects nominated predicates and constants from being changed by repairs [Li *et al*, 2018].

A repair of an incompatibility can be illustrated with $\mathbb{T}_{Tw}$ from Example 1 and the refutation in Example 2. Suppose we observe that $Tweety$ cannot fly, i.e., that $Fly(Tweety) \in \mathcal{F}(\mathbb{S})$. Since refutation 2 proves $Fly(Tweety)$, we have an incompatibility. Suppose we decide to break the unwanted refutation 2 at the highlighted resolution step. One repair suggestion is to apply Reformation 2 from Definition 3. This will give the repaired theory $\nu(\mathbb{T}_{Tw})$[4]:

$$\begin{aligned} Bird(x, Normal) &\implies Fly(x) \\ Bird(x, y) &\implies Feathered(x) \\ Penguin(y) &\implies Bird(y, Abnormal) \\ &\implies Penguin(Tweety) \\ &\implies Bird(Polly, Normal) \end{aligned}$$

where $Normal$ and $Abnormal$ are two new constants. $Fly(Tweety)$ is no longer a theorem of this repaired theory.

The naming of these two new constants was suggested by the observation that new constants introduced by repair Reformation 2, i.e.. by giving P a new argument, often distinguish two kinds of P, where the abnormal kind was from the axiom in the now broken resolution step.

These repair operations have been applied to a wide range of examples, some of which can be found in Table 1. In addition, we have evaluated the scalability of the ABC system by applying it to the alignment of two commercial databases with sample sizes up to 1020 entries[5]. Known misalignments were put into $\mathcal{F}(\mathbb{S})$ and the remainder into $\mathcal{T}(\mathbb{S})$.

[4]Pronounced 'new $\mathbb{T}_{Tw}$'.

[5]The details are subject to NDA, so have been anonymised.

The alignment theory $\mathbb{T}_a$ contains two integrity constraints:

$$\begin{aligned} Range(x,y,In) \wedge DatabaseA(x) \wedge DatabaseB(y) &\implies Match(x,y) \\ Range(x,y,In) \wedge Range(x,y,Out) &\implies \end{aligned}$$

We illustrate the repair process with a very simple example. Suppose that $\mathcal{T}(\mathbb{S}) = \{Match(A,B), Match(C,D)\}$ and $\mathcal{F}(\mathbb{S}) = \{Match(E,F)\}$. The very small database on the LHS column below has an insufficiency and an incompatibility. The middle database has been repaired with Reformation 6 and 3. The one on the RHS has been repaired with Reformation 6 and Belief Revision 1.

$Insufficiency$	$Range(A,B,Out)$	$Range(A,B,In)$	$Range(A,B,In)$
$Correct$	$Range(C,D,In)$	$Range(C,D,In)$	$Range(C,D,In)$
$Incompatibility$	$Range(E,F,In)$	$Range(E,F,Out)$	$Range(E,F,In)$

The time taken to find all repairs for a sample was found to be a quadratic function of the size of the sample, so the ABC system was shown experimentally to have a feasible computational complexity.

2.5 Overproduction of Repair Suggestions

The main problem with the theory repair mechanism outlined in §2.4, is overproduction, i.e., it makes too many repair suggestions. *The contribution of this paper is a Partial Max-Sat-based mechanism for pruning sub-optimal repair suggestions.*

To illustrate the problem, let us consider some of the other repair suggestions that the ABC system generates for repairing the incompatibility in the theory $\mathbb{T}_{Tw}$ from Example 1.

Note that the ABC system can break the unwanted proof in Example 2 at each of the 3 resolution steps, and those steps can be broken using each of the 5 repair operations described in Definition 3, sometimes in more than one way. For the purposes of analysis, let us additionally assume the observations $Feathered(Tweety) \in \mathcal{T}(\mathbb{S})$ and $Fly(Polly) \in \mathcal{T}(\mathbb{S})$. Note that both $Feathered(Tweety)$ and $Fly(Polly)$ are theorems of $\mathbb{T}$. So a new insufficiency will be introduced if either of them is not a theorem of the repaired theory $\nu(\mathbb{T})$. Consider the following repair suggestions to $\mathbb{T}$.

Belief Revision 1: Delete axiom (2), for instance. Note that, $Feathered(Tweety)$ is no longer a theorem, so this deletion will cause an insufficiency.

Belief Revision 2: Add an additional precondition to the body of axiom (2). User interaction is required to suggest a suitable precondition. Moreover, $Feathered(Tweety)$ is no longer a theorem, so this repair will also cause an insufficiency.

Reformation 1: Rename $Bird$ in axiom (2) to the new predicate $Bird'$. Note that $Feathered(Tweety)$ is no longer a theorem, which causes the same insufficiency as in the previous two repairs. If, instead, $Bird$ in axiom 1 were renamed, then $Fly(Polly)$ would cease to be a theorem which would cause a different insufficiency.

Reformation 2: This is the repair described in §2.4. Note that $Feathered(Tweety)$ and $Fly(Polly)$ are still theorems, so this repair avoids the insufficiencies caused by the other four repairs.

Reformation 3: This is not applicable to axiom (2), but could be applied to axiom (3) to rewrite it to $\implies Penguin(Tweety')$. Note that $Feathered(Tweety)$ is no longer a theorem. In addition, a new incompatibility will be caused if it is observed that $Fly(Tweety') \in \mathcal{F}(\mathbb{S})$.

Without pruning sub-optimal repairs, the ABC System makes 10 repair suggestions for this faulty theory. For incompatibilities with several or longer unwanted proofs, the number of repair suggestions can be much more. The pruning mechanism described in §4, will prune all but the Reformation 2 repair described in §2.4.

3 Pruning out Sub-Optimal Repairs

The ABC System is applied to Datalog theories, whereas Partial Max-Sat, which is the main component of our pruning mechanism, and similar Sat-based algorithms, are designed for propositional logic. The theory behind reducing Datalog-like theories to propositional ones is well known, but is briefly discussed in §3.1. This is followed by a brief introduction to Partial Max-Sat in §3.2

and how we use it in §4.

3.1 Turning First-Order Theories into Propositional Logic

All Datalog theories can be converted into equivalent propositional ones. Note that if we ground all axioms in a theory $\mathbb{T}$ by instantiating their variables in all possible ways with constants we will get another theory $Ground(\mathbb{T})$ in which all the axioms are variable-free Horn clauses. Since Datalog theories have no non-nullary functions, $Ground(\mathbb{T})$ has only a finite number of axioms. Moreover, $Ground(\mathbb{T})$ has a model iff $\mathbb{T}$ has one [Herbrand, 1930]. We can view $Ground(\mathbb{T})$ as a propositional theory, so SAT-related algorithms can be applied to it to solve $\mathbb{T}$ problems. Since every occurrence of each variable in $\mathbb{T}$ must be instantiated in $|\mathcal{C}|$ ways then this grounding is an exponential process in time and space.

This holds because we consider free variables to be implicitly universally quantified. By treating each ground proposition as a propositional variable, we can view $Ground(\mathbb{T})$ as a propositional theory. It may appear paradoxical that a formula may simultaneously be variable free when viewed as a Datalog theory, but consist of propositional variables when viewed as a Propositional theory. A Datalog theory has a finite number of axioms and, hence, a finite number of propositions and constants. Therefore, $Ground(\mathbb{T})$ is also finite and SAT-related algorithms can be applied to it. Input to the Ground function is a Datalog-like theory $\mathbb{T}$, which is a set of Horn clauses. The output is an equivalent theory in propositional logic represented in clausal normal form, which is a standard input for many Sat-based algorithms.

Definition 5 (Grounding a Datalog Theory)

$$Ground(\mathbb{T}) \quad = \quad \{\phi\sigma \mid \phi \in \mathbb{T} \wedge \sigma : \mathcal{V} \mapsto \mathcal{C}\}$$

The $Ground$ function is illustrated in Example 3.

Example 3 (Grounding a Theory) *Let $\mathbb{T}_{pqr}$ be the following set of axioms:*

$$P(x) \implies Q(x), \implies P(A), \implies R(B)$$

Then $Ground(\mathbb{T}_{pqr})$ is the set: $\{P(A) \implies Q(A), P(B) \implies Q(B), \implies P(A), \implies R(B)\}$

3.2 Partial Max-Sat

Partial Max-Sat ($pMaxSat$) specifies the problem in which given two arguments, φ_h and φ_s, denoting sets of ground hard and soft clauses respectively, the goal is to find all assignments of truth values to them such that: (a) all clauses in φ_h are satisfied, i.e., have a model, and (b) the maximum number of clauses in φ_s are satisfied. We use Herbrand models instead of Tarskian models. Herbrand [Herbrand, 1930] has shown that a theory has a Tarskian model iff it has a Herbrand model. A Herbrand model that meets this specification is called *optimal.*

Definition 6 (Optimal Herbrand Models)

A Herbrand model assigns a truth value to each propositional variable. In our case these are the ground propositions created by the Ground function.

The Herbrand Base $\mathcal{HB}(\mathbb{T})$ *of a Datalog theory* $\mathbb{T}$ *is:*

$$\mathcal{HB}(\mathbb{T}) \quad = \quad \{P(t_1, \ldots, t_n)\sigma | \sigma : \mathcal{V} \mapsto \mathcal{C} \wedge P(t_1, \ldots, t_n) \in \mathcal{P}\}$$

The Herbrand Models $\mathcal{HM}(\mathbb{T})$ *of* $\mathbb{T}$ *are subsets of* $\mathcal{HB}(\mathbb{T})$ *for which*

$$\forall \alpha \in \mathbb{T}, \forall hm \in \mathcal{HM}(\mathbb{T}).\ hm \models \alpha$$

A Herbrand Model $hm \in \mathcal{HM}(\mathbb{T})$ *is* optimal *iff*

$$\begin{aligned} &\forall \beta \in \varphi_h.\ hm \models \beta \quad \wedge \quad \forall hm' \in \mathcal{HM}(\mathbb{T}). \\ &\quad |\{(\beta)\ \in \varphi_s | hm \models (\beta)\}| \quad \geq \quad |\{(\beta \implies)\ \in \varphi_s | hm' \models (\beta)\}| \end{aligned} \tag{6}$$

Let $pMaxSat$ be an algorithm, specified in Definition 7, that returns size of the subset of φ_s that is *not satisfied* by an optimal Herbrand model. Note that, as a consequence of (6), this size will be the same for all such models.

Definition 7 (Partial Max-Sat Specification)

$$pMaxSat(\varphi_h, \varphi_s) = |\{(\beta)\ \in \varphi_s | hm \not\models (\beta)\}|$$

where hm *is any optimal Herbrand Model.*

We augmented the ABC system with a third-party Partial Max-Sat solver [Ignatiev *et al*, 2018], based on the Fu & Malik algorithm [Fu & Malik, 2006]. The implementation follows the WMSU1 variant [Vasco M. Manquinho, 2009] of the algorithm. This has served well for our initial feasibility study, but as we attempt to scale up, we will investigate more efficient alternatives. Note that deciding whether k clauses can be satisfied is NP-Complete but solving the optimization problem with weighted Max-SAT is FP^{NP}.

3.3 Evaluating Fitness of Repairs

This section discusses which repairs are considered to be sub-optimal and how to detect them using automated reasoning.

We want to find repairs $\nu(\mathbb{T})$ of a faulty $\mathbb{T}$ so as to maximise the size of $\{\phi \in \mathcal{T}(\mathbb{S}) | \nu(\mathbb{T}) \vdash \phi\}$ and minimise the size of $\{\phi \in \mathcal{F}(\mathbb{S}) | \nu(\mathbb{T}) \vdash \phi\}$. It will not, in general, be possible to achieve both of these requirements with a single repair, so we need to find all repairs ν that are optimal wrt some measures of these potentially conflicting requirements. As previously noted, these requirements entail that $\mathcal{T}(\mathbb{S})$ and $\mathcal{F}(\mathbb{S})$ *must not* be altered by ν, so the predicates and constants in them are protected from change.

3.4 Pareto Optimality

It suffices to define what it means for one theory to strictly dominate another. The Pareto front of optimal repairs is then just the maximal set of repairs such that no member is strictly dominated by any other repair. Any repair not in the Pareto front is sub-optimal.

We will first need to define the insufficiency set $\mathcal{IS}(\mathbb{T}, \mathbb{S})$ of members of $\mathcal{T}(\mathbb{S})$ that are *not* theorems and the incompatibility set $\mathcal{IC}(\mathbb{T}, \mathbb{S})$ of members of $\mathcal{F}(\mathbb{S})$ that *are* theorems.

Definition 8 (The Incompatibility and Insufficiency sets) *Let:*

$$\mathcal{IS}(\mathbb{T}, \mathbb{S}) = \{\phi \in \mathcal{T}(\mathbb{S}) | \mathbb{T} \not\vdash \phi\} \qquad \wedge \qquad \mathcal{IC}(\mathbb{T}, \mathbb{S}) = \{\phi \in \mathcal{F}(\mathbb{S}) | \mathbb{T} \vdash \phi\}$$

Then we can define when one repair strictly dominates another.

Definition 9 (Strictly Dominated Repair) *Given two repairs ν_k and ν_j, ν_j is strictly dominated by ν_k iff:*

$$|\mathcal{IS}(\nu_k(\mathbb{T}), \mathbb{S})| \leq^* |\mathcal{IS}(\nu_j(\mathbb{T}), \mathbb{S})| \qquad \wedge \qquad |\mathcal{IC}(\nu_k(\mathbb{T}), \mathbb{S})| \leq^* |\mathcal{IC}(\nu_j(\mathbb{T}), \mathbb{S})|$$

$\leq^*$*: one of the signs has to be a strict inequality.*

Example 4 (Strict Domination) *We compare two of the repairs of $\mathbb{T}_{Tw}$ in Definition 1 from §2.5. Let ν_{b1} be the deletion of axiom (1) and ν_{r2} be the addition of an argument to $Bird$. Where:*

$$\mathcal{T}(\mathbb{S}) = \{Feathered(Tweety), Fly(Polly)\} \qquad \wedge \qquad \mathcal{F}(\mathbb{S}) = \{Fly(Tweety)\}$$

Then $\mathcal{IC}(\nu_k(\mathbb{T}), \mathbb{S})$ is empty for both repairs and $\mathcal{IS}(\nu_k(\mathbb{T}), \mathbb{S})$ is empty for ν_{r2}, but for ν_{b1}, $\mathcal{IS}(\nu_k(\mathbb{T}), \mathbb{S}) = \{Feathered(Tweety), Fly(Polly)\}$. Therefore, ν_{r2} strictly dominates ν_{b1} and, hence, ν_{b1} is sub-optimal.

4 Pruning mechanism

The pruning mechanism provides a way to reduce the search space of repairs of a given faulty theory in an automatic way. The inputs to the pruning mechanism are given by the ABC theory-repair algorithm: a Datalog-like theory $\mathbb{T}$, a set of repairs $\{\nu_1, \nu_2, \ldots, \nu_k\}$ and a pair of sets of environmental observations $\langle \mathcal{T}(\mathbb{S}), \mathcal{F}(\mathbb{S}) \rangle$. The output is the sub-set of repair suggestions that are Pareto optimal: $\{\nu_{n_1}, \ldots, \nu_{n_j}\}$.

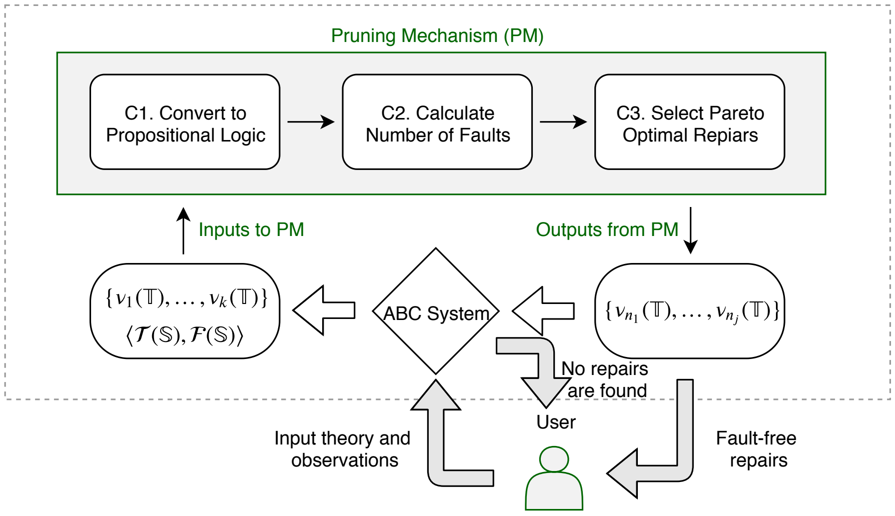
Pruning Mechanism (PM)
C1. Convert to Propositional Logic
C2. Calculate Number of Faults
C3. Select Pareto Optimal Repairs
Inputs to PM
Outputs from PM
$\{\nu_1(\mathbb{T}), \ldots, \nu_k(\mathbb{T})\}$ $\langle \mathcal{T}(\mathbb{S}), \mathcal{F}(\mathbb{S}) \rangle$
ABC System
$\{\nu_{n_1}(\mathbb{T}), \ldots, \nu_{n_j}(\mathbb{T})\}$
No repairs are found
User
Input theory and observations
Fault-free repairs

Figure 1: The components (C1-C3) of the pruning mechanism repair, where $\mathbb{T}$ is a Datalog-like theory, $\{\nu_1, ..., \nu_k\}$ is a set of repairs generated by the ABC algorithm, $\langle \mathcal{T}(\mathbb{T}), \mathcal{F}(\mathbb{T}) \rangle$ are the observations from the environment and the output is a set of optimal, fault-free repairs.

Figure 1 shows the high-level components of the mechanism. At each step the ABC system generates a set of repairs of a faulty theory. C1 applies each generated repair and converts the resulting Datalog-like theory to propositional logic using $Ground$ (see §3.1). C2 is the central part of the mechanism, which uses $pMaxSat$ to determine how many faults were fixed by each repair ν, and how many new faults it introduces to the repaired theory $\nu(\mathbb{T})$. In C3 the set of Pareto optimal repairs are returned to the cycle as (possibly only partially) repaired Datalog theories. This process is repeated on the repaired theories until no faults remain or no further repairs are generated. Any fault-free theories are returned to the user. $\mathbb{S}$ remains unchanged throughout.

Even though, given unbounded resources, the constituent processes of this cycle each terminate, there is the possibility of non-termination of the whole cycle. ABC might reach a situation in which faults still remain, but each repair of them fails to decrease the overall number of faults. This can happen because, as we saw in §2.5, a repair can introduce new faults when fixing an old one. This has not happened in any of our test examples, but it remains a theoretical possibility. It is this kind of whole cycle non-termination that gives rise to the 'usually' caveat in our hypothesis.

4.1 The use of Partial MAX-SAT for Determining Optimal Repairs

Definition 9 is used to determine whether a repair ν_k is sub-optimal and should be pruned. This requires us to calculate the sizes of incompatibility and insufficiency sets: $|\mathcal{IC}(\nu_k(\mathbb{T}), \mathbb{S})|$ and $|\mathcal{IS}(\nu_k(\mathbb{T}), \mathbb{S})|$.

4.1.1 Calculating the size of the incompatibility and insufficiency sets

Definition 10 calculates N_C and N_S by specifying the φ_h and φ_s to apply $pMaxSat$ to. Theorem 1 proves N_C and N_S to be $|\mathcal{IC}(\nu_k(\mathbb{T}), \mathbb{S})|$ and $|\mathcal{IS}(\nu_k(\mathbb{T}), \mathbb{S})|$, respectively.

Definition 10 (Calculating N_C and N_S for ν_k using pMaxSat)
Let $N_C = pMaxSat(\varphi_h, \varphi_s)$, where:

$$\varphi_h = Ground(\nu_k(\mathbb{T})) \qquad \wedge \qquad \varphi_s = \{\beta \implies |\beta \in \mathcal{F}(\mathbb{S})\}$$

Let $N_S = |\mathcal{T}(\mathbb{S})| - pMaxSat(\varphi_h, \varphi_s)$, where:

$$\varphi_h = Ground(\nu_k(\mathbb{T})) \qquad \wedge \qquad \varphi_s = \{\beta \implies |\beta \in \mathcal{T}(\mathbb{S})\}$$

Theorem 1 (Correctness of Definition 10) *Definition 10 correctly calculates the size of the incompatibility and insufficiency sets of a repaired theory $\nu_k(\mathbb{T})$, i.e.*

$$N_C = |\mathcal{IC}(\nu_k(\mathbb{T}), \mathbb{S})| \quad \wedge \quad N_S = |\mathcal{IS}(\nu_k(\mathbb{T}), \mathbb{S})|$$

Proof The proofs for N_C and N_S are similar. First apply the definitions of φ_h given in Definition 10. Use Definition 7 to apply the definitions of N_C and N_S. Then use Definition 8 to show the equivalences, appealing to the consistency of Datalog theories.

Consider first N_C. Recall that $\varphi_h = Ground(\nu_k(\mathbb{T})) \wedge \varphi_s = \{\beta \implies |\beta \in \mathcal{F}(\mathbb{S})\}$

And by definition 7:

$$N_C = pMaxSat(\varphi_h, \varphi_s) = |\{(\beta \implies) \in \varphi_s | hm \not\models (\beta \implies)\}|$$

where hm is any optimal Herbrand Model.

Then:

$$\begin{aligned} \beta \in \mathcal{IC}(\nu_k(\mathbb{T}), \mathbb{S}) &\iff \nu_k(\mathbb{T}) \vdash \beta \wedge \beta \in \mathcal{F}(\mathbb{S}) \\ &\iff hm \not\models (\beta \implies) \wedge (\beta \implies) \in \varphi_S \end{aligned}$$

The first step follows in both directions directly from unpacking definition 8. The second step breaks into two parts: hm is a model of $\nu_k(\mathbb{T})$, which as a Datalog theory is necessarily consistent, so:

$$\nu_k(\mathbb{T}) \vdash \beta \iff hm \models \beta \iff hm \not\models (\beta \implies)$$

From the definition of φ_s: $(\beta \implies) \in \varphi_S \iff \beta \in \mathcal{F}(\mathbb{S})$

So, substituting $\beta \implies$ for β: $\beta \in \varphi_S \iff (\beta \implies) \in \mathcal{F}(\mathbb{S})$

So: $N_C = |\{(\beta \implies) \in \varphi_s | hm \not\models (\beta \implies)\}| = |\mathcal{IC}(\nu_k(\mathbb{T}), \mathbb{S})|$

Now consider N_S. Recall that: $\varphi_h = Ground(\nu_k(\mathbb{T})) \wedge \varphi_s = \{\beta \implies |\beta \in \mathcal{T}(\mathbb{S})\}$

And by definition 7,

$$\begin{aligned} N_S &= |\mathcal{T}(\mathbb{S})| - pMaxSat(\varphi_h, \varphi_s) \\ &= |\{(\beta \implies) \in \varphi_s\}| - |\{(\beta \implies) \in \varphi_s | hm \not\models (\beta \implies)\}| \\ &= |\{(\beta \implies) \in \varphi_s | hm \models (\beta \implies)\}| \end{aligned}$$

where hm is any optimal Herbrand Model.

$$\begin{aligned} \beta \in \mathcal{IS}(\nu_k(\mathbb{T}), \mathbb{S}) &\iff \mathbb{T} \not\vdash \beta \wedge \beta \in \mathcal{T}(\mathbb{S}) \\ &\iff hm \models (\beta \implies) \wedge (\beta \implies) \in \varphi_s \end{aligned}$$

The proof of these steps are direct analogues of those N_C, so are omitted. So:

$$N_S = |\{(\beta \implies) \in \varphi_s | hm \models (\beta \implies)\}| = |\mathcal{IS}(\nu_k(\mathbb{T}), \mathbb{S})| \quad \square$$

5 Evaluation

In this section we evaluate the hypothesis:

> *Our Partial Max-Sat based algorithm prunes sub-optimal repairs from ABC's output. It usually terminates successfully with a significantly smaller set of fault-free, optimal repaired theories.*

By construction, the Pareto-fronts generated by the Partial Max-Sat based filter consist solely of optimal repairs. Table 1 shows the result of repairing a test set of 10 faulty theories[6]. It shows the reductions in size between the set of repair suggestions originally generated by the ABC system and these Pareto-fronts.

No one standard benchmark test set is available that could be used to evaluate the diverse abilities of the ABC system. In order to show the generality of our techniques and avoid bias in the evaluation, these test examples were instead drawn from benchmark test and development sets used in research papers in a diverse range of areas of AI, including non-monotonic reasoning, belief revision, etc.

As previously noted, even these optimal repairs may not eliminate all the faults in the input theory. Further rounds of repair may be required to the resulting partially repaired theories. The size reductions are given for only the first round of repairs. This recursive process may be viewed as a search tree, where the nodes are labelled with theories and the arcs between them with optimal repairs. For success, we require only one branch of this search tree to terminate with a leaf node labelled with a fault-free theory, but sometimes multiple fault-free theories are found. Table 1 shows that success was achieved in all 10 examples.

The columns of Table 1 give the following statistics:

Name: The names of the 10 faulty theories in our test set, plus Tweety (1). The remaining theories can be found in Prolog form in Appendix A.

#A: The number of axioms in each faulty theory.

#Unf: The number of first-round, unfiltered repair suggestions.

#Fil: The size of the Pareto front after the first round. The percentages in parentheses indicate the reduction achieved.

#H and S: The size of the initial hard and soft clause sets.

#PV: The size of the initial propositional variable set.

Time: The average, over 3 runs, of the time (μs) to generate a fault-free theory.

Succ(n): n is the number of fault-free theories generated, if any.

Reference: The citations of the source of the example, with a note on any adaptions.

Note that the repair process terminates with success for all our 11 examples. The size reduction achieved by our filtering process varies widely from 0% to 93%. This variation can be partially explained by the number of fault-free theories that are eventually returned - where a large number of fault-free repairs exist, then at least that number of repair sequences are needed to find them all. *From these results we conclude that our hypothesis has been empirically confirmed.*

[6] Plus our development example 1.

[7] By adding bird(Polly) and brokenWing(Polly) as suggested in [Gómez *et al*, 2010]; birds with broken wings cannot fly.

[8] Adapted by adding goodPrice(Blockbuster) and closing(Blockbuster) as suggested in [Gómez *et al*, 2010]; ontology should infer that you should not buy stocks of a company which has good stock price but is closing down.

Name	#A	#Unf	#Fil	#H	#S	#PV	Time	Succ(n)	Reference
CapOf	3	14	12 (15%)	29	4	18	575 ms	Y (12)	[Bundy & Mitrovic, 2016]
TooManyMums	3	14	14 (0%)	29	3	19	188 ms	Y (14)	[Bundy & Mitrovic, 2016]
Tweety	5	10	1 (90%)	8	3	8	158 ms	Y (1)	[Strasser & Antonelli, 2019]
MarriedWoman	5	12	6 (50%)	8	2	8	159 ms	Y (6)	From Exa. 3.6 [Gómez *et al*, 2010]
Researcher	5	13	5 (62%)	9	2	10	154 ms	Y (5)	From Exa. 1 [Rodler & Eichholzer, 2019]
Swan	6	20	11 (45%)	8	4	9	1940 ms	Y (54)	[Gärdenfors, 1988]
Bat	6	12	4 (67%)	11	4	14	4199 ms	Y (24)	[Wan *et al*, 2016]
SuperPenguin	6	16	14 (13%)	6	2	6	164 ms	Y (14)	From Exa. 3.1 [Gómez *et al*, 2010]
BuyStock	8	20	18 (10%)	15	2	16	156 ms	Y (18)	From Exa. 3.3 [Gómez *et al*, 2010]
SuperPenguin_v2[7]	8	20	12 (40%)	12	4	12	169 ms	Y (12)	From Exa. 3.1 [Gómez *et al*, 2010]
BuyStock_v2	10	27	2 (93%)	26	4	23	6260 ms	Y (2)	From Example 3.3[8] [Gómez *et al*, 2010]

Table 1: Experimental results showing the comparison between the first-round of repairs using the baseline unfiltered ABC system vs. the pruning mechanism.

5.1 Researcher ontology repair example

To illustrate the usefulness of the pruning mechanism consider the following simple ontology from [Rodler & Eichholzer, 2019]:

$$Researcher(x) \implies Writes(x, Papers) \tag{7}$$
$$Writes(x, Papers) \implies Author(x) \tag{8}$$
$$Author(x) \implies Employee(x) \tag{9}$$
$$Author(x) \implies Person(x) \tag{10}$$
$$\implies Researcher(Ann) \tag{11}$$

In natural language terms the ontology states that all researchers write papers, everyone who writes papers is an author and authors are both employees and people. As suggested in [Rodler & Eichholzer, 2019], the observation in made that Ann is not an employee but an independent researcher (*Employee(Ann)* $\in \mathcal{F}(\mathbb{S})$), although Ann most certainly is a person (*Person(Ann)* $\in \mathcal{T}(\mathbb{S})$). The pruning mechanism suggests the following repairs:

1. Rename the $Author$ predicate in axiom (9), changing the axiom into:

$$Author\_dash(x) \implies Employee(x)$$

2. Increase the arity of $Author$, which changes axioms (8), (10) and (9) respectively to:

$$Writes(x, Papers) \implies Author(x, Abnormal)$$
$$Author(x, Normal) \implies Employee(x)$$

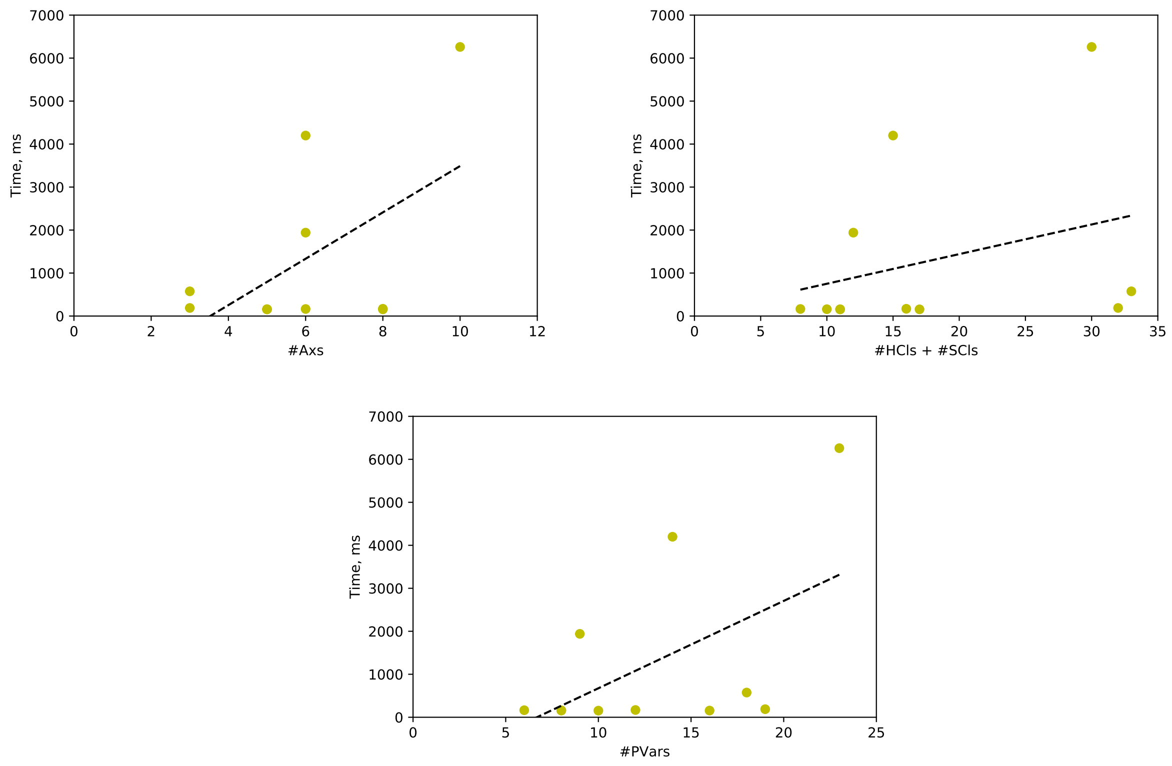

Figure 2: Experimental results showing the dependency between different measures of theory size and time it took to reach a faultless theory. A dashed linear regression line shows the general trend of time growth. Despite the theoretical exponential worst case complexity of the algorithms we have used, the empirical complexity is roughly linear.

3. Increase the arity of Author, which changes axioms (8), (10) and (9). changing axioms (9) and (10) respectively to:

$$\begin{aligned} Writes(x, Papers) &\implies Author(x, Normal) \\ Author(x, Abnormal) &\implies Employee(x) \end{aligned}$$

4. Weaken the variable x in the $Employee$ predicate to a new constant in axiom (9).

$$Author(x) \implies Employee(Ann\_dash)$$

5. Delete axiom (9).

Some examples of repairs which were pruned out by $pMaxSat$ were: the retraction of either axiom (7), (8) or (11); an increase in the arity of $Writes$; and changing the name of predicate $Researcher$ in axioms (7) or (11).

The incompatibility with the observation that Ann is not an employee stems from the fault in the ontology which fails to consider that not every author has to be an employee, but instead can be an independent author doing it in their free time.

One of the effects that the pruning mechanism has in this case is that every suggested repair directly targets axiom (9) which is the source of the erroneous assertion. Repairs which are discarded by $pMaxSat$ are those that target axioms which do not necessarily change axiom (9) and therefore are not useful in helping to remodel theory $\mathbb{T}_{Re}$ to fit with underlying environment $\mathbb{S}$ better.

Since, in this case, repairs suggested by the pruning mechanism directly target the faulty axiom, it is also easy to find reasonable interpretations for most suggested repairs. But note that the ABC system does not have the capability to ground new predicates or constants, so these interpretations are provided manually. Suggested Repair 1 can be interpreted as changing $Author$ predicate into $FundedAuthor$ which would imply that this author is getting paid and therefore is an employee. Repairs 2 and 3 suggest that predicate $Author$ could have additional property which could be interpreted as 'independent' or 'funded'. These changes however do not allow to model environment $\mathbb{S}$ completely correctly as the theory would imply that everyone who writes papers has to be an independent author. More observations would be required to find subsequent optimal repairs or reject these repair suggestions. Repair 4 and repair 5 also target axiom (9), but not, however, in a desirable way, additional observations would make it easy to prune these repairs out, e.g., adding authors who *are* employees.

6 Conclusion

The ABC system automates the repair of faulty Datalog theories. It detects faults in a theory $\mathbb{T}$ by testing it against observations of the environment $\mathbb{S}$, represented as a pair of sets of ground propositions $\langle \mathcal{T}(\mathbb{S}), \mathcal{F}(\mathbb{S}) \rangle$, where $\mathcal{T}(\mathbb{S})$ is a set of true ground propositions and $\mathcal{F}(\mathbb{S})$ is a set of false ones. A fault can be an incompatibility, where $\mathbb{T} \vdash \phi$ for some $\phi \in \mathcal{F}(\mathbb{S})$, or an insufficiency, where $\mathbb{T} \not\vdash \phi$ for some $\phi \in \mathcal{T}(\mathbb{S})$. ABC repairs theories by a combination of abduction, belief revision and reformation.

Unfortunately, the ABC system overproduces repair suggestions. In this paper, we describe a pruning mechanism based on Partial Max-Sat, which outputs a Pareto front of optimal repair suggestions. Our empirical results confirm that, by pruning out sub-optimal repair suggestions, this mechanism significantly reduces the number of repair suggestions while retaining repairs that lead to a successful outcome of a fault-free theory. Further details can be found in [Urbonas, 2019].

Inductive Logic Programming [Muggleton *et al*, 2012] also constructs logic programs from positive and negative examples and can invent new intermediate predicates to complete recursive programs. The main differences between ILP and the ABC system is that: ABC repairs faulty theories, which may not be recursive programs; it can change the arity of predicates; split/merge constants, predicates and preconditions. We are currently working with Muggleton's team to apply both techniques to the modelling of virtual bargaining [Misyak *et al*, 2016], which will provide a vehicle to further compare and contrast them.

In future work, we intend to explore the application of similar pruning mechanisms to richer logics, including some of those for which we have previously implemented reformation [Mitrovic, 2013, Bundy & Mitrovic, 2016]. When these logics are not decidable or when examples require infeasible run times, we will need to impose resource limits. These limits might mean that we sometimes fail to detect incompatibilities or misclassify an insufficiency. But it will still be useful to find a subset of all incompatibilities or to repair a false insufficiency so that the repaired theory finds a shorter proof of it. It is also the case that for large enough problems the Partial Max-Sat algorithm might not terminate in a feasible amount of time. We will look into using intermediate solutions.

Our test examples are currently quite small. To see whether our techniques can scale up, we have just started a collaboration with Huawei to apply ABC to the large knowledge graphs they are using. Despite the exponential complexity of several of the algorithms that ABC uses, we are optimistic. This is because we can focus them on just the sub-theories consisting of only the axioms used in a (partial) proof. In many practical applications, these proofs are often small and shallow.

A Examples in Prolog Form

A.1 bat.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Theory:


fact([-bat(\x),+can_fly(\x)]).
fact([-bat(\x),+live_in(\x, cave)]).
fact([-live_in(\x, cave),+trogloxene(\x)]).
fact([-mammal(\x),+can_not_fly(\x)]).
fact([-can_fly(\x),+bird(\x)]).
fact([+bat(barry)]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Preferred Structure:

trueSet([mammal(barry), can_fly(barry)]).
falseSet([can_not_fly(barry), bird(barry)]).
protect([]).
```

A.2 buyStock.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Theory:

fact([-goodPrice(\x),+buyStock(\x)]).
fact([-goodPrice(\x),-riskyCompany(\y),+dontBuyStock(\x)]).
fact([-inFusion(\x,steel),+riskyCompany(\x)]).
fact([-closing(\x,\y),+riskyCompany(\x)]).
fact([-inFusion(\x,steel),-stong(steel),+notRiskyCompany(\x)]).
fact([+goodPrice(acme)]).
fact([+inFusion(acme,steel)]).
fact([+strong(steel)]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Preferred Structure:

trueSet([buyStock(acme)]).
falseSet([dontBuyStock(acme)]).
protect([]).
heuristics([]).
```

A.3 **buyStock**$_v 2.pl$

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Main theory:

fact([-goodPrice(\x),+buyStock(\x)]).
fact([-goodPrice(\x),-riskyCompany(\y),+dontBuyStock(\x)]).
fact([-inFusion(\x,steel),+riskyCompany(\x)]).
fact([-closing(\x,\y),+riskyCompany(\x)]).
fact([-inFusion(\x,steel),-stong(steel),+notRiskyCompany(\x)]).
fact([+goodPrice(acme)]).
fact([+inFusion(acme,steel)]).
fact([+strong(steel)]).
fact([+closing(blockbuster)]).
fact([+goodPrice(blockbuster)]).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Preferred Structure:

trueSet([buyStock(acme), dontBuyStock(blockbuster)]).
```

```
falseSet([dontBuyStock(acme), buyStock(blockbuster)]).
protect([]).
heuristics([]).
```

A.4 capOf.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Capital of Theory:

fact([+cap_of(london,britain)]).
fact([+cap_of(edinburgh,britain)]).
%fact([+cap_of(tokyo,japan)]).
%fact([+cap_of(kyoto,japan)]).
fact([-cap_of(\x,\y),-cap_of(\z,\y),+eqq(\x,\z)]).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Preferred Structure:
trueSet([]).
falseSet([eqq(edinburgh,london), eqq(tokyo,kyoto),
          eqq(london,edinburgh), eqq(kyoto,tokyo)]).
protect([eqq, arity(eqq)]).
```

A.5 marriedWoman.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Main theory:

fact([-hasHusband(\x),+marriedWoman(\x)]).
fact([-hadHusband(\x),+marriedWoman(\x)]).
fact([-marriedWoman(\x),+notDivorcee(\x)]).
fact([+hasHusband(flor)]).
fact([+hadHusband(leticia)]).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Preferred Structure:

trueSet([notDivorcee(flor)]).
falseSet([notDivorcee(leticia)]).
protect([]).
heuristics([]).
```

A.6 motherhood.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Main theory:

fact([+mum(diana,william)]).
fact([+mum(camilla,william)]).
fact([-mum(\x,\z),-mum(\y,\z),+eq(\x,\y)]).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Preferred Structure:
trueSet([]).
falseSet([eq(diana,camilla), eq(diana,william), eq(camilla,william)]).

protect([eq,arity(eq),camilla, diana, william]).
```

A.7 researcher.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Main theory:

fact([-activeReasercher(\x),+writes(\x, papers)]).
fact([-writes(\x, papers),+author(\x)]).
fact([-author(\x),+emploee(\x)]).
fact([+author(\x),+person(\x)]).
fact([+activeReasercher(ann)]).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Preferred Structure:

trueSet([person(ann)]).
falseSet([emploee(ann)]).
protect([]).
heuristics([]).
```

A.8 superPenguin.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Main theory:

fact([-penguin(\x),+bird(\x)]).
fact([-superPenguin(\x),+penguin(\x)]).
fact([+superPenguin(opus)]).
fact([+brokenWing(opus)]).
fact([-brokenWing(\x),-bird(\x),+cannotFly(\x)]).
fact([-superPenguin(\x),+fly(\x)]).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Preferred Structure:

trueSet([fly(opus)]).
falseSet([cannotFly(opus)]).
protect([]).
heuristics([]).
```

A.9 superPenguin$_v 2.pl$

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Main theory:

fact([-penguin(\x),+bird(\x)]).
fact([-superPenguin(\x),+penguin(\x)]).
fact([+superPenguin(opus)]).
fact([+brokenWing(opus)]).
fact([+brokenWing(polly)]).
fact([+bird(polly)]).
fact([-brokenWing(\x),-bird(\x),+cannotFly(\x)]).
fact([-superPenguin(\x),+fly(\x)]).
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Preferred Structure:

trueSet([fly(opus), cannotFly(polly)]).
falseSet([cannotFly(opus), fly(polly)]).
protect([]).
heuristics([]).
```

A.10 swan.pl

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Main theory:

fact([-swan(\x),-european(\x),+white(\x)]).
fact([+german(bruce)]).
fact([+german(wolfgang)]).
fact([+swan(wolfgang)]).
fact([+swan(bruce)]).
fact([-german(\x), +european(\x)]).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Preferred Structure:
trueSet([black(bruce), white(wolfgang)]).
falseSet([white(bruce), black(wolfgang)]).
protect([]).
heuristics([]).
```

A.11 tweety.pl

```
fact([+penguin(lucy)]).
fact([+bird(polly)]).

fact([-bird(\x), +fly(\x)]).
fact([-penguin(\y), +bird(\y)]).
fact([-bird(\y), +feath(\y)]).

trueSet([feath(lucy), fly(polly)]).
falseSet([fly(lucy)]).
```

References

[Bundy & Mitrovic, 2016] Bundy, A. and Mitrovic, B. (February 2016). Reformation: A domain-independent algorithm for theory repair. Technical report, University of Edinburgh.

[Ceri *et al*, 1990] Ceri, Stefano, Gottlob, Georg and Tanca, Leitzia. (1990). *Logic Programming and Databases*. Surveys in Computer Science. Springer-Verlag, Berlin.

[Fu & Malik, 2006] Fu, Zhaohui and Malik, Sharad. (2006). On solving the partial max-sat problem. In Biere, Armin and Gomes, Carla P., (eds.), *Theory and Applications of Satisfiability Testing - SAT 2006*, pages 252–265, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Gärdenfors, 1988] Gärdenfors, Peter. (1988). *Knowledge in flux: modeling the dynamics of epistemic states*. MIT Press, Cambridge, Mass.

[Gómez *et al*, 2010] Gómez, Alejandro, Sergio, Iván, Carlos, Chesñevar and Ricardo, Simari Guillermo. (2010). Reasoning with inconsistent ontologies through augmentation. *Applied Artificial Intelligence*, 24(1-2):102–148.

[Herbrand, 1930] Herbrand, J. (1930). Researches in the theory of demonstration. In van Heijenoort, J, (ed.), *From Frege to Goedel: a source book in Mathematical Logic, 1879-1931*, pages 525–581. Harvard University Press, Cambridge, Mass.

[Ignatiev *et al*, 2018] Ignatiev, Alexey, Morgado, Antonio and Marques-Silva, Joao. (2018). PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT*, pages 428–437, Oxford, UK. Springer, Cham.

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

[Li *et al*, 2018] Li, Xue, Bundy, Alan and Smaill, Alan. (September 2018). ABC repair system for Datalog-like theories. In *10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, volume 2, pages 335–342, Seville, Spain. SCITEPRESS.

[Misyak *et al*, 2016] Misyak, Jennifer, Noguchi, Takao and Chater, Nick. (2016). Instantaneous conventions: The emergence of flexible communicative signals. *Psychological science*, 27(12):1550–1561.

[Mitrovic, 2013] Mitrovic, B., (2013). Repairing inconsistent ontologies using adapted reformation algorithm for sorted logics. UG4 Final Year Project, University of Edinburgh.

[Muggleton *et al*, 2012] Muggleton, S.H., Lin, D., Pahlavi, D. and Tamaddoni-Nezhad, A. (2012). Meta-interpretive learning: application to grammatical inference. In *Proceedings of the 22nd International Conference on Inductive Logic Programming*, Dubrovnik, Croatia. Springer.

[Rodler & Eichholzer, 2019] Rodler, Patrick and Eichholzer, Michael. (2019). On the usefulness of different expert question types for fault localization in ontologies. In Wotawa, Franz, Friedrich, Gerhard, Pill, Ingo, Koitz-Hristov, Roxane and Ali, Moonis, (eds.), *Advances and Trends in Artificial Intelligence. From Theory to Practice*, pages 360–375, Cham. Springer International Publishing.

[Strasser & Antonelli, 2019] Strasser, Christian and Antonelli, G. Aldo. (2019). Non-monotonic logic. In Zalta, Edward N., (ed.), *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Stanford, California, summer 2019 edition.

[Urbonas, 2019] Urbonas, Marius. (2019). A heuristic approach for guiding automated theory repair for the ABC theory repair system. University of Edinburgh UG4 Project Dissertation.

[Vasco M. Manquinho, 2009] Vasco M. Manquinho, Jordi Planes. Joao Marques Silva. (2009). Algorithms for weighted boolean optimization. In Biere, Armin and Gomes, Carla P., (eds.), *Theory and Applications of Satisfiability Testing - SAT 2009*, pages 495–508, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Wan *et al*, 2016] Wan, Hai, Zhang, Heng, Xiao, Peng, Huang, Haoran and Zhang, Yan. (2016). Query answering with inconsistent existential rules under stable model semantics. In *AAAI'16: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, page 1095–1101, Phoenix, Arizona,USA. AAAI.