Automating the Repair of Faulty Logical Theories

Xue Li

Doctor of Philosophy
Artificial Intelligence Applications Institute
School of Informatics
University of Edinburgh
2021

Abstract

This thesis 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 (ABC). Given an observed assertion and a current theory, abduction adds axioms, or deletes preconditions, which explain that observation by making the corresponding assertion derivable from the expanded theory. Belief revision incorporates a new piece of information which conflicts with the input theory by deleting old 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. But they have not previously been combined into one system. We are working on aligning these three techniques in ABC, which is capable of repairing logical theories with better result than each individual technique alone. Datalog is used as the underlying logic of theories in this thesis, but the proposed system has the potential to be adapted to theories in other logics.

Acknowledgements

I am extremely grateful to people who have helped me during this PhD project, which makes it an incredible journey in my life.

The support from my parents, Yuefu Li and Fengyun Shi, means a lot to me. They have always been generous and considerate. Thanks to their trust, I have more faith in completing this work. Meanwhile, my brother Wei Li is the silent supporter who always does what is good for me without many words.

I have been so lucky to be a PhD student of my experienced and supportive primary supervisor, Alan Bundy. As a supervisor, Alan has given me a great example of how to organise research work to approach high efficiency. Moreover, he provides valuable advice which is essential to improve the quality of my work. Also, Alan is open to discussion and supportive to the communication within and outside of our research group. There are more academic skills learnt from Alan, which are worthwhile for life-long learning and practising. Beyond a supervisor, Alan is a friend who is kind and always willing to help. He respects and listens to his PhD students. There is nothing one can expect from a great supervisor that cannot be found from Alan.

My second supervisors Ewen Maclean, Alan Smaill and Eugene Philalithis, supported at different stages. Although Ewen left the school after several months of my joining, he had been a good supervisor who is very supportive. For most of this project, Alan Smaill has been my second supervisor. Whenever I was stuck in a question, Alan Smaill was always willing to discuss and giving hints to guide me approaching the solution step by step. In the final stage of writing up, Eugene is very kind to supervise me. He provides lots of useful feedback which improve the quality of this thesis. Besides, he brings new ideas and views based on his philosophy background, which increases the diversity of our discussions and broaden my mind about the project.

Thanks should go to my thesis examiners Stefano V. Albrecht and Stephen H. Muggleton, who are experts and very kind to provide feedback on the content of this thesis and suggest potential extensions to explore in the future. With their help, the quality of this thesis and my research skills have progressed significantly.

I am also very grateful to the panellist of my annual review: Perdita Stevens and Vaishak Belle, who have provided insightful comments on my work. Another thank you to members in the institute of AIAI and DReaM research groups. Their seminars and the feedback that I collected when I reported my work had inspired me during the whole project. The institute of AIAI has been very supportive by not only providing opportunities for PhD students to travel for workshops and conferences but also valuing students’ opinions and addressing their advice as best as they can.

Friends are also important to me. Although Wang Wei and Geng Qian are far away from the UK, they are always there in my heart, and that makes me stronger. Special thank to Francisco Jos´e Quesada Real and Imogen Morris for their valuable feedback on the thesis and the great time that we spent together.

Declaration

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Hypothesis

Knowledge can be represented as logical theories, which play an important role in many AI tasks, e.g., question answering, planning, learning, reasoning and so on (Barr and Feigenbaum, 2014). However, logical theories can become faulty when incorporating new information, or when they are used in another system with a different representation, or even born to be faulty due to the mistakes of their developers. These faults are usually refer to inconsistency, incorrect and so on. In this thesis, the faults will be defined as the theory's insufficiency and the incompatibility in terms of describing the users' observations of the real world.

An example of a faulty theory is given in Example 1.1.1 below. It has four axioms saying that German is part of Europe, all European swans are white, and the swan named Bruce is German. One theorem is that Bruce is white. Imagine the user observes the fact that Bruce is black. In this scenario, the theory is insufficient to conclude that Bruce is black and incompatible with saying that Bruce is white. Thus it needs to be repaired.

Example 1.1.1. *Swan Theory.*

$$\begin{aligned}
German(x) &\implies European(x).\\
European(x) \wedge Swan(x) &\implies White(x).\\
&\implies German(Bruce).\\
&\implies Swan(Bruce).
\end{aligned}$$

The issue is how to repair a faulty theory. When comparing automatic repair techniques, domain experts are still the most reliable authority to do the job. This is because the techniques of theory repair usually overproduce and are not able to generate desired repairs in various forms as a human can. For example, abduction (Cox and Pietrzykowski, 1986) and belief revision (Gärdenfors, 2003) add or delete axioms but are not be able to modify the signature of the theory, while conceptual change via the more recent approach of reformation (Bundy and Mitrovic, 2016) operates on the signature but not axioms[1]. On the other hand, repair techniques usually require varying levels of human intervention from both domain experts and ontology engineers (Flouris et al., 2006).

However, even human work is not perfect. The domain experts may make mistakes or fail in repairing a faulty theory when the search space is too large to deal with for a human. Additionally, human resource and labour are limited so that it is hard and expensive to find experts covering all jobs. Therefore, the automation of the repair process is vital.

The first step of narrowing the gap between an automatic repair technique and the domain experts is that the former should be able to generate as many kinds of repairs as the latter do. This task includes developing novel repair operations and combining existing ones.

The task of automatically producing a wider range of repairs is the aim of this project. Among the existing repair techniques, abduction, belief revision and conceptual change via reformation are complementary: abduction adds new axioms, belief revision deletes conflicting axioms, while reformation changes the language of the theory. In this thesis, these three techniques are combined in one repair system, named the ABC Repair System (ABC), so it is capable of producing *best* repairs based on the benchmark of a logical theory's correctness, called the *preferred structure*, which describes users' observations, formally defined by Definition 4.2.3.

Best repairs are the sets of necessary repair operations for producing theories that are not only fault-free w.r.t. the benchmark, but also follow certain commonsense meanings of the predicates and constants in the theory. The repairs which cannot be produced by any of the three repair technique individually but can by their combinations are called the *hybrid*. The hybrid repairs tend to be better than non-hybrid ones in terms of embracing commonsense meanings, which can be seen from #H column in Table 8.2 in the evaluation chapter on page 167, where the most of gold standards of best repaired theories contain hybrid repairs. Imagine that the

[1]These repair techniques will be introduced in details in the rest of this thesis.

theory in Example 1.1.1 is faulty because that Bruce is a Germany resident while only European variety swans are white. In this scenario, best repairs that can fully represent that reason are hybrids, e.g., the one shown in Example 1.1.2, is the hybrid of reformation, which increases the arity of European and German, and abduction, which adds axiom $Black(Bruce)$.

For comparison, the original paper of Example 1.1.1 is (Gärdenfors, 1992), where their best repairs are deleting one of the original axioms. Then none of their repairs correctly describes the above scenario.

Example 1.1.2. *Swan Theory.*

$$\begin{aligned} German(x, y) &\Longrightarrow European(x, y). \\ European(x, Variety) \wedge Swan(x) &\Longrightarrow White(x). \\ &\Longrightarrow German(Bruce, Resident). \\ &\Longrightarrow Swan(Bruce). \\ &\Longrightarrow Black(Bruce). \end{aligned}$$

In principle, ABC can be used in any logic, because all of its three repair techniques have been applied to other logics, e.g., reformation has been applied to FOL (Bundy and Mitrovic, 2016). In this project, Datalog (Ceri et al., 1989) is chosen for the sweet-point between expressibility and feasibility[2]. Accordingly, the hypothesis driving this project is:

Hypothesis: by combining abduction, belief revision and conceptual change via reformation, ABC is capable of repairing Datalog theories with the best results within the scope of the repair operations based on all the individual techniques.

A fully repaired theory will be fault-free by respecting its preferred structure. And, in addition, a fully repaired theory is one of the *best results* if it is produced with non-redundant repair operations and embraces commonsense meanings. The satisfaction of preferred structure and the selection of the equally good repairs which solve the most faults with fewest operations are objective. Still, to go further, the heuristic measure of human judgement can check which repairs embrace commonsense meanings. Just like in chess, a move is objective good if it follows rules and leads to a win, but subjectively, a chess expert can judge whether a move is smart or not. Main challenges of this project include the conflicts between three candidate

[2]More details are discussed at the end of §3.1

techniques and the over-production of repairs, which are discussed more detailed in §2.3.5.

1.2 Organisation of Thesis

This thesis has nine chapters. The remaining chapters are briefly introduced here.

Chapter 2 is the literature survey where the state of the art w.r.t. our research is discussed in terms of automated reasoning; typical logics; theory repair techniques including abduction, belief revision, conceptual change and inductive logic programming; Max-Sat which provides a strategy to reduce the search space of the repair generation and machine learning which is a technique that some theory repair algorithms are based on. In this thesis, Max-Sat is not employed as our sub-optimal pruning algorithm but a non-SAT-based version.

Chapter 3 is the background of this project which introduces the prior work based on which our work is built, including: Datalog which is the logic used in this project; Herbrand structure, the foundation of the benchmark of the correctness of a theory; the selected literal resolution implemented as the automated theorem prover for Datalog; repair techniques to combine which are abduction, belief revision, conceptual change, and epistemic entrenchment, which can refine repairs.

Chapter 4 describes our original work which defines the faults to repair; discusses the fault detection; provides the postulates for guiding the design of repair process and analyses the candidate repair techniques. These analyses constitute the theoretical basis of the repair algorithm introduced in Chapter 5.

Chapter 5 introduces the algorithm for generating repair plans w.r.t. a fault in ABC, which is the key work in this thesis. The algorithm has two components: the repair plans generated based on a targeted resolution step and the analogical abduction based on an existing rule.

Chapter 6 is the original work which refines ABC's repair process by introducing the unique name assumption with exceptions (UNAE) to simplify the representation of the equalities and the inequalities; an algorithm to grade preference entrenchment of an axiom, a precondition in a rule, and a signature element (a predicate name and each of its arguments); the computation of maximal sets of commutative repair plans to reduce the search space of fault-free theories; the sub-optimal pruning based on estimated cost score to get rid of bad search branches of repairs, and a set of specifications and heuristics to assist the repair process.

Chapter 7 introduces the implementation of ABC in Prolog, where UNAE is built based on equivalence classes together with inference rules for equalities and inequalities; the representation of a proof (or the evidence of a partial proof) in ABC is built so that the original axiom which introduces a predicate name or a constant can be found by tracing back that proof. In the end, the repair generation algorithm is summarised based on that represented proof and evidence.

Chapter 8 evaluates the hypothesis. Seven test plans are performed to test the features of ABC. These evaluation results are analysed which shows the advantages and some limitations of our work.

Chapter 9 summarises our work by outlining the original work and the limitations of ABC followed by the suggestions for future work. In addition, some possible applications are discussed.

1.3 Contributions

The contributions of the original work involved in this thesis are summarised briefly.

- A repair system is established, which blocks unwanted proofs and builds wanted proofs by combining abduction, belief revision and conceptual change via reformation.

- Variants of abduction and belief revision have been developed. The former deletes a precondition from a rule to unblock a proof for a previously unprovable goal while the latter adds a new precondition to a rule to block an unwanted proof.

- An algorithm of analogical abduction has been established based on reasoning with a goal and the most relevant rule for proving that goal. The algorithm will retain as many existing preconditions as possible by only deleting unprovable ones. By combining with variant belief revision, new preconditions will be added if the produced rule is involved in unwanted proofs. As a result, the analogised rule unblocks a proof of the goal while not being involved in any unwanted proofs.

- Reformation has been supplemented with trace-back so that when the fault is caused by the signature element contained an unwanted precondition, the original axiom where that precondition comes can be traced back and then the repair can be generated to correct that element.

- Based on the common requirement of blocking proofs or unblocking proofs, a framework has been established which produces fault-free theories w.r.t. the preferred structure when it terminates by combining all candidates of repair techniques including abduction, belief revision and conceptual change via reformation.
- Sub-optimal pruning based on SL-Resolution has been developed to reduce the search space of repair generation.
- The unique name assumption with exceptions (UNAE) has been developed to simplify the representation of the equalities and the inequalities. Similar to the original UNA, inequalities do not need to be written explicitly in UNAE. Meanwhile, equalities are allowed in UNAE as exceptions.
- Based on the preferred structure, algorithms for scoring the entrenchment of axioms, preconditions in a rule, and signature elements including predicate name and arguments, have been developed. The entrenchment of axioms and preconditions reduce the search space of the fault-free theories, and the one for signature ranks the final fault-free theories.
- Some optional heuristics are provided to the user to add domain knowledge to the repair system. Consequently, repair operations on what is entrenched (and therefore should not be changed) can be avoided. These heuristics are domain independent, but their choices rely on the user's domain knowledge. They are optional because they only reduce the number of bad repairs but not the best ones.
- Evaluation has been done by employing faulty theories from relevant literature, based on which, the performance of ABC is evaluated from the perspectives of its ability to generate hybrid repairs, the quality of its output, the effects of sub-optimal pruning and optional heuristics, the influence of preferred structures that are consistent but in different sizes, the informational loss and its running time.

Together, these innovations allow the ABC system to find fault-free theories automatically and efficiently with belief revision, abduction, conceptual change, or hybrid repairs if they exist.

1.4 Publications

Some of the work in this thesis has previously been published, summarised as below.

1. Li, X., Bundy, A. and Smaill, A., 2018. ABC Repair System for Datalog-like Theories. In KEOD (pp. 333-340).

2. Urbonas, M., Bundy, A., Casanova, J. and Li, X., 2020, December. The Use of Max-Sat for Optimal Choice of Automated Theory Repairs. In International Conference on Innovative Techniques and Applications of Artificial Intelligence (pp. 49-63). Springer, Cham.

3. Bundy, A., Philalithis, E., and Li, X. (2020). Modelling virtual bargaining using logical representation change. In Machine Intelligence 21 workshop (pp. 1135–1149).

The first paper contains content from Chapter 4 and §5.1. The second paper is built based on an earlier version of ABC which only contains content from Chapter 4 and 5, where I am a co-author due to my contribution of providing the implementation of that ABC. My work in the last paper is about the application of ABC to the theories which model virtual bargaining games.

The algorithm of the variant belief revision and abduction, the analogical abduction and the whole Chapter 6 have not been published yet. They are currently being written up in a comprehensive journal paper.

The system code and the evaluation data are available at `https://github.com/XuerLi/ABC_Datalog`.

Chapter 2

Literature Survey

This chapter describes relevant literature to this project including automated reasoning (AR) in §2.1; typical logics in §2.2; theory repair techniques including abduction, belief revision, conceptual change, inductive logic programming, and a sub-optimal pruning method based on Max-Sat in §2.3, followed by applications of theory repair in Knowledge Graph (KG)s and programme debugging in §2.4. Basic ideas about machine learning and its differences from automated reasoning are discussed in §2.5. In the end, §2.6 summarises this chapter.

2.1 Automated Reasoning

Reasoning is one of the core cognitive capabilities of human beings. Before making a decision, we (ideally) evaluate the consequences of each option based on existing knowledge. For example, we select the best means of transport by considering their features, including efficiency, ticket price, traffic situation and so forth. These problems and information about the real world can be represented in a way that allows computers to utilise them. Then the software can reason automatically based on the logical theories, a set of inference rules and logics (such as first-order logic, non-monotonic logic and so on) (Wos et al., 1984). AR covers a broad range of applications including proving mathematical theorems (Paulson, 1990), developing reliable software (Benavides et al., 2005), robot planning (Zhang et al., 2014), query answering (Nuamah and Bundy, 2018) and so forth.

AR performs logical reasoning by being interactive or totally automated. By employing automated manipulations, e.g., inference rules, AR does its job based on logical theories which represents theorems in a formal language.

There are several limitations which make AR semi-automatic in real tasks. For example, there can be infinite branching, as with finding lemmas for inductive proofs; the logic it works on can be undecidable; the theorem prover can be incomplete which cannot find all existing proofs, due to time, space limits; or the input theory being not adequate to convey all necessary information, e.g., an input theory of incomplete flight information for a task of finding the best flight for the user.

2.2 Typical Logics

Logic plays an important role in both knowledge representation and reasoning including first-order logic (its decidable fragment: Datalog and description logic), higher-order logic (especially second-order logic) and non-monotonic logic. Those typical logics are briefly introduced in this section.

2.2.1 First-Order Logic (FOL)

First-order logic is also known as predicate logic, whose subsets include propositional logic, Datalog and most description logics as special cases (Van Harmelen et al., 2008).

The syntax of FOL will be introduced by listing the key components briefly, after which, the semantic interpretation of a FOL signature will be given.

The primitives of FOL include the following members (Smullyan, 1995), among which some complicated ones are giving by BNF expressions.

Variables: symbols e.g., X, Y[1].

Quantifiers: the universal and the existential quantifiers are $\forall$ and $\exists$ respectively. A variable is *free* if it is not in any scope of $\forall$ or $\exists$, otherwise, it is a *bound* variable.

Logical connectives: $\neg$ for not, $\wedge$ for and, $\vee$ for or, $\implies$ for implies, $\iff$ for if and only if.

Equality symbols: equality and inequality are denoted as $=$ and $\neq$ respectively.

A signature in first-order logic contains the following elements.

Function symbols: a function symbols maps individuals to individuals, whose arity is a non-negative integer.

[1]To be consistent with Prolog, variables in this thesis will start with uppercase while constants, functions and predicates with lowercase.

Predicate symbols: a predicate symbols maps individuals to truth values: true and false, whose arity is a non-negative integer.

Constant symbols: a constant symbols stands for an individual. It can be a function or a predicate of zero arity.

The components of a theory in FOL include terms, propositions, literals, formula, sentences and axioms. In FOL, a proposition is an atomic formula.

Term: a term is a variable, a constant or an n-place function, e.g., $f(t_1,t_2,...t_n)$ where f is a function symbol and $t_i,\ 1 \geq i \geq n$ are other terms. The BNF expression of a term is:

$$Term ::= ConstantSymbol|Variable|functionSymbol(Term_1,\ Term_2,\ ...,\ Term_n)$$

Proposition: if p is a predicate symbol, and $t_i,\ 1 \geq i \geq n$ is a term, then $p(t_1,t_2,...t_n)$ is a proposition.

Literal: a literal is either a positive proposition, e.g., $+p(t_1,t_2,...t_n)$ or a negative proposition, which is a negated proposition, e.g., $\neg p(t_1,t_2,...t_n)$.

Ground proposition/literal: a proposition/literal is ground if it does not contain any free variable.

Formula: a formula is constituted by propositions connected with logical connectives and/or quantifiers. Equation 2.1 is the BNF expression of a formula where $\mathcal{C}$ is one of the logical connectives includes $\neg,\ \wedge,\ \vee,\ \implies,\ \iff$ and $\mathcal{QV}$ are quantifier variables, e.g., $\forall X,\ \exists Z$. A formula is also called well-formed formula (WFF).

$$\begin{aligned} Formula ::= & Proposition|\neg Proposition| \\ & Proposition\ \mathcal{C}\ Proposition|\mathcal{QV}\ Proposition \end{aligned} \tag{2.1}$$

Sentence: if there are no free variables in a formula, then it is a sentence.

The meaning of a FOL formula is conveyed by an interpretation (I) which is a mapping between its signature and a non-empty universe (U), a.k.a. domain. Each constant symbol is mapped to an individual (object) in U, each predicate symbol to a property (unary predicate) or a relation (non-unary predicate) on U, each function symbol to a function on U, and then each sentence is assigned a truth value under I. If I assigns True to a sentence or all the axioms of a theory, then I is a model of that sentence or theory (Van Harmelen et al., 2008).

Inference rules for FOL include modus ponens, and-introduction, and-elimination, paramodulation, resolution and so on (Bundy, 1985), among which resolution is a basic component of ABC. The rule of a full resolution is given by Equation 2.2. Predicate $\equiv$ means 'is identical with'.

$$\frac{C' \vee P'_1 \vee \ldots \vee P'_m \qquad C'' \vee \neg P''_1 \vee \ldots \vee \neg P''_n}{(C' \vee C'')\phi}\ Resolution \tag{2.2}$$

where a *most general unifier* of $P'_1 \vee \ldots \vee P'_m$ and $P''_1 \vee \ldots \vee P''_n$ is ϕ:

$$\phi = \phi_1\phi_2 \tag{2.3}$$

$$(P'_1 \vee \ldots \vee P'_m)\phi_1 \equiv (P''_1 \vee \ldots \vee P''_n)\phi_2 \tag{2.4}$$

In automated reasoning, FOL formulae are converted into an equivalent conjunctive normal form (CNF) where *skolemization* is employed to remove existential quantifiers by introducing new constants and functions called Skolem constants and Skolem functions respectively (Bundy, 1985). CNF is a widely used form not only because it is the form for many inference rules, e.g., SL-resolution, and the practical satisfiability algorithms (a.k.a. SAT solvers), but also because it is natural to describe problems with a conjunction of preconditions or rules where a conjunction of preconditions imply a proposition (Van Harmelen et al., 2008).

From the view of theorem proving, FOL is semi-decidable. Although all theorems of a FOL theory can be proved with an exhaustive search, there is no limit to the search space so a conjecture cannot be concluded as unprovable even if its proof is not found (Smullyan, 1992).

There are decidable fragments of FOL, including Description Logic introduced in the next section and Datalog in §3.1. Because Datalog is the logic on which the theory to repair is based in this thesis, so it is introduced as a part of background.

2.2.1.1 Description Logic

Description logics (DLs) are a family of decidable fragments of FOL, which are the basis for widely used ontology languages including W3C Web Ontology Language (OWL) (Antoniou and Van Harmelen, 2004). Domain knowledge is described based on its concepts (a.k.a. classes), roles (a.k.a. relationships), and individuals in DL

(Baader et al., 2017). This section follows the upper/lower case convention in DL literature which is the opposite to the convention in the rest of this thesis.

The basic DL is the attributive concept language with complements (ALC) (Schmidt-Schauß and Smolka, 1991). The course theory $\mathbb{T}_c$ in Example 2.2.1 (Baader et al., 2017) gives some ACL-concepts including concept names ($\mathcal{N}_c$): teacher, person, student; role names ($\mathcal{N}_r$): teaches, attends and individuals: Mary and Cs600. Its equivalent FOL formulae is given in Example 2.2.2.

Example 2.2.1. *Course Theory $\mathbb{T}_c$ in ACL.*

$$\begin{aligned} Teacher &\equiv Person \exists teaches.Course,\\ Student &\equiv Person \exists attends.Course,\\ &\exists attends.\top \sqcup \neg Student,\\ &Mary : Person,\\ &Cs600 : Course,\\ &(Mary, Cs600) : teaches. \end{aligned}$$

Example 2.2.2. *Equivalence FOL Formulae of ACL-concepts in $\mathbb{T}_c$.*

$$\begin{aligned} &\forall x\, (Teacher(x) \iff Person(x) \wedge \exists y\, (teaches(x, y) \wedge Course(y)),\\ &\forall x\, (Student \iff Person(x) \wedge \exists y\, (attends(x, y) \wedge Course(y)),\\ &\forall x\, (\exists y\, (teaches(x, y)) \implies \neg Student(x)),\\ &Person(Mary),\\ &Course(Cs600),\\ &teaches(Mary, Cs600). \end{aligned}$$

An ALC knowledge base is a pair of a *terminological box* (TBox), which is a finite set of concept inclusion axioms, and an *assertional box* (ABox), which is a finite set of assertional axioms (Baader et al., 2017).

In Example 2.2.1, the TBox is constituted by the first three statements and the ABox by the last three. When a TBox only contains definitions, it is definitorial, e.g., a TBox constituted by the first two statements in Example 2.2.1.

By separating the TBox and ABox, some reasoning processes can only deal with one of them when the other is not relevant. Since the TBox usually plays an important role in deciding the complexity of the reasoning so the separation highlights that key part.

DL has been widely used in the field of knowledge representation (Consortium, 2019; Sanfilippo et al., 2019; Janowicz et al., 2019) and reasoning with database schema and queries (Kendall and McGuinness, 2019; Chu et al., 2020; Koopmann, 2019). There are also many computational services for ontology tools and applications developed based on DL (Knublauch et al., 2004; Liebig and Noppens, 2004; Rector et al., 1993; Visser et al., 2002).

Because a Description Logic (DL) concept has arity one and role arity two and those are not changeable, reformation's repair of changing the arity of a predicate, introduced in §3.4.3, will break the TBox and the ABox of a DL theory. Thus, DL is not the employed logic in this research, although it is decidable.

2.2.2 Higher-Order Logic (HOL)

First Order Logic allows variables ranging over objects but not over functions or predicates. In HOL, functions and predicates can be represented by variables and quantified. The HOL Prover (Gordon and Melham, 1993) is interactive so that the user can give some guidance for the search of proofs. A more automatic prover is TPS, which is equipped with the component search to handle larger search space (Bishop, 1999). Other well known HOL proof tools include Isabelle HOL (Nipkow et al., 2002), PVS (Owre et al., 1996), Nuprl (Constable et al., 1986) and so on.

HOL is more expressive than FOL and proofs of HOL are usually shorter than of FOL. But higher-order unification is considerably more complex than first-order unification so that HOL provers are more complicated than FOL ones. For example, there could be infinite independent most general unifiers for unifiable terms so the most general unifier is not unique (Qian, 1993; Goldfarb, 1981).

2.2.3 Non-Monotonic Logic

In a non-monotonic logic (NML), logical consequences are not monotonic, e.g., adding a new axiom can invalid old theorems (McDermott and Doyle, 1980). NMLs are designed to employ *defeasible inference* where rules can have exceptions or there are sub-classes that follow a different rule than others, so the truth of premises does not always guarantee the truth of the conclusion (Strasser and Antonelli, 2019a).

When there are conflicts among rules, preferences play an important role in deciding which rule should be applied. A common preference is that the most specific subclass dominates more general ones (Delgrande et al., 2004). For example, from rules: *all birds fly; penguins are birds; penguins do not fly* and assertion: *tweety is a bird*, we can conclude: *tweety flies*. However, if there is an extra assertion: *tweety is a penguin*, then *tweety*'s most specific subclass is *penguin* rather than *bird*. Consequently, it can be concluded that *tweety cannot fly*.

One issue of NML is that it is hard to formalise a theory which has intended conclusions (Nebel, 2001). When preferences or conflicting rules are complicated, unwanted logical consequences may be derived. By contrast, the repair technique of reformation distinguishes sub-classes or exceptions by adding new argument to the relevant predicate so that different rules can be applied based on the new label argument. For example, the *tweety* example will be repaired by given predicate *bird* an extra argument, e.g., *normal* and *abnormal*. Then the rules become *all normal birds fly and penguins are abnormal birds*. Thus, it is not a theorem that *penguins fly*. Consequently, reformation does NML's job on a monotonic theory.

2.3 Theory Repair Techniques

The repair techniques that are combined in this thesis are introduced in this section, including abduction, belief revision and conceptual change via reformation. Additionally, inductive logic programming is described briefly. At the end of this section, the relationship between these techniques is discussed.

2.3.1 Abduction

Abduction finds the hypotheses that explain a given set of observations. After adding these hypotheses to the original theory, the observations can be logically entailed (Sakama and Inoue, 2003). These hypotheses are called *explanations* sometimes. To evaluate the explanations, (Thagard, 1978) and (Hobbs et al., 1993a) propose the criteria of consilience, simplicity, consistency and analogy, while analogy is a particular type of abduction rather than a criterion in some literature (Duval, 1991; Schurz, 2008). As the first three criteria are more basic, they are briefly introduced below.

- Consilience corresponds to a theory's explanatory power. Thus, a theory is more consilient if it proves more facts[2] than another.

[2] The original definition of consilience is based on classes of facts rather than facts. But the notion of a class of a fact relies on the background knowledge in a particular domain, we omit it in our general introduction.

- Simplicity is based on the size of the explanations, e.g., the number of the axioms to add, with the smaller being the better.
- Consistency ensures that the theory has no contradictions after expanding the explanation.

Besides theory repair, abduction also applies to a wide variety of reasoning tasks, e.g., the medical diagnosis of a patient's symptoms (Pople, 1973; Reggia et al., 1983); the interpretation of why a sentence is said in natural language (Hobbs et al., 1993b).

In the field of view update in database theory, an abductive framework is developed in (Sakama and Inoue, 2003) which not only works on finding the positive explanation of observations but the negative ones which are the axioms to be removed to explain observations, which is important for theories in NML. In (Fernandez et al., 1996), all possible models from the base of observations are computed, then minimal models that satisfy observations are constructed from those models.

2.3.2 Belief Revision

There are three generally considered kinds of belief changes[3]: contraction, expansion and revision (Hansson, 2003).

- Contraction: some beliefs are retracted to obtain a consistent belief system.
- Expansion: new beliefs are added to expand the beliefs with no guarantee of consistency after the operation.
- Revision: a new belief is added and some old ones may be retracted if it is necessary to keep the belief system consistent.

Considering whether the justification for these beliefs should be considered or not, belief revision research is divided into two groups: foundations and coherence approaches. If there are no justifications for a belief, then it should not be accepted in the foundations approaches. While in coherence approaches, one belief is acceptable as long as it logically coheres with others (Doyle, 1992).

Belief revision can be seen as a function from one belief state into another. How is a belief state represented? The simplest way is as a (possibly infinite) set of sentences K, which satisfies the integrity constraint that K should have all the logical consequences

[3]A sentence in the belief system is considered to correspond to an axiom or a theorem in a logical theory in this thesis.

in it (De Raedt and Bruynooghe, 1992; Gärdenfors and Makinson, 1988; Katsuno and Mendelzon, 1991). In formal epistemology, logically closed sets are called "corpora", "knowledge sets", or (more commonly) "belief sets" (Keeler and Priss, 2013). Another way of representing belief states is as a belief base which is a finite set of axioms that are not closed under logical consequences. Based on a belief base, the additional theorems (the belief set K) can be derived (Fuhrmann, 1991). Then, revision and contraction functions defined on belief bases will be called base revisions and base contractions, respectively. Compared to belief sets, belief bases seem easier to handle in computer science since they are usually finite structures (Gärdenfors, 2003).

Since the 1980s, several algorithms for the implementation of belief revision have been proposed by (Nebel, 1991; Alchourrón and Makinson, 1985; Jin and Thielscher, 2008) and so on. In addition, some researchers have incorporated belief revision within different logics, such as description logic (Lee and Meyer, 2004), and conditional beliefs logic (Dubois et al., 1994). As unaided computer system cannot understand the semantics of theories, epistemic entrenchment (EE) is proposed for prioritising axioms which evaluates the overall informational value of axioms (Gärdenfors, 1988). There is a constructive approach which is adopted based on EE (Gärdenfors and Makinson, 1988). As EE plays an important role in this thesis, it is introduced in more detail in §3.4.2.

2.3.3 Conceptual Change

In the field of psychology, conceptual change studies the process that the learner substantially revises prior knowledge and acquires new concepts (Wrobel, 1994). In this thesis, conceptual change refers to a technique of logical theory repair based on algorithms rather than in psychology. Conceptual change here modifies the signature in which the logical theory is written in the way of invent new predicates, constants or modifying old ones (Bundy and Mitrovic, 2016).

Several relevant systems based on AR have been developed, which contain the relevant algorithms, including the ORS (Ontology Repair System) (McNeill and Bundy, 2007), the GALILEO System (Guided Analysis of Logical Inconsistencies Leads to Evolved Ontologies) (Lehmann et al., 2013), and the Hyper System with a form of knowledge reformation method (Prendinger et al., 2000). However, the first two algorithms are domain-dependent and the last one only reforms function-free FOL acyclic Horn theories to their logically equivalent propositional theories by introducing new predicates, which does not aim at repairing a faulty theory but rewriting a theory. MIL is also a system which conduct conceptual change by inventing new predicates, which is introduced in §2.3.4.

Reformation is an algorithm of conceptual change based on AR, which changes the signature of a logical theory for blocking or unblocking proofs (Bundy and Mitrovic, 2016), which is introduced in more detailed in §3.4.3. It is particularly powerful because it does not only invent new predicates as other techniques do, it can also merge existing predicates according to the evidence of faults globally, or rename one occurrence of a predicate name or individual name locally, or change the arity of a predicate.

2.3.4 Inductive Logic Programming (ILP)

Based on inductive inference, ILP derives logic programs based on given observations including the positive and negative ones and background knowledge, which employs techniques from both machine learning (ML)[4] and logic programming (Muggleton and De Raedt, 1994). It is easier to apply ML algorithms in ILP than others, e.g., LISP programs, because programmes in ILP are written in clausal logic so that their axioms and literals can be changed independently without considering orders.

Given a set of positive examples E^+; a set of negative example E^- and background knowledge $\mathbb{T}$, ILP adds a hypothesis H to $\mathbb{T}$ so that:

$$\forall\alpha \in E^+,\ \mathbb{T} \cup \{H\} \vdash \alpha \tag{2.5}$$

$$\forall\beta \in E^-,\ \mathbb{T} \cup \{H\} \nvdash \beta \tag{2.6}$$

Different degrees of satisfiability are defined to describe how much a hypothesis follows the positive examples while avoiding the negative ones (Muggleton and De Raedt, 1994). Also, induction is claimed to be a process of combining abduction and justification. Here justification refers to evaluating the confidence of candidate hypotheses (Muggleton and De Raedt, 1994).

In (Shapiro, 1982), the debugging of a Prolog bug is investigated employing induction inference. A broken branch is found by examining the truth value of subgoals in a top-down process, where only the true subgoals are carried for further debugging. It is an interactive debugging system which addresses restricted types of bugs, while ABC automatically repairs faults without type restrictions.

[4]Machine learning will be briefly introduced in §2.5.

Among numerous ILP algorithms, MIL is particularly powerful which can achieve automatic conceptual change in the way of introducing new predicates and also decompose a concept to, e.g., introduce sub-definitions (Muggleton, 2015; Muggleton, 2017). Given meta-rules $\mathbb{M}$, background knowledge $\mathbb{B}$ and ground positive and negative examples, MIL finds the hypothesis $\mathbb{H}$. Then the positive examples become derivable from $\{\mathbb{M}, \mathbb{B}, \mathbb{H}\}$ while the negative examples are not derivable.

2.3.5 The Relation Among Repair Techniques

Deduction, induction[5], and abduction (Hintikka, 1999) are three forms of inference. Taking rule R1 as an example, deduction starts from knowing R1 and the truth of $p(c)$ then concluding the truth of $q(c)$, while induction starts from knowing $p(c)$ and $q(c)$, usually multiple $p(c)$s and $q(c)$s, then summarises R1 as the result for describing their relations based on statistical data. When $q(c)$ is observed, abduction seeks for its cause (explanation) which is $p(c)$ when R1 is known or R1 when $p(c)$ is known.

$$p(X) \Longrightarrow q(X) \tag{R1}$$

Induction and abduction are *ampliative* in the way that the consequence goes beyond its premise (Douven, 2017).

The common ground of abduction, belief revision, conceptual change and ILP is that they do their job by blocking unwanted proofs and/or unblocking wanted proofs. On the other hand, their differences include their repair operations and their default inference of the repair generation.

Different literature have different views of each technique so that there are overlaps among them especially when they serve similar tasks, e.g., forwarded induction reasoning is interpreted based on the process of belief revision in (Battigalli and Siniscalchi, 2002). To distinguish these techniques from the perspective of theory repair in this thesis, belief revision is considered to be a technique which blocks unwanted proofs; abduction unblocks wanted proofs and these two techniques change a literal or an axiom as a whole. As for reformation, it modifies the signature of the theory, which can do both blocking and unblocking.

In this thesis, abduction, belief revision and reformation are the techniques to combine because they are complementary and all of them can be done based on AR. Since reformation's ability to change signature cannot be replicated by any other

[5]The induction in this thesis is not the mathematical induction, which is a form of deduction.

technique currently to our knowledge, e.g., it can modify the arity of a predicate, a repair system which combines reformation and reformation's complementary techniques of abduction and belief revision can generate richer repairs and then narrows the gap between the automatic theory repair techniques and domain experts.

The main challenges of their combination include the conflicts between three candidate techniques and the over-production of repairs. The former occurs in particular scenarios. For example, a signature change from reformation is applied by updating the axioms which are written in the changed signature element, e.g., a predicate. If those axioms include ones that should be deleted by belief revision, then belief revision won't find these axioms after reformation is applied. Similarly, reformation and abduction should not be applied together when abduction adds an axiom that contains a predicate of two arguments and reformation changes that predicate by increasing its arity into three, then that predicate will be overloaded by having arities of both two and three. On the other hand, abduction and belief revision can cause a loop when the former adds an axiom to unblock a proof while the latter deletes that axiom to block another proof. These issues are avoided by calculating the maximal set of commutative repair plans so that applying them together won't cause conflicts, introduced in §6.3. The latter challenge is tackled by resembling epistemic entrenchment: calculate the entrenchment of axioms, signature, preconditions and then only change the least entrenched ones, introduced in §6.2. Furthermore, repairs are evaluated as either the optimal or the sub-optimal based on the number of the faults remaining in their corresponding repaired theories. In the end, only the optimal ones are provided as the final result, introduced in §6.4.

There is one further relevant repair technique in the recent literature: the aforementioned Meta-Interpretive Learning (MIL) in the field of ILP. The key advantage of MIL is that it can both create new predicates and induce appropriate rules for using these new predicates. Moreover, although it employs ML, it can learn from limited examples via logical inference rather than require a large training dataset. And MIL is also an AR-based system (Muggleton, 2015), which fits the inference base of our system. However, MIL is different from the repair techniques that we have chosen in that it only adds new axioms while not changing any old one. As a result, from the perspective of conceptual change, MIL cannot merge two existing predicates whereas reformation can. Consequently, MIL is not suitable in the scenario when the user wants to repair the input theory rather than purely expand it. We do not further consider ILP or MIL in this thesis, but it is worthwhile to explore as a part of the future work because it is a complement of our work: where our method only create new predicate MIL instead can also induce appropriate rules for using new predicates.

The difference of summarising a rule by induction and adding a rule by abduction, is that the former is more based on statistics, while the latter aims at the best explanations (Douven, 2017). For example, the grass is wet in the morning of most of the days in Scotland. Then the grass will be wet tomorrow morning would be induced based on the statistics, while most evenings are rainy in Scotland can be abduced as the explanation of the wet grass in the morning.

2.3.6 The Sub-optimal Pruning based on Max-Sat

Similar to many repair techniques, our combination repair mechanism suffers the common issue of overproducing repairs. Marius Urbonas has applied the Partial Max-Sat algorithm to prune sub-optimal repairs (Urbonas et al., 2020), whose work flow is shown in Figure 2.1.

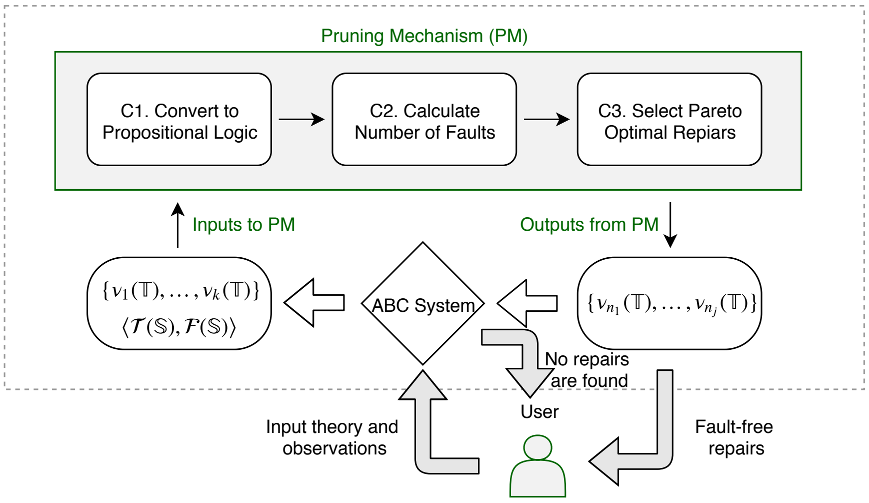

Pruning Mechanism (PM)
C1. Convert to Propositional Logic
C2. Calculate Number of Faults
C3. Select Pareto Optimal Repiars
Inputs to PM
Outputs from PM
$\{\nu_1(\mathbb{T}), \dots, \nu_k(\mathbb{T})\}$ $\langle \mathcal{T}(\mathbb{S}), \mathcal{F}(\mathbb{S}) \rangle$
ABC System
$\{\nu_{n_1}(\mathbb{T}),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

Figure 2.1: The Partial Max-Sat sub-optimal pruning system whose components are C1, C2 and C3.

In Figure 2.1, ABC is the repair mechanism which is introduced in Chapter 4 and 5. Note that it does not include any refinement introduced in Chapter 6. The input Datalog theory is $\mathbb{T}$ and the set of suggested repairs by ABC is $\{\nu_1, ..., \nu_k\}$. Here the observation of the environment is formalised in $\langle \mathcal{T}(\mathbb{S}), \mathcal{F}(\mathbb{S}) \rangle$ and the output of the Max-Sat pruning system is a set of optimal repairs.

The main work of the Max-Sat sub-optimal pruning system is to compare the remaining faults after applying each Repair Plan (RP) to the faulty theory and then only retain the optimal ones which do not have more remaining faults than others from the perspectives of both fault types that we considered.

This sub-optimal pruning should only be employed to evaluate the theories which have applied the same number of RPs. Otherwise, it is unfair to compare the remaining fault number if one theory $\mathbb{T}_1$ is generated based on three RPs while another $\mathbb{T}_2$ is based on only one RP. In that example, if they are the only candidates to compare and have the same number of remaining faults, then the comparison mechanism in (Urbonas et al., 2020) will conclude that both are optimal while $\mathbb{T}_1$ should not be.

In addition, (Urbonas et al., 2020) relies on an external SAT algorithm by (Fu and Malik, 2006), which requires the translation of the object theory from Datalog to propositional logic. As SL Resolution has been implemented in ABC Repair System (ABC), which is capable of evaluating whether a repair is optimal but does not need the translation process. Therefore, resembling the idea of Urbonus's work, SL Resolution is employed to support the sub-optimal pruning mechanism introduced in §6.4.

2.4 Theory Repair in Knowledge Graphs

Knowledge engineering is a popular topic in AI which includes the Semantic Web, Linked Data, data integration and so on. It has been explored how to automatically detect faults in a knowledge represented database, e.g., Knowledge Graph (KG), but how to repair such a faulty database is an open question currently.

In the field of KGs, fault detection, especially the inconsistency check, has become a hot research topic for decades, especially in the sub-fields of entity embedding and quality assessment. For example, the quality of links between entities from multiple datasets are evaluated by a combination of graph metrics (van Harmelen et al., 2018; Raad et al., 2018) and the inconsistencies among RDF triples w.r.t. their schema are detected by schema aware triple classification (SATC), where the fault patterns are formalised as logical rules (Wiharja et al., 2018).

But there is no solution of how to repair a detected inconsistency automatically. The difficulties include the lack of a gold standard (GS) because a ground truth database is often not available; the involvement of manual work which is prone to error and cumbersome for large scale database; and the lack of experts with essential domain knowledge.

To tackle that issue, (Dimou et al., 2015) has applied fault detection to the embedding algorithm rather than the final RDF triples that embedding algorithm generates. Here the embedding algorithm is considered as the theory to repair. When the embedding algorithm is fault-free, it will generate fault-free RDF triples, where the fault-free is in terms of rules given in the detection process. Their work shows that the earlier the fault detection and repair is applied, the better. This work is limited because it is only applicable to RDF triples so they require that the embedding algorithm has to be written as RDF triples as well. Meanwhile, their repair is semi-automatic so that there will be a huge amount of manual work required to apply their method on a large-scale embedding algorithm.

Because the published KGs are usually in a large scale, even an automatic inconsistency check can be quite time consuming and the repair process still requires manual work currently. Thus, it is nearly an impossible task to repair all of the detected faults in KGs at the current stage. The reason why the existing automatic repair techniques are not employed is that they only suggest repairs in limited form and they overproduce at the same time (Li et al., 2018). Consequently, even if inconsistencies are detected, they will usually continue to be included in KGs, which means that fault detection is not incorporated in KGs' construction or update.

Our work in this thesis aims at providing repairs in richer form which will narrow the gap between what an automatic repair technique can provide and what a repair agenda in KGs calls for.

2.5 Machine Learning (ML)

Machine learning is a sub-field of AI but is often also referred to as predictive analysis or modelling. A general goal of ML is to develop algorithms so that patterns in data can be learnt/discovered, and then predictions can be made accordingly (LeCun et al., 2015). Some ML techniques employ background knowledge to learn from limited examples rather than just learn statistically from large samples (Muggleton, 2015). In this section, ML refers to the traditionally statistical ML techniques over large samples.

In ML, conceptual change refers to problems of concept formation where a new predicate can be invented for aggregating a set of concepts (Wrobel, 1994). But it can neither modify the name nor the arity of existing ones as (Bundy and Mitrovic, 2016) does. Since ML learns from the training dataset, it can only summarise existing patterns, while ABC create novel patterns that may not follow a prior one, but are the best fit for the given observations. More reasons why ML is not used in ABC are discussed in detail in the following section.

2.5.1 Relationship between AR and ML

The main difference between AR and ML is that the former starts from a theory and then infers the consequences by employing theorem provers while the later needs to learn the 'theory' by applying the ML algorithm on a training data and then use the learnt 'theory' to compute its conclusion on test data.

As the theory in AR is given, symbols in their signatures usually follow commonsense, or otherwise pre-established meanings so that theory is explainable. However, the 'theory' learnt from ML could lack rich semantics. To solve that issue, scientists working on the field of explainable AI are trying to find solutions (Samek et al., 2019).

AR can be a good choice when the theory of the object data is known or can be formalised in a logic, e.g., for a small set of observations. Alternatively, ML is preferred when the pattern of the data is not clear to the user, e.g., a complex pattern among the data or the data is of a large size and high quality. Good quality data is one of the key factors enabling an ML algorithm to capture accurate features from the data.

In some tasks, AR and ML have been combined. For example, use ML techniques to assist AR in theorem provers(Bridge et al., 2014; Urban et al., 2011), e.g., ML is used to suggest which lemmas to include in a proof attempt. AR is incorporated to make ML explainable (Bride et al., 2018).

In this thesis, AR is used but not ML for the following reasons. We define the gold standard (GS) as the desired repairs which describes scenarios that make sense in the real world. Since those scenarios are various as there are dynamic possibilities in the real world, it is manual work to decide the GS. For small theories, it is still possible to manually search for GS. But it is much harder to do it manually for large theories.

1. The algorithms of repair techniques we combined are explicit, so it is not needed to learn them from examples by ML.

2. The open question of GS decides that ML is not employed because GS is an essential part of the training set for ML techniques.

3. Assume that GS is available. If we want to use ML to learn repair plans, it is hard to find high-quality training data of fault theories which can cover all patterns of repair plans.

Therefore, AR is the basic technique of the ABC Repair System in this research.

2.6 Summary

This section briefly introduces relevant literature to our project including AR in §2.1; typical logics in §2.2; the repair techniques: abduction, belief revision, conceptual change, ILP and the sub-optimal pruning based on the Max-Sat algorithm in §2.3; the current stage of repairing a faulty KG in §2.4 and ML in §2.5. It can be seen that only if a repair system can generate better repairs, the gap between its application requirements and its capability can be narrowed. By analysing the relationship among those repair techniques, our ABC Repair System can narrow that gap by combining abduction, belief revision and reformation.

Chapter 3

Background

This chapter will introduce the background of our research about the ABC Repair System (ABC). First of all, Datalog will be introduced in §3.1. In this project, the theory to repair is based on Datalog, which is a decidable logic, so a proof can be found if it exists. Herbrand structures are introduced in §3.2, on which the benchmark of the correctness of a Datalog theory can be formalised. In ABC, a selected-literal resolution is implemented for Datalog theories, which is discussed in 3.3. To repair a faulty Datalog theory, our repair mechanism combines abduction, belief revision, and a conceptual change via the reformation algorithm[1], which will be discussed in §3.4. The summary of this chapter is given in §3.5.

From now on, all constants and predicates start with lowercase while variables start with uppercase. The logical connectives are represented as $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction) and $\implies$ (implication). We use $\mathbb{T} \vdash \alpha$ to represent that the formula α is provable within the theory $\mathbb{T}$ and $\mathbb{T} \nvdash \alpha$ to show that α is not a theorem of $\mathbb{T}$.

3.1 Datalog

Datalog is a declarative logic programming language in first-order logic (FOL), which has resurged in the database community in recent years (Ceri et al., 1989). As a decidable subset of FOL (Ceri et al., 1989; Pfenning, 2006), Datalog is used as the logic of this thesis, e.g., a piece of Datalog program is seen as a logical theory based

[1]These repair techniques cannot be directly integrated as originally they were implemented in a different context. Therefore, they are revised to be incorporated into our framework. Also, we have developed novel repair plans for belief revision and abduction respectively, e.g., splitting a rule. These original works will be introduced in Chapters 4 and 6.

on Datalog logic.

In comparison with FOL, Datalog excludes negations, non-nullary function symbols, and existential quantification. Assertions and rules in Datalog are formalised as Horn Clauses, which is a clause that has at most one positive literal. In this thesis, clauses are written in Kowalski normal form (Bundy, 1985).

$$\neg Q_1 \vee \ldots \vee \neg Q_m \vee R_1 \qquad \text{(Disjunction Form)}$$

$$Q_1 \wedge \ldots \wedge Q_m \implies R_1 \qquad \text{(Kowalski Normal Form)}$$

The grammar of a Datalog theory is given by Definition 3.1.1. Note that a constraint axiom is a rule without head, which is in the same form as a goal clause.

Definition 3.1.1 (Grammar of Datalog Logic)**.**

$$\begin{aligned}
Term &::= Constant|Variable \\
Proposition &::= Predicate(Term_1, \ldots, Term_n) \\
Assertion &::= \implies Proposition \\
Rule &::= Proposition_1 \wedge \ldots \wedge Proposition_m \implies Proposition \\
Constraint\ Axiom/Goal\ Clause &::= Proposition_1 \wedge \ldots \wedge Proposition_m \implies \\
Empty\ Clause &::= \implies
\end{aligned}$$

where $m \in \mathbb{Z}^+$, $n \in \mathbb{N}$, *i.e., n might be* 0.

For example, $fly(X) \wedge nonfly(X) \implies$ is a constraint axiom, which says that an individual cannot be both flyable and non-flyable at the same time. Similar to the integrity constraints in databases (Fan and Siméon, 2003), it can be fundamental for constraint axioms to express semantic constraints without negation in Datalog. Without constraint axioms, a Datalog theory is inherently consistent. Otherwise, inconsistencies can be caused by the violation of constraint axioms. That potential fault will be discussed in detail in the section of fault detection §4.3.

There are different variants of Datalog. Our project is restricted to the basic Datalog, where a Datalog theory should satisfy the following *safety conditions* (Ceri et al., 1989), which guarantee that all assertion theorems of a Datalog theory are ground and the number of them is finite.

1. Each assertion does not contain any variables.

2. Each variable which occurs in the head of a rule also occurs in the body of the same rule.

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 because the occurs check (Bundy, 1985) is not needed as there is no function nesting, and the reformation algorithm is greatly simplified, as will be introduced in §4.4.3.2. In fact, Datalog has been applied successfully in a wide variety of problem domains (Bárány et al., 2017; Kaminski et al., 2016; Motik et al., 2019; Shkapsky et al., 2016).

An example of a Datalog theory is given below. In Example 3.1.1, four axioms say that Germany is part of Europe and all European swans are white and Bruce is a German swan. One theorem of this theory is that Bruce is white. Imagine that Bruce is black based on the user's observation . In this scenario, the theory is faulty, so it needs to be repaired.

Example 3.1.1. *Swan Theory.*

$$german(X) \implies european(X) \quad (A1)$$

$$european(X) \wedge swan(X) \implies white(X) \quad (A2)$$

$$\implies german(bruce) \quad (A3)$$

$$\implies swan(bruce) \quad (A4)$$

Developing our repair process based on a basic version of Datalog allow us applications of databases and various extensions in future work. Datalog was designed to interact with large scale databases. The optimisation methods for various types of Datalog rules and their efficiency have been studied (Bancilhon and Ramakrishnan, 1989; Ullman, 1985; Bancilhon and Ramakrishnan, 1988). There are various extended Datalog variants for knowledge representation and reasoning. For example, Disjunctive logic programming (DLP) is a system for knowledge representation and reasoning that is suited to deal with larger amounts of input data whose kernel language is the disjunctive Datalog (Leone et al., 2006). Existential rules is a paradigm for query answering over ontologies, which extends Datalog with value invention so that the existence of a new individual can be inferred from a given situation (Calì et al., 2012). Vadalog system is an implementation of Warded Datalog+/-, which is designed to convey complex logic reasoning, e.g., in knowledge graphs (Bellomarini et al., 2018).

In summary, Datalog theories are decidable but they are still sufficiently expressive to allow a wide range of practical applications.

3.2 Herbrand Structure

A Datalog theory's signature is constituted by predicates, constants and variables. For simplification, a signature will be written as a triple $\langle \mathcal{P}, \mathcal{C}, \mathcal{V} \rangle$, where $\mathcal{P}$, $\mathcal{C}$ and $\mathcal{V}$ are finite sets of predicates, constants, and variables respectively.

The Herbrand universe is a set of all possible expressions of ground terms in FOL (Herbrand, 1930). As there are no non-nullary functions in a Datalog theory, the Herbrand universe is simplified to be the set of the constants of that Datalog theory.

Definition 3.2.1 (Herbrand Universe)**.** *Given a Datalog theory, if its signature is $\langle \mathcal{P}, \mathcal{C}, \mathcal{V} \rangle$, then the Herbrand universe of this Datalog theory is $\mathcal{C}$.*

Definition 3.2.2 (Herbrand Base($\mathbb{B}$))**.** *Given a Datalog theory, if its signature is $\langle \mathcal{P}, \mathcal{C}, \mathcal{V} \rangle$, then the Herbrand base $\mathbb{B}$ of this Datalog theory is defined by the following rule.*

$$\mathbb{B} = \{p(c_1, ...c_n) | \forall n,\ p/n \in \mathcal{P},\ \forall c_i \in \mathcal{C},\ 1 \le i \le n\}$$

where p/n denotes that p is a predicate of arity n.

Herbrand structure interprets a term as its semantic value, e.g., constant *dog* as the animal "dog". Accordingly, in a Herbrand structure, ground propositions from the Herbrand base become both syntactic and semantic, and are assigned with the value of either true or false.

Definition 3.2.3 (Herbrand Structure)**.** *Given a Datalog theory $\mathbb{T}$, if its Herbrand base is $\mathbb{B}$, then a Herbrand structure $\mathbb{S}$ of that Datalog theory gives an interpretation in which a subset of $\mathbb{B}$ is true.*

$$\exists B_0 \subset \mathbb{B},\ \forall P \in B_0,\ \mathbb{S} \models P$$

In Definition 3.2.2, the subset B_0 is a set of propositions while $\mathbb{S}$ is an interpretation which interprets all propositions in B_0 by assigning them true.

Definition 3.2.4 (Herbrand Model)**.** *Given a Datalog theory $\mathbb{T}$, $\mathbb{M}$ is a Herbrand model iff it is a Herbrand structure for $\mathbb{T}$, and $\mathbb{M} \models \alpha$, where α is a theorem of $\mathbb{T}$.*

Resembling the idea of Herbrand structure, the benchmark for the correctness of a Datalog theory will be defined as the Preferred Structure ($\mathbb{PS}$) in Definition 4.2.3 on page 50. The Preferred Structure ($\mathbb{PS}$) represents all the ground propositions explicitly given by the user: like a Herbrand structure, the true ground propositions are the true set of $\mathbb{PS}$; but unlike a Herbrand structure, the false ground propositions are a second, false set of $\mathbb{PS}$. Similar to a Herbrand structure, the propositions in the true set of $\mathbb{PS}$ are interpreted to be true. Inversely to a Herbrand structure, the ones in the false set are instead interpreted to be false. If a theory violates either set of $\mathbb{PS}$, it will be regarded as faulty and will then be repaired.

3.3 Linear Resolution with Selection Function

One of the core component of ABC is the automated theorem prover. The inference rule, Linear Resolution with Selection Function (SL-Resolution) is employed, which is not only sound and complete (Gallier, 2003), but also decidable (Pfenning, 2006) for Datalog theories so that proofs can always be detected if there are any.

SL-Resolution (Kowalski and Kuehner, 1971) tries to derive the empty clause from the goal clause and the axioms in the theory. The principles of the SL-Resolution for Datalog implemented in ABC are as the following, where *the resolution between a goal and a clause* is called a Resolution Step (RS).

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

In Datalog, it is either an assertion or a rule to prove. To prove the former, the goal clause is the negation of that assertion. As for the latter, if it is a constraint axiom, which has no head but preconditions, e.g., $p(X) \wedge q(Y) \implies$, then the constraint axiom is directly used as the goal clause. On the other hand, if it is a normal rule with a head, e.g., $p(X) \wedge q(Y) \implies r(X,Y)$, then it has to be rewritten by the following steps first.

1. Negate the rule and push $\neg$ inwards. The rewritten rule is:

$$\exists X,Y.\neg[p(X) \wedge q(Y) \implies r(X,Y)]$$

2. Skolemise the above rule by replacing bound variables with Skolem constants c_1 and c_2 that are new to the signature[2] of the theory (Bundy, 1985).

$$\neg[p(c_1) \wedge q(c_2) \implies r(c_1,c_2)]$$

[2]The signature describes the representation language in which a logical theory is written (Hodges and Scanlon, 2018).

3. Write the above rule in conjunctive normal form. It becomes the three sentences below.

$$\begin{aligned} &\implies p(c_1) \\ &\implies q(c_2) \\ r(c_1, c_2) &\implies \end{aligned}$$

To prove the original rule $p(X) \wedge q(Y) \implies r(X,Y)$ equals to take $r(c_1,c_2) \implies$ as the goal clause and then to derive the empty clause with the other axioms in the theory and the temporary assertions $\implies p(c_1)$ and $\implies q(c_2)$.

Suppose that the goal clause is $p \implies$[3]. Then we resolve the goal with an input clause whose head is p. Since an axiom in Datalog theory is a Horn clause, which has at most one positive literal, the most recently introduced literals of a non-empty resolvent could only be a disjunction of negative literals, called sub-goals. Then, we continually resolve the first sub-goal, with an input clause. Again, the result is either an empty clause or a clause of a disjunction of negative literals. This process is repeated until no sub-goal is left, or no input clause is available to resolve the first sub-goal. If it ends with the former (an empty clause), refutation occurs, which means that the input theory proves the goal clause.

In our SL-Resolution, *ancestor resolution*, where a resulted clause from a previous RS will be used to resolve the selected literal (Kowalski and Kuehner, 1971), cannot arise because we are dealing with a set of Horn clauses, and our resolution starts with resolving the goal clause. As a result, sub-goals in the current goal clause and all its ancestors are negative literals, therefore they cannot resolve with each other. On the other hand, occurs check is not needed because there is no function in a Datalog theory.

3.4 Repair Techniques

In this section, the existing repair techniques that are involved in our project will be introduced. These techniques are evolved and combined in our repair framework. By discussing their existing version, the motivation of our work will be revealed and it will help the understanding of our original work in the following chapters.

[3]In Kowalski Form, $\neg p$ is written as $p \implies$, which is more natural in resolution proofs (Bundy, 1985). Notice that $\neg p$ only exists in proofs, rather than being an axiom in the logical theory. Writing in Kowalski Form $p \implies$ helps avoid the confusion caused by the occurrence of negation in $\neg p$, considering that the negated assertion is not allowed in a Datalog theory.

3.4.1 Belief Revision

When a new belief needs to be added, belief revision deletes some old ones to keep the belief set consistent. The main question that belief revision algorithms seek to answer is which old ones should be deleted.

Postulates $(K\dot{+}1)$ - $(K\dot{+}6)$ below show the most widely used framework of belief revision: the AGM postulates, which is named after their proponents, Alchourrón, Gärdenfors, and Makinson (Gärdenfors, 2003). Here $\dot{+}$ and $+$ are the operations of revision and expansion respectively. Both of $\dot{+}$ and $+$ add a new belief to a belief system and output a revised belief system. Note that $\dot{+}$ may delete old ones while $+$ does not. K is a consistent belief set which is logically closed; ϕ and ψ are two beliefs, and $K_\perp$ is an inconsistent belief set, which is the set of all formulae.

$(K\dot{+}1)$ For any sentence ϕ and any belief set K, $K\dot{+}\phi$ is a belief set.

$(K\dot{+}2)$ $\phi \in K\dot{+}\phi$.

$(K\dot{+}3)$ $K\dot{+}\phi \subseteq K+\phi$.

$(K\dot{+}4)$ If $\neg\phi \notin K$, then $K+\phi \subseteq K\dot{+}\phi$.

$(K\dot{+}5)$ $K\dot{+}\phi = K_\perp$, if and only if $\neg\phi$ is true.

$(K\dot{+}6)$ If $\phi \leftrightarrow \psi$, then $K\dot{+}\phi = K\dot{+}\psi$.

AGM postulates are the constraints that implementations of belief revision should follow. The first three postulates are self-explanatory. Combining $(K\dot{+}3)$ with $(K\dot{+}4)$, shows when the input ϕ does not contradict what is already in K, that is $\neg\phi \notin K$, then revision is identified with expansion. $(K\dot{+}5)$ means the result of a revision should be a consistent belief set unless the input ϕ is logically impossible. $(K\dot{+}6)$ explains that the revision works on the knowledge level rather than syntactic level, which means two beliefs will be considered as equivalent and will produce the same revised belief set if they have the same content (Gärdenfors, 2003).

Based on the criterion of the informational economy: "*information is in general not gratuitous, and unnecessary losses of information are therefore to be avoided, the underlying motivation for AGM postulates is to retain old beliefs as much as possible*" (Gärdenfors, 2003, p9). From the perspective of the relation between revision and contraction, (Gärdenfors, 1992) argues that the process of revision can be reduced to that of contraction via the so-called *Levi identity*, shown in Equation 3.1, where $\dot{-}$ is a contraction operation, which deletes a belief from a belief set.

$$K \dot{+} \phi = (K \dot{-} \neg\phi) + \phi \tag{3.1}$$

The explicit models of the contraction function include the following three kinds (Alchourrón and Makinson, 1985).

- Maxi-choice contraction.
- Partial meet contraction.
- Full meet contraction.

The basic idea of *maxi-choice contraction* is to select one maximal consistent subset. If ϕ is the belief to be contracted, then the definition of a maximal subset is that a belief set K_1 is the maximal subset of belief set K that fails to imply ϕ, if and only if:

1. $K_1 \subseteq K$,
2. $\phi \notin Cn(K_1)$,
3. for any K_2 such that $K_1 \subset K_2 \subseteq K$, $\psi \in Cn(K_2)$.

Here $Cn(K)$ represents the set of all logical consequences of K. *Full meet contraction* contains only the beliefs that are common to all of the maximal subsets. The maxi-choice contraction function generates max belief sets and the full meet contraction function results in contracted belief sets that are too small sometimes. As the result of the full meet contraction being too small, *partial meet contraction* was developed, which chooses some of the maximal subsets. In ABC, maxi-choice contraction is chosen to conduct the minimal change by keeping as many axioms as possible. Because all axioms are potentially valuable, we do not want to delete more than necessary for repairing a fault.

Example 3.4.1. *Swan Theory.*

$$german(X) \implies european(X) \quad \text{(A1)}$$
$$european(X) \wedge swan(X) \implies white(X) \quad \text{(A2)}$$
$$\implies german(bruce) \quad \text{(A3)}$$
$$\implies swan(bruce) \quad \text{(A4)}$$

The following theorem is derivable from S1 - S4:

$$\implies white(bruce) \quad \text{(A5)}$$

The observation is:

$$\implies black(bruce) \quad \text{(A6)}$$

In Example 3.4.1, the derived theorem A5 inconsistent with the observation A6. To embed the observation, belief revision calculated based on maxi-choice contraction will result in a theory constituted of $\{A1,A2,A3,A6\}$ or $\{A1,A3,A4,A6\}$ or $\{A2,A3,A4,A6\}$. Revision from full meet contraction results in $\{A3,A6\}$ and revision from partial meet contraction outputs $\{A3,A6,A1\}$ or $\{A3,A6,A2\}$ or $\{A3,A6,A4\}$.

The algorithms which implement belief revision have been developed by (Alchourrón and Makinson, 1985; Jin and Thielscher, 2008; Nebel, 1991) among others.

3.4.2 Epistemic Entrenchment

In this section, the idea of epistemic entrenchment (EE) is introduced, which is a component of ABC.

An unaided computer system cannot understand the semantics of theories, so it cannot directly judge which axioms have the most overall informational value. For example, an axiom from an academic book is more entrenched than a speculative one from someone's blog.

In belief revision, epistemic entrenchment is proposed for prioritising axioms (Gärdenfors, 1988). The more entrenched an axiom is, the more valuable it is, and the system will be less inclined to change it.

A set of postulates are given to describe the qualitative properties of EE by (Gärdenfors and Makinson, 1988). We will use $\mathcal{E}$ to represent the function which returns the EE value of a belief with higher values for more entrenchment; A, B, C for distinct beliefs and $K_{\perp}$ for an inconsistent belief set. The the range of EE is totally ordered by the relation represented in EE1. The name of each postulate is given in the bracket on right.

EE1. If $\mathcal{E}(A) \leq \mathcal{E}(B)$ and $\mathcal{E}(B) \leq \mathcal{E}(C)$, then $\mathcal{E}(A) \leq \mathcal{E}(C)$. (Transitivity)

EE2. If $A \implies B$, then $\mathcal{E}(A) \leq \mathcal{E}(B)$. (Dominance)

EE3. $\forall A,\ B, \mathcal{E}(A) \leq \mathcal{E}(A \wedge B)$ or $\mathcal{E}(B) \leq \mathcal{E}(A \wedge B)$. (Conjunctiveness)

EE4. $\forall B \in K, K \neq K_{\perp},\ A \notin K$, iff $\mathcal{E}(A) \leq \mathcal{E}(B)$. (Minimality)

EE5. If $\forall B,\ \mathcal{E}(B) \leq \mathcal{E}(A)$, then $\implies A$. (Maximality)

EE1 constrains a measurement to be transitive. EE2 decides that when either A or B has to be retracted during revision process, then A should be retracted. Otherwise, B would still be derived from A. EE3 claims that the retraction of $A \wedge B$ can be done by either retracting A or B. EE2 is also applicable because $A \wedge B \implies A$ and $A \wedge B \implies B$. Thus, combining EE2 and EE3, we have the following conclusion.

$$\mathcal{E}(A) < \mathcal{E}(B) \implies \mathcal{E}(A \wedge B) = \mathcal{E}(A)$$

EE4 describes that if A is not a part of a consistent K, then it is less entrenched than a belief in K. EE5 gives the maximum entrenchment to a tautology belief A, which follows EE2 as well because for all B, B implies A and then A is the most entrenched belief.

Even though the properties above have been proposed, it is still hard to quantitatively define a measurement for EE that works for all domains. Because the factors that effect the value of EE are diverse and their interactions are complicated. Thus, (Gärdenfors and Makinson, 1988) does not assume that one can quantitatively measure degrees of EE (Gärdenfors, 1988).

For evaluating EE, it is natural to think about the factors which influence an axiom's place in its domain. For example, these factors could include the source where an axiom comes from, the time when an axiom was established, and the impact of an axiom's semantic content in that domain, and how often it will change, e.g., compare the position of a *chair* to the position of a *house*.

An academic publication is a good example of how the source of an axiom could affect its EE. If a theory has been published and cited by lots of academic papers, then it is more entrenched than one that has not.

Even if all factors are successfully formalised, their impact on the entrenchment can be opposite in different domains. Two examples are given for further explanation. The following discussion is based on the unique name assumption so all constants are unequal to all others.

Example 3.4.2. *A Theory of Residence Postcode.*

$$postcode(X,\ Z) \wedge postcode(X,Y) \implies Z = Y \qquad (A1)$$

$$\implies postcode(david, eh11ls) \qquad (A2)$$

$$\implies postcode(david, eh93dh) \qquad (A3)$$

In the above theory, each citizen should have only one residence postcode. But David has two postcodes. In this scenario, it would be more likely that David has moved to a new place, and then the newer recorded axiom should be the more entrenched one than the older axiom. Here the background knowledge is that people can move to another place. Or David might have a second home, which violates A1 and then the background knowledge is that one person can have multiple addresses.

Different from the impact that time has on the addresses, in science, a classic theory e.g., Newton's physics, is more entrenched than a new discovery. An old theory is long-tested and then be considered to be trustworthy, while a new discovery does not. For example, the suggestion that neutrinos travelled 0.002% faster than light was made in 2011, and later disproved in 2012. The mistake was caused by a loose fibre optic cable which introduced a delay in the timing system (Cartlidge, 2012). Even before the mistake was discovered, the result had been considered anomalous as it challenges a cornerstone in physics. A new challenging theory is accepted if and only if there is strong evidence, e.g., reliably replicated experiments.

EE for one sentence can be inverted in come cases. For example, the suspect of the murder of a wife is her husband, because the evidence pointed to him. But along with the investigation, more and more details and evidence come to light and they reveal the innocence of the husband. Then the EE of his blame for the crime is inverted.

In summary, it is difficult to find all factors that influence the entrenchment of a belief, especially domain-dependent ones, where a whole picture of background knowledge is necessary. Besides, it is not clear how to quantitatively evaluate the impact that these factors have on EE. Last but not the least, the same factor could have different dynamic impacts in different scenarios.

Due to these difficulties, the measurement of epistemic entrenchment was an open question in the belief revision literature. In our project, EE is measured as the *preference entrenchment* based on $\mathbb{PS}$, which is introduced in §6.2. $\mathbb{PS}$ can be seen as partial background knowledge that provides the truth value of some sentences, so the entrenchment of an axiom can be measured according to how much it respects $\mathbb{PS}$.

3.4.3 Conceptual Change via Reformation

Reformation is a domain-independent repair algorithm for conceptual change. Based on changes at the syntactic level, reformation repairs a fault in a logical theory by blocking unwanted but successful proofs or unblocking a partial but wanted proof (Bundy and Mitrovic, 2016). A proof can be blocked by breaking one of the unifications in its resolution steps and a partial proof can be turned into a proof by repairing its necessary unifications. We call both predicate and function *functor*.

Case	**Before**	**Condition**	**Block**	**Unblock**
Base	$\top$		Failure	Success
CC_s	$F(\vec{s}^m) \equiv G(\vec{t}^n)$ $\wedge u$	$F = G$ $\wedge\, m = n$	Make $F(\vec{s}^m) \neq F(\vec{t}^m)$ $\bigvee_{i=1}^n$ Block $s_i \equiv t_i$ $\vee$ Block u	 $\bigwedge_{i=1}^n$ Unblock $s_i \equiv t_i$ $\wedge$ Unblock u
CC_f		$F \neq G$ $\vee\, m \neq n$	Success	Make $F(\vec{s}^m) = G(\vec{t}^n)$ $\bigwedge_{i=1}^n$ Unblock $\nu(s_i) \equiv \nu(t_i)$ $\wedge$ Unblock $\nu(u)$
VC_s	$x \equiv t \wedge u$	$x \notin \mathcal{V}(t)$	Make $x \in \mathcal{V}(t)$ $\vee$ Block $u\{x/t\}$	 Unblock $u\{x/t\}$
VC_f	or $t \equiv x \wedge u$	$x \in \mathcal{V}(t)$	Success	Make $x \notin \mathcal{V}(t)$ $\wedge$ Unblock $\nu(u\{x/t\})$

Table 3.1: The Reformation Algorithm for First-Order Logic, where repair plans inverts the outcome of the targeted proof step: *F and G are functors; t is a non-variable term and s is any term; s_i and t_j are terms; $m,n > 0$; x and y are distinct variables; and u is a unification problem; $s \equiv t$ is the problem of unifying s and t; and $\mathcal{V}(t)$ is a set of free variables of term t. CC names a rule that is applicable when the inputs are both compound terms or constants, and VC when just one of the inputs is a variable. Their s and f subscripts indicate rules resulting in success and failure, respectively (Bundy and Mitrovic, 2016).*

Figure 3.1 describes the reformation algorithm which is adapted from unification cases based on a non-standard version of the unsorted FOL unification algorithm in clausal form. Each case in Figure 3.1 has a pair of complementary conditions leading to success or failure respectively. For example, a unification is successful (CC_s) iff it satisfies CC_s's condition $F = G \wedge m = n$, while the condition $F \neq G \vee m \neq n$ refers to a failed unification (CC_f).

By changing the signature, the result of a unification can be inverted from success to failure and vice versa. For example, if a CC_s unification is in an unwanted proof, it can be broken by the repair which renames either F or G to make them unequal or changes the arity of the functor in either side to make their arities different, provided functors can be overloaded with different arities, as in Prolog.

In general, the typical signature changes of reformation include the splitting of a functor into two functors, the merging of two distinct functors into one, and the change of the arity of a functor or the insertion of a variable to fail the occurs check (Bundy, 1985). Because arity changes in a description logic can have side effects, e.g., turn a concept name to a role name, reformation's arity change does not directly work well for description logics. On the other hand, non-monotonic logics are not necessary since reformation can add a new argument to distinguish sub-classes and exceptions. Although reformation is based on full FOL, our ABC Repair System is based on Datalog which is a decidable subset of FOL.

Reformation is self-inverse so that no information is lost during the repairing processes, which shows that reformation repairs are in some sense minimal (Bundy and Mitrovic, 2016). So far, various heuristics have been suggested and implemented to control the search space of reformation. Additionally, reformation has been adapted to several logics, such as unsorted first-order logic which is mentioned above (Bundy and Mitrovic, 2016), many-sorted first-order logic (Mitrovic, 2013), and a common description logic (Tsialos, 2015).

3.4.4 Abduction

Deduction, induction, and abduction (Hintikka, 1999) are three forms of inference. In our project, deduction is the inference form of the framework of repair processes and abduction is one of our repair techniques, which add the axioms of the cause of a previously unprovable observation to the input Datalog theory.

A set of the desired properties of a cause w.r.t. an observation and the process of seeking for such a cause have been developed by (Cox and Pietrzykowski, 1986), which is employed as a repair technique in our project.

We use K to represent a knowledge base written in well-formed formulae and D for a domain which is a subset of K. A cause c of the observation o with K satisfies $c \wedge K \implies o$. Here c and c' represent distinct causes so $c \not\equiv c'$. The properties below are based on a knowledge base K, which will not be mentioned repeatedly.

P1. c is *minimal* iff $\forall c'$, if $c \implies c'$ then $c \equiv c'$.

P2. c is *acceptable* iff there is a subset S of K and $c \wedge S \wedge D$ is satisfiable.

P3. c is *nontrivial* iff $c \not\Longrightarrow o$.

P4. c is *basic* iff all the acceptable causes of c are trivial.

P5. c is *fundamental* iff it has all of the above properties.

P1 eliminates the causes which are unnecessary in deducing o. P2 corresponds to consistency, e.g., when $D = K$, the condition that $c \wedge S \wedge D$ is satisfiable, e.g., c is consistent with K. Example 3.4.3 below is adapted from (Cox and Pietrzykowski, 1986) to illustrate why they propose P2 as a desired property, where the last axiom is a constraint axiom.

Example 3.4.3. *Inconsistent Bird Theory.*

$$\begin{aligned} bird(X) &\implies fly(X) && \text{(A1)}\\ bird(X) \wedge unusual(X) &\implies nonfly(X) && \text{(A2)}\\ penguin(X) &\implies bird(X) && \text{(A3)}\\ penguin(X) &\implies unusual(X) && \text{(A4)}\\ fly(X) \wedge nonfly(X) &\implies && \text{(A5)} \end{aligned}$$

Here K is constituted by five axioms $(A1 - A5)$ by directly translating the example sentences in (Cox and Pietrzykowski, 1986). Assuming the observation that $nonfly(lily)$, and S is (A2, A3, A4, A5) and then $penguin(lily)$ is the cause. The reason they abstract S from K is that if $A1$ is considered, then it is inconsistent so that $penguin(lily)$ would not be the cause.

It can be seen that their discussions are based on the accepting of an inconsistent theory. But generally, a theory needs to be consistent before performing its duty of providing information or knowledge. In our project, inconsistency will be repaired first. For example, reformation can increase the arity of predicate $bird/1$ and then only the normal bird can fly, as formalised in Example 3.4.4. Alternatively, the predicate symbol 'bird' can be split into 'normalBird' and 'abnormalBird'.

Example 3.4.4. *Repaired Bird Theory.*

$$bird(X,normal) \implies fly(X) \quad \text{(A1)}$$

$$bird(X,abnormal) \implies nonfly(X) \quad \text{(A2)}$$

$$penguin(X) \implies bird(X,abnormal) \quad \text{(A3)}$$

$$fly(X) \wedge nonfly(X) \implies \quad \text{(A5)}$$

Now given the observation $nonfly(lily)$, the cause $penguin(lily)$ can be added by abduction as the cause. Therefore, we only discuss consistent theories, not only because it is a basic property at a useful theory, but also because we are capable at repairing an inconsistent one. In this case, we can revise P2 into P2' where c is acceptable iff it is consistent with K.

P3 prunes the cause which is irrelevant to the knowledge based but directly implies the observation, e.g., the observation itself. From the view of repairing a faulty theory, expansion of the observations to the input theory is desired in some scenarios and this kind of repair is incorporated in belief revision, but not abduction. P4 eliminates intermediate causes. In the end, only the causes which have all of these four properties are fundamental.

These properties provide a good direction for evaluating whether an explanation is of good quality. The task of abduction is to find the fundamental cause for an observation.

The following five-step process to compute the fundamental causes of an observation are also given by (Cox and Pietrzykowski, 1986). In Step1, the resolution process may not terminate in First-order logic (FOL) but it does in Datalog.

Step 1. **Resolution by refutation.** Negate o and resolve $\neg o$ with axioms in K. Continue the resolution until the remaining sub-goals are irresolvable, which are called *a dead end* of o. The set of dead ends w.r.t. one observation is written as G.

Step 2. **Retain basic causes G_b.** For each dead end b in G, negate all literals in it and obtain a conjunction of positive literals $P_1 \wedge ... \wedge P_n$. This conjunction is a basic cause of o, written as c_b. The set of all c_b is G_b.

Step 3. **Retain acceptable causes G_{ba}.** From the basic set, get the acceptable causes.

$G_{ba} = \{c_b \mid c_b \in G_b, c_b \cup K \not\Longrightarrow\}$. For a Datalog theory, if there are no axioms in the form of A5 in Example 3.4.3, it is always consistent so $G_b = G_{ba}$.

Step 4. **Retain nontrivial causes G_{ban}.** The set of basic, acceptable and nontrivial causes is; $G_{ban} = \{c_{ba} \mid c_{ba} \in G_{ba},\ c_{ba} \not\Longrightarrow o\}$.

Step 5. **Retain minimal causes G_{banm}.** The set of fundamental causes G_{banm} is; $G_{banm} = \{c_{ban} \mid \forall c_{ban1} \in G_{ban},\ c_{ban1} \neq c_{ban},\ c_{ban} \not\Longrightarrow c_{ban1}\}$.

The causes found by the above process are only assertions without rules. Based on the categories of abduction defined in (Schurz, 2008), the above abduction is a *factual abduction*, where both the observation and the cause are singular facts. Apart from this factual abduction, we also developed an *analogical abduction* algorithm particularly for finding a rule as the cause of an observation by splitting an existing rule in the input theory, which is introduced in §5.2.

3.5 Summary

In this chapter, we briefly introduced the related work including Datalog, Herbrand Structure, belief revision together with epistemic entrenchment, conceptual change via reformation, abduction. Datalog is the logic of the input theory of our repair processes. Although it has less expressive power than FOL, Datalog still has wide applications. Meanwhile, it makes inference decidable and allows a simplified reformation algorithm. To evaluate whether a Datalog theory is correct w.r.t. the observations, a benchmark is needed which should be able to represent both the true assertions and the false propositions. To formalise the desired benchmark, Herbrand structure is employed, because its ground propositions are both syntactically and semantically with true values.

Given a benchmark, a fault in a Datalog theory can be detected by the implemented SL-Resolution, and then fixed by the repair techniques including abduction, belief revision and reformation, which have been introduced in this chapter. In our project, a framework is developed which not only combines these techniques but also introduces new repair plans, e.g., a variant of abduction which deletes unprovable preconditions from a rule.

These three techniques are applied mostly independently based on one targeted proof of a fault, because their postulates don't overlap with each other, e.g., the

AGM postulates do not apply to reformation, and reformation's algorithm do not apply to the other two either (Bundy, 2015). As a theory is faulty usually due to more than one incorrect proof steps, and each of them could be addressed by different repair techniques, the final result will usually be the combination of repair operations generated from different techniques.

The original work of fault definition, detection and repair generation are introduced in Chapter 4 and 5. In addition, some refinements of the repair process in ABC are given in Chapter 6. Their implementations are introduced in Chapter 7.

Chapter 4

ABC Repair System: Before Repair Generation

The components of ABC Repair System is given by Figure 4.1. In this chapter[1], the processes before C3 are presented, which are C1 in §4.1 and C2 in §4.2-4.3. In addition, §4.4 gives the theoretical basis of the repair generation of C3 by answering the essential questions of what the postulates and search strategy of the repair generation are and analysing the repair techniques of abduction, belief revision and conceptual change based on the defined faults individually. Then the repair algorithm of C3 is introduced in Chapter 5, the definition of maximal sets of commutative repair plans (MSCR) and the sub-optimal pruning of C4 in Chapter 6. The properties of the algorithm of repair generation are important but especially difficult to analyse, which can be a good extended research topic. Although we do not pursue these analysis in this thesis, we outline it as point 1 in the future work section §9.1 on page 183.

The inputs of ABC are a Datalog theory ($\mathbb{T}$) and a Preferred Structure ($\mathbb{PS}$) which is defined in §4.2. The output is a set of repaired fault-free theories. Note that $\mathbb{PS}$ is like a benchmark in our repair mechanism, which is not allowed to be changed in ABC. Because each original repair technique could generate multiple alternative repairs for each fault, it is also common that the repair algorithm generates more than one repair solution. Thus, ABC outputs a set of the repaired theories, rather than a single one. It is possible that no repair solution can be found, then the output is an empty set and the repair process of the current branch ends in failure.

C1 is a pre-process which checks whether the $\mathbb{PS}$ is self-contradictory and

[1]Recall that all predicate symbols and constants start with lowercase letters, and variables start with capital letters.

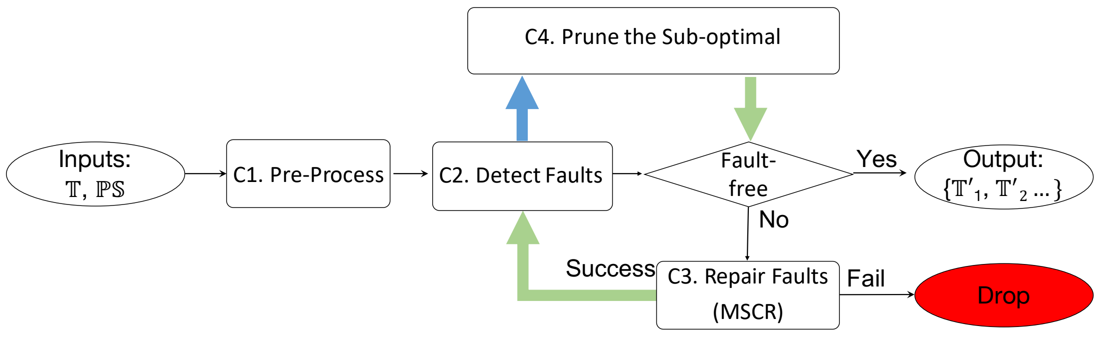

C4. Prune the Sub-optimal
Inputs: $\mathbb{T}$, $\mathbb{PS}$
C1. Pre-Process
C2. Detect Faults
Fault-free
Yes
No
Output: $\{\mathbb{T}'_1, \mathbb{T}'_2 ...\}$
Success
C3. Repair Faults (MSCR)
Fail
Drop

Figure 4.1: Flowchart of the ABC Repair System: green arrows deliver a set of theories one by one to the next process; the blue arrow collects and delivers theories as a set; When some faulty-theories are not repairable, they will be dropped from the repair process.

calculates a minimal set of axioms by pruning redundancy, as discussed in §4.1. It reduces the search space of both fault detection and repair generation. Given a minimal set, faults are detected by C2. If there is no fault, then the theory is collected as a output. Otherwise, the information about the fault, which could be either proofs or failed proofs, is provided to C3 to generate repairs by combining abduction, belief revision and reformation. If no repairs can be found or the resource threshold, defined in Definition 4.0.1, is reached, the process will be terminated with the failure of finding any repaired theory. The resource threshold may cause the problem of not being able to find a solution even if there is one, especially when the threshold is improperly small, but it helps to return the simplest repairs which contain the least repair operations, and avoid non-termination of a long search branch. This threshold can be set to be infinite, so no threshold is enforced.

Definition 4.0.1. *The resource threshold includes the depth limit of search branches and the maximum number of repair operations given by the user.*

Otherwise, C3 generates all possible repair plans in parallel. There could be more than ten repairs for one fault depending on the components of that fault's proofs: the more axioms or signature items are involved in its proofs, the more repair plans are available. All of these repair plans are combined into MSCRs. Then C3 outputs a candidate set of semi-repaired theories after applying the MSCRs respectively. Then C2 will check the remaining faults of these candidates one by one. This search process is introduced in §4.4.2.

By comparing the number of the remaining faults of all candidates, sub-optimal ones will be pruned by C4. Entrenchment is measured to order repairs. This is discussed in §6.2 . If an optimal repaired theory is still faulty, it will be further repaired by C3. Otherwise, it will be collected as one of the outputted fault-free theory. The process among C2, C3 and C4 repeats until no repairable faulty theories are left in the process.

4.1 Minimal Sets Computation

In a logical theory, any axiom which is derivable from the others can be seen as a redundancy, which increases the search space of reasoning when detecting faults. Worse still, if we repair a redundant axiom, the fault caused by it will still be derivable from the others. Then further repairs to axioms have to be given. Therefore, pruning redundant axioms first contributes a smaller search space for both detecting faults and generating repairs. This section introduces how we compute a finitely minimal set of axioms from the theory.

Based on SL-Resolution implemented by ourselves, the general process of computing the minimal sets is to check the provability of each axiom by trying to prove it from the others. If an axiom is provable, it will be deleted, resulting in a smaller set. Repeat checking the provability until all of the sentences in a set are unprovable from the others, then this set is a minimal set. It needs to be noticed that the minimal sets of a theory can be multiple, so that an axiom in one minimal set could be a redundancy in another.

Given an input theory, smaller sets are explored step by step for searching for minimal sets. Depth-first is chosen as our search algorithm, just as in (Russell and Norvig, 1995), which does not need to record all the search branches as breadth-first does.

There can be multiple minimal sets for a given theory. Before discussing about the selection among these minimal sets, their differences should be analysed first. Although they contain different axioms and their size can be different as well, these minimal sets are equal up to variable renaming in terms of their theorems and signatures. Because they are all computed from one original theory.

Therefore, rather than interactively select an arbitrary one of the minimal sets as in (Plotkin, 1971), the first one being and found by our depth-first search is automatically chosen as the output of the computation of the minimal set in this thesis. This choice keeps the whole process in ABC being automatic.

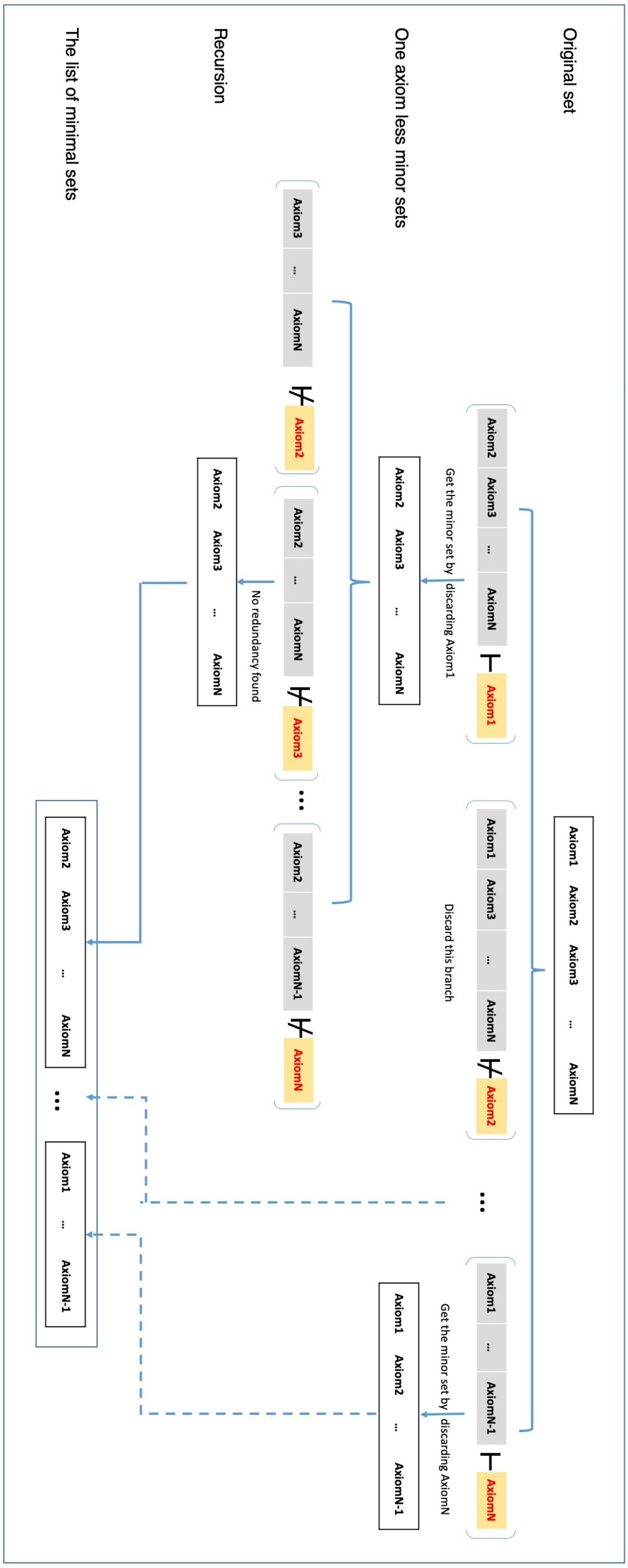

Original set
Axiom1 Axiom2 Axiom3 … AxiomN
One axiom less minor sets
Axiom2 Axiom3 … AxiomN ⊢ Axiom1
Get the minor set by discarding Axiom1
Axiom1 Axiom3 … AxiomN ⊬ Axiom2
Discard this branch
Axiom1 … AxiomN-1 ⊢ AxiomN
Get the minor set by discarding AxiomN
Axiom2 Axiom3 … AxiomN
Axiom1 Axiom2 … AxiomN-1
Recursion
Axiom3 … AxiomN ⊬ Axiom2
Axiom2 … AxiomN ⊬ Axiom3
Axiom2 … AxiomN-1 ⊬ AxiomN
No redundancy found
Axiom2 Axiom3 … AxiomN
The list of minimal sets
Axiom2 Axiom3 … AxiomN
Axiom1 … AxiomN-1

Figure 4.2: The computation of the minimal sets.

It can be a future work to investigate whether the repairs are significantly different depending on the minimal sets systematically, discussed as point 9 in §9.1.

Given one minimal set, the detection and repair of its faults are discussed in the following sections.

4.2 Fault Definition Based on a Preferred Structure

Common faults of a logical theory are inconsistency and incompleteness. Without constraint axioms[2], a Datalog theory is guaranteed to be consistent by being unable to prove a negative statement. When there are constraint axioms, a Datalog theory can be inconsistent due to the *constraint violation*, as defined in Definition 4.2.1.

Definition 4.2.1 (Constraint Violation)**.** *The inconsistency caused by a constraint axiom is a constraint violation, which occurs when all the preconditions of that constraint axiom are provable as theorems of the Datalog theory.*

As for completeness, it may be too strong a requirement.

Definition 4.2.2. *A complete theory is one in which every sentence is either provable or its negation is.*

However, we do not have a complete theory in most of applications, e.g., knowledge graphs. Some existing theories are inherently incomplete, e.g., Peano Arithmetic, as shown in Godel's Incompleteness Theorem (Smullyan, 1992). Meanwhile, it is unrealistic and unnecessary in some sense for a theory to be complete.

Instead of incompleteness, we are going to focus on those propositions which should or should not be the theorems of an object theory according to the users' requirements. For example, a scientist verifies a physics theory which gives the information about a bottle by experimental observations. Then the observations are the ground truth of the theory. Suppose that the observation tells that the bottle is white and is located in a corner, then *colour(bottle, white)* and *location(bottle, corner)* should be theorems of the theory. Meanwhile, if the observation is that the bottle is not empty, then *empty(bottle)* should not be a theorem of the theory. As for statements which make no sense to the theory, e.g., *location(bottle, bottle)*, or the ones which makes sense but describes not important things to the current experiment, e.g., *clean(bottle)*,

[2]Constraint axioms are defined in Definition 3.1.1 on page 28.

they are not interesting to the scientist and would not be of concern while verifying the theory.

It can be seen that the guidance information for adjusting a logical theory could be considered as the users' requirements which tell which propositions should or should not be the theorems of that theory, while ignoring the meaningless propositions. Resembling the idea of Herbrand Structures (Wikipedia contributors, 2017), those different sets of ground propositions are designed to constitute the following defined *Preferred Structure*. In this context, *structure* refers to the relation constructed by different sets of propositions based on their truth values according to the user.

Definition 4.2.3 (Preferred Structure)**.** *A Preferred Structure ($\mathbb{PS}$) is a pair of structures constructed over the signature of a logical theory ($\mathbb{T}$):*

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

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

A $\mathbb{PS}$ describes the user's requirements that a *faithful* $\mathbb{T}$ should follow. As shown in Figure 4.1, $\mathbb{PS}$ is a required input in addition to the logical theory. It is similar to the positive and negative examples in the field of ML. Although a $\mathbb{PS}$ is unchangeable in this thesis, it will be able to be modified for some extensions of ABC in the future, discussed in point 6 in §9.1. Giving it a specific definition with a distinguishing name contributes to a clear definition of the faults to repair in the following context. Also, the intention of modifying a $\mathbb{PS}$ in future work is different from the operation of the positive and negative examples in ML. Thus, another name helps to distinguish them from each other.

Note that a $\mathbb{PS}$ could be partial. Some propositions could occur neither in the true set, nor in the false set. Because the user's knowledge is limited, or the user only focuses on some particular assertions which represent the content that the user cares about.

On the other hand, it is vital that $\mathcal{T}(\mathbb{PS})$ and $\mathcal{F}(\mathbb{PS})$ do not overlap. Otherwise, $\mathbb{PS}$ is *ineffective* by being self-contradictory.

Theorem 4.2.1. *No theory can respect* $\mathbb{PS}$ *if* $\mathcal{T}(\mathbb{PS}) \cap \mathcal{F}(\mathbb{PS}) \neq \emptyset$.

Proof. Assume that $\alpha \in \mathcal{T}(\mathbb{PS}) \cap \mathcal{F}(\mathbb{PS})$ and $\mathbb{T}$ respect that $\mathbb{PS}$.

Then $\mathbb{T}$ should be sufficient to prove all propositions in $\mathcal{T}(\mathbb{PS})$ including α. Thus we have:

$$\mathbb{T} \vdash \alpha \tag{4.1}$$

Meanwhile, $\mathbb{T}$ should be compatible with $\mathbb{PS}$ by not proving any proposition in $\mathcal{F}(\mathbb{PS})$ including α. Thus we have:

$$\mathbb{T} \nvdash \alpha \tag{4.2}$$

Because Equation 4.1 and 4.2 conflict with each other, the assumption of $\mathbb{T}$ respecting that $\mathbb{PS}$ is false. □

If there is a proposition in both $\mathcal{T}(\mathbb{PS})$ and $\mathcal{F}(\mathbb{PS})$, our system's process stops with a warning. In this case, the user has to correct the self-contradictory $\mathbb{PS}$ before giving it as the input of the ABC system.

Example 4.2.1. *Swan Theory.*

$$\begin{aligned} german(X) &\Longrightarrow european(X) && \text{(A1)} \\ european(X) \wedge swan(X) &\Longrightarrow white(X) && \text{(A2)} \\ &\Longrightarrow german(bruce) && \text{(A3)} \\ &\Longrightarrow swan(bruce) && \text{(A4)} \end{aligned}$$

In Example 4.2.1, there are four axioms saying that Germany is part of Europe (A1), all European swans are white (A2), and the swan named Bruce (A4) is a German swan (A3). Then one theorem is that Bruce is white. Imagine the user observes the fact that Bruce is black. In this scenario, the $\mathbb{PS}$ for formalising the user's knowledge that Bruce is black rather than white is:

$$\mathcal{T}(\mathbb{PS}) = \{black(bruce)\}$$
$$\mathcal{F}(\mathbb{PS}) = \{white(bruce)\}$$

An object theory should follow its $\mathbb{PS}$ by proving the preferred propositions, while not proving the violative propositions, which can be written as logical notations 4.3 and 4.4. This makes the theory correct and useful to the user.

$$\forall \alpha,\ \alpha \in \mathcal{T}(\mathbb{PS}),\ \mathbb{T} \vdash \alpha \tag{4.3}$$

$$\forall \beta,\ \beta \in \mathcal{F}(\mathbb{PS}),\ \mathbb{T} \nvdash \beta \tag{4.4}$$

Accordingly, the possible faults of a logical theory are *incompatibility* and *insufficiency* w.r.t. an effective $\mathbb{PS}$.

Definition 4.2.4 (Incompatible and Insufficient)**.**

- *An* incompatible theory $\mathbb{T}$ *w.r.t.* $\mathbb{PS}$ *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 $\mathbb{T}$ *w.r.t.* $\mathbb{PS}$ *is one in which a proposition 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 $\mathbb{T}$ and $\mathbb{PS}$. In the swan example, the original theory proves that Bruce is white, which is an incompatibility, and does not prove that Bruce is black, which is an insufficiency. It is clear that $white(bruce)$ and $black(bruce)$ are the sources of the incompatibility and insufficiency respectively.

One application of our framework is in physical experiments. The observations could be represented in a $\mathbb{PS}$, and its corresponding logical theory should follow that $\mathbb{PS}$. If there is any conflict between a theory and the experimental results, then the conflict corresponds to incompatibility and insufficiency.

Although formalising a $\mathbb{PS}$ by observations is a very important application, it is not the limit of our framework. For example, $\mathbb{PS}$ can also formalise the user's requirements for guiding plan generation, or represent the secure operating specifications for an automated operating system. As the benchmark for a fault-free logical theory, $\mathbb{PS}$ is not allowed to change in this thesis.

Both the incompatibility and the constraint violations are caused by the existence of unwanted proofs. Thus, constraint violation will be seen as a special case of incompatibility which gives a compact discussion in terms of repair generation. In conclusion, the faults of an object theory to be repaired in this thesis are insufficiency and incompatibility w.r.t. a $\mathbb{PS}$, which represents the user's knowledge and opinion. Note that $\mathbb{PS}$ is not allowed to be changed in ABC.

4.3 Fault Detection

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

To detect constraint violation, each constraint axiom is taken as the goal clause directly. If the empty clause can be derived, then a proof of constraint violation is found.

To detect insufficiency or incompatibility, we need to check the provability of a target proposition from a Datalog theory.

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. Each violative proposition, in turn, is checked in the same way. Incompatibility is detected if the input theory proves a violative proposition. In this thesis, $\perp$ represents false.

$$\begin{aligned}&\text{A theory } \mathbb{T} \text{ is incompatible w.r.t. a violative proposition } \beta,\\ &\qquad\text{iff } \mathbb{T}' \vdash \perp, \text{ where } \mathbb{T}' = \mathbb{T} \cup \{\neg\beta\},\ \beta \in \mathcal{F}(\mathbb{PS}).\end{aligned} \tag{4.5}$$

In this thesis, we use the left-hand side term of each resolution to represent a goal/sub-goal, while the right is the input clause side. For example, Figure 4.3 shows how the Swan theory proves the violative proposition of the $\mathbb{PS}$, $white(bruce)$. There are four RSs in total with each highlighted in different colours. In RS1, the initial goal is the negation of the violative proposition, $white(bruce) \Longrightarrow$, which is resolved by the head of axiom A2, *white(X)*, by substituting *bruce* for X. In the same way, *european* and *german* are resolved in RS2 and RS3 respectively. In RS4, the empty clause is derived, which means that the input Swan theory proves $white(bruce)$, so the Swan theory is incompatible.

$$\begin{array}{cl}
white(bruce) \Longrightarrow & \\
\cline{1-1}
european(bruce) \wedge swan(bruce) \Longrightarrow & european(X) \wedge swan(X) \Longrightarrow white(X) \\
\cline{1-1}
german(bruce) \wedge swan(bruce) \Longrightarrow & german(X) \Longrightarrow european(X) \\
\cline{1-1}
swan(bruce) \Longrightarrow & \Longrightarrow german(bruce) \\
\cline{1-1}
\Longrightarrow & \Longrightarrow swan(bruce)
\end{array}$$

Figure 4.3: RSs of the incompatibility w.r.t. $white(bruce)$ of the Swan theory: *same colour is used to highlight a pair of propositions that unify with each other.*

In 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.

$$\begin{aligned}&\text{A theory } \mathbb{T} \text{ is insufficient w.r.t. a preferred proposition } \alpha,\\ &\qquad\text{iff } \mathbb{T}' \nvdash \perp, \text{ where } \mathbb{T}' = \mathbb{T} \cup \{\neg\alpha\},\ \alpha \in \mathcal{T}(\mathbb{PS}).\end{aligned} \tag{4.6}$$

The insufficiency of the Swan theory is caused by the failure to prove the preferred proposition $black(bruce)$. The inference of the insufficiency starts by negating the proposition as the initial goal: $black(bruce) \Longrightarrow$. In the Swan theory, there is no available literal to resolve the goal, so the inference stops with $black(bruce) \Longrightarrow$ left. Because the inference ends with a non-empty clause, it can be concluded that the input Swan theory fails to prove the preferred proposition $black(bruce)$. Therefore, the Swan theory is insufficient.

Insufficiency search has to be exhaustive in order to not miss any resolution possibility. This kind of proof, which results in un-resolvable non-empty clauses, is partial or a failed proof, which is called the *evidence of the insufficiency* in this thesis and the *evidence* for short.

The statistical data of the faults of a theory can be formalised as in Definition 4.3.1. Given a theory ($\mathbb{T}$) and a $\mathbb{PS}$ ($\mathbb{PS}$), the fault sets of that theory are defined as the incompatibility set ($\mathcal{IC}(\mathbb{T},\mathbb{PS})$) and the insufficiency set ($\mathcal{IS}(\mathbb{T},\mathbb{PS})$).

Definition 4.3.1 (The Incompatibility and Insufficiency sets)**.** *Let:*

$$\mathcal{IS}(\mathbb{T},\mathbb{PS}) = \{\phi \in \mathcal{T}(\mathbb{PS}) | \mathbb{T} \nvdash \phi\}$$
$$\mathcal{IC}(\mathbb{T},\mathbb{PS}) = \{\phi \in \mathcal{F}(\mathbb{PS}) | \mathbb{T} \vdash \phi\}$$

Theorem 4.3.1. *A theory is fault-free if and only if its incompatibility and insufficiency sets are both empty.*

The detection of incompatibility and insufficiency of a theory are independent processes that can be done separately. Again, as a special case of incompatibility, constraint violation is detected during the detection of incompatibility. Therefore, it is immaterial which fault should be checked for first.

4.4 Theoretical Basis of Repair Generation

In this section, a set of repair postulates is proposed, after which the search strategy for fault-free theories is discussed, and then candidates of repair techniques: conceptual change, belief revision and abduction, are analysed based on the faults of insufficiency and incompatibility.

4.4.1 Repair Postulates

Resembling the idea of the AGM postulates for belief revision (Gärdenfors, 1992), several postulates are formulated in this section. These postulates are used for guiding the design of repair generation, rather than properties to be proved. The underlying motivation is that we want to generate reasonable repairs when abduction, belief revision and conceptual change are combined.

Postulate 1 (Repair Generation)**.** *Let $\mathbb{T}$ and $\mathbb{T}'$ be the logical theory and its repaired theory, respectively. The empty set is written as $\varnothing$, and α, β are ground propositions. The repaired theory $\mathbb{T}'$ fully follows $\mathbb{PS}$ by:*

$$\forall\alpha,\ \alpha \in \mathcal{T}(\mathbb{PS}),\ \mathbb{T}' \vdash \alpha$$
$$\forall\beta,\ \beta \in \mathcal{F}(\mathbb{PS}),\ \mathbb{T}' \nvdash \beta$$

Then the postulates for the repair generation are as the following.

1. *$\mathbb{T}$ can be repaired, iff $\mathcal{T}(\mathbb{PS}) \cap \mathcal{F}(\mathbb{PS}) = \varnothing$*
   An input theory can always be repaired, if and only if $\mathbb{PS}$ is not self-contradictory. It can be seen that $\mathbb{PS}$ must not be self-contradictory in order to achieve a faithful theory.

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 $\mathbb{PS}$.

3. *If $\mathbb{T}$ is a Datalog theory, then $\mathbb{T}'$ is also a Datalog theory.*
   This postulate requires that the repaired theory also follows the restrictions of Datalog if the input theory is a Datalog theory. We assume that no matter which logic we are working in, repairs should not break the restriction of the logical format convention. Otherwise, it would come up with something that the system which uses $\mathbb{T}$ or $\mathbb{T}'$ cannot understand, and cause unexpected problems.

4. *Make minimal changes.*
   All the information conveyed by axioms is considered to be important, and we do not want any unnecessary informational loss. Thus, the signature and all axioms of the input theory are considered to be correct so that repairs should not change them unless they are involved in an unwanted proof.

From the flowchart of ABC in Figure 4.1, it can be seen that the first two postulates have been employed: C1 checks whether $\mathbb{PS}$ is self-contradictory and fault-free theories are output before C3 without further changes. In our system, the maxi-choice belief revision (Gärdenfors, 1992) is chosen, which corresponds to the minimal change. Meanwhile, all possible repairs are generated by applying the repair techniques to the axioms or signature elements which are involved in the proofs of a fault. However, as the over-production of repairs is a common issue for each of the individual repair technique, and it becomes worse for the combination repair system (Urbonas et al., 2020). Thus, optimal repairs, which solve the most faults with the fewest operations, will be selected as the final repair solutions, discussed in §6.4. When repairs are generated for the currently targeted fault, these new repairs will be combined with the existing ones which solve previous faults. Then all of these currently possible repair combinations will be evaluated, from which the sub-optimal ones will be pruned to reduce the search space of the final solutions.

When the repair solution is compound, the prune sub-optimal repair is a necessary stage in the whole repair process. Then a desired repair is possibly pruned as the sub-optimal. But it rarely happens in our evaluation. If one hopes to loosen the restriction of only applying the optimal repairs, a weight w can be employed to keep all the repairs whose corresponding theories have at least w fewer faults than others.

4.4.2 Search Strategy for Fault-free Theories

The effect of a repair on the logical consequence of a theory is hard to predict, which means that one repair can fix multiple faults, and alternatively, one repair can introduce new faults while fixing its targeted fault. Then applying repairs of all detected faults at the same time can be inaccurate. For example, there are two faults $F1$ and $F2$ in $\mathbb{T}$, of which $R1$ and $R2$ are repairs generated for $F1$ and $F2$ based on $\mathbb{T}$ respectively. It is possible that $R1$ repairs $F1$ and its side effect fixes $F2$, in which case $R2$ becomes either unnecessary or even incorrect by introducing new faults to the repaired theory of $R1$. Then it is not a minimal change to apply both $R1$ and $R2$ to $\mathbb{T}$.

To avoid the above discussed mistake, a basic solution is the search strategy shown in Figure 4.4, where repairs are applied respectively, and then faults are detected based on each resulting semi-repaired theories. As shown in Figure 4.4, repair $R1$ is generated for fixing fault $F1$, after which apply $R1$ to $\mathbb{T}$ resulting $\mathbb{T}_1$, and then detect the faults of $\mathbb{T}_1$ and generate corresponding repairs until a fault-free theory is found or no repairable theory is left. The same process is conducted for $F2$. Note that either $\mathcal{R}_3$ or $\mathcal{R}_4$ fixes the fault $\mathcal{F}_3$.

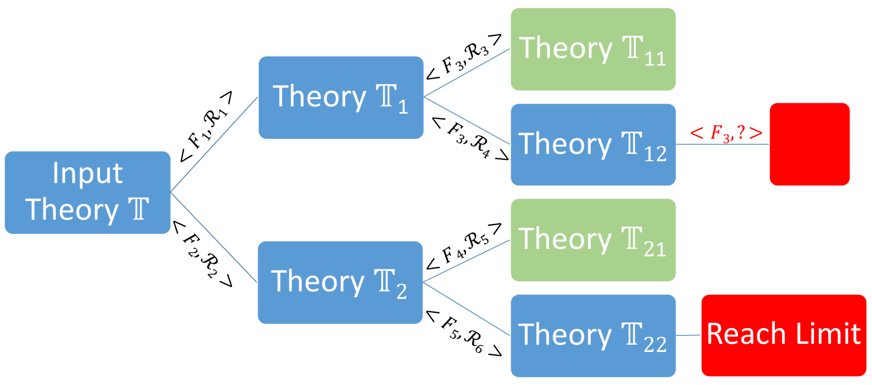


Figure 4.4: Basic search strategy of fault-free theories: *each edge is labelled with $< x, y >$, where x refers to the repaired fault and y the repair which fixes x; a search branch terminates with a fault-free theory denoted by green nodes or a non-repairable theory denoted by red nodes*[3] *or when it reaches resource threshold denoted by the red nodes with 'Reach Limit'.*

A search branch terminates when it satisfies one of the following conditions.

1. A fault-free theory is found.

2. No repair can be generated for the detected fault.

3. The resource threshold is reached, which is either the depth limit of search branches or the limit of the number of applied repairs in the current theory.

Although we aim at generating all possible repairs, it is possible that a repair is not covered by any candidate repair technique that ABC combines. Then it is possible that a fault cannot be repaired. As a repair can introduce new faults when repairing one, a search branch can be infinite in an extreme situation. Thus, the resource threshold is essential for ABC to stop that infinite search.

The search strategy in Figure 4.4 is basic because the refinement mechanism of combining commutative repair plans as MSCRs and the sub-optimal pruning has been developed to reduce the search space of fault-free theories, which are introduced in §6.3 and §6.4, respectively.

4.4.3 Repair Technique Analyses: Conceptual Change, Belief Revision and Abduction

The existing repair techniques including abduction, belief revision and conceptual change via reformation have been introduced in §3.4.4, §3.4.1 and §3.4.3 respectively. In this section, these repair techniques are combined together with the variants of the first two. The original abduction and belief revision add or delete axioms as a whole respectively, while conceptual change via reformation focuses on changing the signature of a theory. Additionally, the variants (of the first two) delete or add preconditions to an axiom[4].

The analysis of repair techniques in this section is the theoretical basis of the repair algorithm introduced in Chapter 5.

In the following discussion, the argument of a proposition is written as a vector, e.g., $p(\vec{c})$, and the argument vectors can be considered as sets in some equations, e.g., $v \in \vec{c}$ means that v is an argument of proposition $p(\vec{c})$. When an argument only contains constants, it is written as a vector in lowercase, e.g., $p(\vec{c})$. When it is a mix of variables and constants or only variables, it is represented with uppercase, e.g., $p(\vec{T})$.

4.4.3.1 Conceptual change

Conceptual change has different meanings in the literature. A clear definition of our usage is necessary. In this section, we define and analyse possible conceptual changes in a logical theory, and explain why each conceptual change is necessary.

First of all, let us start with analysing the Datalog theory of mothers ($\mathbb{T}_m$) given in Example 4.4.1. There are five facts and a rule in $\mathbb{T}_m$, whose signature is constituted by $mum/2$ and $female/1$.

Example 4.4.1. *Motherhood Theory* $\mathbb{T}_m$.

$\Longrightarrow mum(lucy, tom)$	$\Longrightarrow mum(lily, victor)$
$\Longrightarrow mum(lily, tina)$	$\Longrightarrow mum(anna, victor)$
$\Longrightarrow mum(anna, david)$	$mum(X, Y) \Longrightarrow female(X)$

In $\mathbb{T}_m$, the concept of mother is given by the predicate $mum/2$, which indicates that there are two components of the concept of mother. The components include the

[4]The domain dependent repair of deleting or adding preconditions are proposed by (McNeill and Bundy, 2007).

mother and her child. Meanwhile, mothers are always female according to the rule. Here the instances of mothers include that $lucy$ is the *mum* of tom; $lily$ is the *mum* of $tina$, etc.

From the example, it can be seen that a predicate name and the arity of that predicate are crucial for formalising the corresponding concept. The predicate symbol corresponds to the name of a concept, which is the most important part. The arity of a predicate could be seen as the number of components that constitute the corresponding concept with a certain order. Apart from the predicate name and its arity, another factor that plays a role in the signature are arguments. The two aspects of arguments of a predicate include the order of the arguments and the domain of each argument.

The order of the arguments of a predicate is relevant to how the concept is described. Taking $code(lgw, \ london)$ as the example, the code of the airport of a city is written by the predicate $code/2$, where the first argument is the code of an airport, and the second argument is the name of the city where that airport is located. Alternatively, $code'(london, \ lgw)$ represents that the first argument is the city where the airport is located, and the second argument is that airport's code. It can be seen that if the order of the arguments is different, the relation among arguments is different. Therefore, apart from the predicate name and its arity, the concept represented by a predicate also depends on the order of its arguments. Currently, reformation does not change any argument order, so changing argument order is not a RP of ABC in this thesis. But it is worthwhile to explore as a piece of future work, discussed as point 5 in §9.1. Particularly, repairing argument orders can be useful when aligning two databases where the mismatch of argument orders can happen.

Apart from the order, argument domain is another property of an argument of a predicate, defined in 4.4.1. For example, the domain of the first argument of predicate $mum/2$ in the above motherhood theory is $\mathcal{D}(mum, 1, \mathbb{T}_m) = \{anna, \ lucy, \ lily\}$. It can be concluded that there are three mothers including Anna, Lucy and Lily, according to the theory. When there is a constant disappearing or a new constant discovered, the corresponding argument domain is retracted or expanded respectively.

Definition 4.4.1 (Argument Domain)**.** *In logical theory $\mathbb{T}$, the domain of an argument w.r.t. a predicate is the set of all constants that can appear in the position of that argument in the theorems of the theory. The function $\mathcal{D}(p, \ n, \mathbb{T})$ returns the argument* *domain of the n_{th} argument of predicate p in $\mathbb{T}$.*

$$\mathcal{D}(p,i,\mathbb{T}) = \begin{cases} \{c_i | \mathbb{T} \vdash p(\vec{c}),\ c_i = \gamma(p(\vec{c}),i)\} & 1 \leq i \leq arity(p) \\ \varnothing & otherwise \end{cases} \tag{4.7}$$

where $\gamma(p(\vec{c}),i) = \gamma(p(c_1,c_2,...c_n),i) = c_i$, and $c_1,c_2..c_n$ are constants.

As a part of conceptual change, the change of argument domain could be a vital repair for the theory in sorted logic. But Datalog is unsorted, so the change of argument domains is not an option of generating a repair but a possible consequence of repairing a faulty theory because its theorems are changed.

In summary, a concept in a Datalog theory can be changed in three main ways: expansion, contraction and revision.

Definition 4.4.2 (Conceptual Changes)**.**

Concept Expansion: *A new predicate/constant is added to the signature.*

Concept Contraction: *An existing predicate/constant is removed from the signature.*

Concept Revision: *The argument domain of a predicate is changed; or the arity of a predicate is changed in the signature.*

Concept expansion and contraction are easy to understand, while concept revision is more complex. When the order of the arguments of a predicate is changed, the concept represented by the predicate is changed regarding the relationship between the arguments. On the other hand, the change of arity corresponds to the change of the numbers of the components that constitute the concept which the predicate represents.

An example of concept revision of argument order is aligning the theories from two different systems. Assume that it is the airport code from two different systems to align, and one of them write the airport code as $code(lgw, london)$, while another as $code(london, lgw)$. We decide to uniform the signature into the way $code(lgw, london)$ is expressed, then the arguments order of all instances of $code/2$ from the other system will have to be reversed.

Concept revision of arity change is necessary when the number of the components of a concept changes. We need to increase the arity of a predicate when there is a new component of the concept discovered which cannot be described by the existing arity. Assume that Lily is Victor's birth mother, and all other mothers are also birth mothers, except that Anna is the *stepmother* of Victor. For distinguishing different types of mothers, a new argument could be added as the component of describing motherhood type. The revised concept mum and the corresponding theory which correctly represents the mum concept are shown in Example 4.4.2, with changes highlighted in red.

Example 4.4.2. *Enriched Motherhood Theory by Conceptual Change.*

$$\implies mum(lucy, tom, birth)$$
$$\implies mum(lily, tina, birth)$$
$$\implies mum(anna, david, birth)$$
$$\implies mum(lily, victor, birth)$$
$$\implies mum(anna, victor, step)$$
$$mum(X, Y, Z) \implies female(X)$$

It can be seen that the new argument tags each instances of the concept mother into different groups. As for the rule, it is a new variable added as the new argument: without the evidence of a non-female mum, all types of mum are concluded to be female. This is an application of the minimal change: because there is no fault caused by that rule, the rule is kept to express the same relation that all mums are female after propagating the arity increment of mum.

Conceptual revision does not equal conceptual contraction followed by conceptual expansion. For example, one may think that increasing the arity of a predicate would be equivalent to remove the predicate's old arity and then add that predicate's new arity. However, that is not true because the contraction causes informational loss, and the expansion function does not know what axioms to add back. As aforementioned, Example 4.4.1 is the input theory and Example 4.4.2 is the output theory repaired by conceptual revision, where the fact that Anna is a different type of mum is represented by the third argument of *mum*. To illustrate why the combination of contraction and expansion cannot achieve what revision does, we model the process by assuming that a contraction function deletes all axioms of $mum/2$ from the input theory. Then all axioms of that predicate are lost. Even if the contraction function backs up the axioms being deleted, they cannot be directly used as the input for conceptual expansion because they have an incorrect arity. The gap is that a new argument needs to be added to all the backed up axioms, which creates new axioms with a new arity for the expansion function to add. A detailed question is what the value of the new argument of these axioms' should be. The discovery of the value is the trigger of the conceptual revision: one axiom is of a different type from the others. In summary, a conceptual revision function can be formalised by combining a conceptual contraction function which records all deleted axioms, a new argument supplement function which creates new terms for the new argument positions, and a conceptual expansion function.

The components of a concept are dynamic. When a component becomes useless for a concept, the corresponding argument should be abandoned, thus the arity of the corresponding predicate decreases. For example, to distinguish British citizen and British resident, the relevant information about David is represented as below:

$\Longrightarrow country(david, uk, citizen)$
$\Longrightarrow country(david, uk, resident)$

Imagine that policy changed, and the resident is the same as the citizen from all the perspectives, then the last argument becomes useless. The axiom should be written as:

$\Longrightarrow country(david, uk)$

Consequently, the arity of the predicate decreases along with the deletion of the argument which describes the useless feature of the concept.

In this section, we have defined conceptual changes, which include concept expansion, concept contraction and concept revision. We analysed that concept revision is necessary by giving examples. However, we have not given an algorithm of conceptual change yet. In the next section, reformation, an algorithm of conceptual change is given for repairing the defined faults in our framework, together with abduction and belief revision.

4.4.3.2 Repair Techniques Analysis based on the Defined Faults

The strategy for addressing each fault is shown in Table 4.1: 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 RS in it, while unblocking a proof requires building all necessary RSs.

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

Table 4.1: The strategy of repairing a faulty theory $\mathbb{T}$.

In this section, we discuss the repair strategy of technique candidates: abduction, belief revision, and conceptual change, for repairing the defined faults. An algorithm, reformation, is introduced for generating repairs of conceptual changes. Additionally, a rule can be changed by deleting or adding a precondition (McNeill and Bundy, 2007), which are the variants of abduction and belief revision.

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 this Thesis.

1. The application of the original abduction. 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, and generalising specific facts into rules results in a better understanding and a theory from which we can derive all theorems and discover similar facts that follow the rules. For example, 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 propositions, while not proving any proposition in $\mathcal{F}(\mathbb{PS})$.

By summarising relevant theorems of the goal to prove, a new rule can be generalised from the body of relevant theorems and the head of the goal. Alternatively, the new rule can be found by analogising an existing and relevant rule.

2. The variant of abduction. The targeted observation can be unprovable due to improperly restricted preconditions in rules. Then the observation can be provable by deleting those improper preconditions, in which case the repaired rule is the explanation of the corresponding observation. This kind of fault is detected when a rule is involved in a partial proof of a preferred proposition and at least one of its preconditions is unprovable. Then that proof can be unblocked by removing all unprovable preconditions from that rule. In this case, the repair of deleting the preconditions is classified as a variant of abduction.

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 the 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 this Thesis.

1. The application of the original belief revision. To repair the incompatibility, belief revision is used to block the proofs of the violative propositions from $\mathcal{F}(\mathbb{PS})$. In order to block these unwanted proofs, belief revision deletes axioms from the logical theory.
2. The variant of the belief revision. When the preconditions of a rule is not strict enough to summarise its conclusion, new preconditions should be added to block the unwanted proofs. Thus, to block the unwanted proof of a violative proposition, a new precondition is added to a rule which has been involved in the unwanted proof. This new precondition should be unsatisfied w.r.t. the violative proposition. In this case, the repair is classified as a variant of belief revision that blocks unwanted proofs.

Original Reformation.

Reformation 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 was originally in FOL. By changing the signature of a theory, reformation inverts the outcome of the targeted RS, which contributes to blocking or unblocking proofs. The repair types of the reformation for FOL include changing the name or the arity of a predicate or function, changing the name of a constant or changing a constant into a term containing a variable.

Reformation in this Thesis.

Because our work is based on Datalog, reformation is much simpler than that for FOL. For example, changing a constant into a term containing a variable would introduce a function. As function symbols are not permitted in the circumstance of Datalog, that kind of repair is omitted in our RPs of reformation.

Table 4.2 gives reformation RPs on different types of resolution problems, in which R1 and R2 block unwanted but successful RSs (incompatibilities), while R3 unblocks wanted but failed RS (insufficiencies). Each repair plan can be applied to either the left side or the right side of the unification. For simplicity, only the ones applied to the

RS	Code	Reformation Repair Plan
$p(\overrightarrow{S}^n) \equiv p(\overrightarrow{T}^n)$	R1.1	Rename input side p: $p(\overrightarrow{S}^n) \not\equiv p'(\overrightarrow{T}^n)$.
	R1.2	Rename one constant in $\overrightarrow{T}^n$, e.g., let $s_i \neq t'_i$, then $p(\overrightarrow{S}^n) \not\equiv p(\overrightarrow{T'^n})$..
	R1.3	Add distinguished constants c, c' as new arguments: $p(\overrightarrow{S}^n, c) \not\equiv p(\overrightarrow{T}^n, c')$.
$c \equiv S$	R2	Weakening S to c': $c \neq c'$.
$p_1(\overrightarrow{S}^n) \not\equiv p_2(\overrightarrow{T}^m)$	R3	Replace either side by the other: $p_1(\overrightarrow{S}^n) \equiv p_1(\overrightarrow{S}^n)$.

Table 4.2: RPs of reformation for inverting the outcome of RS in this Thesis: *the changes are highlighted in red; unification is denoted by* $\equiv$*, of which goal/sub-goal is on the left-hand side, and input clause on the right; '=' denotes the equality between terms or predicate symbols; P is a predicate; S and T are either constants or variables;* $n \geq 0, m \geq 0$*;* $1 \leq i \leq n$*; c and c′ are constants,* $c \neq c'$*; p refers to a constant when its arity is 0;the last row is for repairing the insufficiencies while the others for incompatibilities.*

right side are denoted in Table 4.2.

R1 and R2 make the originally successful RSs fail: R1.1 renames a predicate on either side, R1.2 changes one argument of the predicate on either side, and R1.3 increases arity by 1, and then adds distinguished constants as arguments to predicates on both sides. R1.3 is an application of concept revision. When the resolution is between a constant goal and a variable from input clause side, it can always succeed unless weakening the variable to a distinguished constant as in R2. R3 replaces $p_2(\overrightarrow{T}^m)$ by $p_1(\overrightarrow{S}^n)$ for making the originally failed RS between $p_1(\overrightarrow{S}^n)$ and $p_2(\overrightarrow{T}^m)$ successful. Alternatively, $p_1(\overrightarrow{S}^n)$ can be replaced by $p_2(\overrightarrow{T}^m)$ to unblock the RS.

Considering the definition of $\mathbb{PS}$, reformation is capable of repairing both incompatibility by R1 or R2, and insufficiency by R3.

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

Table 4.3: Technique candidates for repairing faults.

To summarise, all technique candidates can be summarised according to the defined faults of the incompatibility and the insufficiency in Table 4.3.

Although this section analyses each repair technique based on the defined faults, it does not give the corresponding algorithms' details. The algorithm will be described in the next chapter.

4.5 Summary

In this chapter, a $\mathbb{PS}$ is defined for formalising user's requirements. The possible ways in which a faulty theory can fail in following its $\mathbb{PS}$ are by being insufficient or incompatible. When there are constraint axioms, a Datalog theory can be faulty due to constraint violations. The insufficiency is caused by the lack of wanted proofs, while both of the incompatibility and the constraint violation are caused by the existence of unwanted proofs. Thus, the constraint violation will be seen as a special case of incompatibility for a compact discussion in this thesis.

A set of repair postulates are proposed for guiding the system to generate reasonable repairs. Based on the signature of a logical theory, conceptual change is defined including concept expansion, concept extraction and concept revision. The definition of conceptual change can clearly reflect the concept changes both on the level of predicates and the arity and arguments of a predicate. Repair techniques, including abduction, belief revision, reformation and their variants are analysed based on the defined faults of incompatibility and insufficiency. The above work constitute the theoretical basis of the repair algorithm of ABC introduced in Chapter 5.

Chapter 5

ABC Repair System: Repair Algorithm

In this chapter[1], the repair algorithm of the ABC Repair System (ABC) is presented, which combines abduction, belief revision and conceptual change for repairing faulty Datalog theories (Li et al., 2018).

As the Preferred Structure ($\mathbb{PS}$) is not allowed to change in this thesis, the following generation condition is applied to the repair algorithm to protect $\mathbb{PS}$.

- **Generation condition:** *the targeted proposition to repair is not from the Preferred Structure.*

This algorithm includes two parts: generating the Repair Plan (RP) based on a targeted Resolution Step (RS) in the proof of an incompatibility or an evidence of a partial proof of an insufficiency, which is discussed in §5.1, and adding a new rule by analogising a useful existing rule for fixing an insufficiency, which is introduced in §5.2. The evidence of a partial proof is called the *evidence* for short in this thesis.

5.1 Repair Plans

A proof or an evidence is a sequence of Resolution Steps (RSs). By targeting at one RS to block or unblock, Repair Plans (RPs) are automatically generated following the tables of the repair algorithm in this section.

[1]Recall that all predicate symbols and constants start with lowercase letters, and variables start with capital letters.

5.1.1 Incompatibility Repair

The proof of an incompatibility can be broken by blocking one RS of its. Table 5.1 gives the RPs which break the targeted RS in an incompatibility's proof.

RS	Repair Plan		Technique
$p(\overrightarrow{S}^n) \equiv p(\overrightarrow{T}^n)$	CR1. Rename predicate on either side: $p(\overrightarrow{S}^n) \not\equiv p'(\overrightarrow{T}^n)$ or $p'(\overrightarrow{S}^n) \not\equiv p(\overrightarrow{T}^n)$.		Reformation
	CR2. $s_i \equiv t_i \equiv c$, rename either to c'.	Then: $\overrightarrow{S}^n \not\equiv \overrightarrow{T'}^n \vee \overrightarrow{S'}^n \not\equiv \overrightarrow{T}^n$	
	CR3. $(s_i = X \wedge t_i = c) \vee (s_i = c \wedge t_i = X)$, weaken variable X to c'.		
	CR4. Add different constants: $p(\overrightarrow{S}^n, c) \not\equiv p(\overrightarrow{T}^n, c')$.		
	CR5. Delete the axiom A_u.		Belief Revision
	CR6. Add an unprovable precondition $q(\overrightarrow{Z^w})$ to A_u.		Belief Revision (variant)

Table 5.1: Incompatibility Repair Plans: block a proof by breaking one RS of it. *All symbols mean the same as those in Table 4.2; the input clause is on the right of the unification; $p(\overrightarrow{S}^n)$ is either from a precondition of a rule or from $\mathbb{PS}$, while only in the former case $p(\overrightarrow{S}^n)$ is changeable, and A_u is the input clause where $p(\overrightarrow{T}^n)$ comes from.*

New expressions: $p', \overrightarrow{Y'}, c'$ are introduced into the theory in Table 5.1. Without additional information, the meaning of a new expression is unknown to ABC. Therefore, dummy terms are employed to name new expressions.

For newly introduced constants, we provide *dummy* as the initial argument. Instead of a fixed *dummy*, we add a serial number at the end of it, such as *dummy*1, *dummy*2, and so on. The serial number is crucial for distinguishing different *dummy* constants. For example, in a motherhood theory, we added distinguished constants to the mother predicate to describe the various types of mother. At the beginning, we introduced *dummy*1 and *dummy*2, which could be interpreted as birth mother and stepmother respectively. If a new type of mother needs to be represented, such as godmother, which is different from the previously introduced *dummy*1 and *dummy*2, then this new type could be represented by constant *dummy*3. As for the predicate symbol, we retain the original predicate symbol between *dummy* and the serial number, e.g., *dummymum*1. The serial number and the connection are represented by $\mathcal{S}$ and $\oplus$ respectively. For example, $dummy \oplus \mathcal{S} = dummy2$ when $\mathcal{S}$ is 2.

In the following section, the details of how to apply each incompatibility repair plan to the input theory are introduced. The application of a substitution $\{c/X\}$ to an axiom A is written as $A \cdot \{c/X\}$. The RS to break is the *target RS*, and the original proposition in an input axiom to be changed is the *target proposition*. Because $\mathbb{PS}$ is given by the user so it is not allowed to be changed. Therefore, the proposition from $\mathbb{PS}$ cannot be the target proposition. The input axiom where a target proposition comes form is a *target axiom*.

CR1. Break $p(\overrightarrow{S}^n) \equiv p(\overrightarrow{T}^n)$ by renaming predicate symbol: $p(\overrightarrow{S}^n) \not\equiv p'(\overrightarrow{T}^n)$.

The target RS can be broken by renaming the predicate symbol on its either side unless the goal on the left is from $\mathbb{PS}$. When renaming the right side, it is a local repair without trace-back or propagation needed and the target axiom is the input axiom which occurs in the targeted RS. Alternatively, when renaming the left side, the target is the original input axiom which introduced the goal $p(\overrightarrow{S}^n) \implies$ needs to be found by tracing back to the previous RSs.

In a target axiom, predicate p is renamed as p' which is constituted by the original predicate symbol p, and the serial number $\mathcal{S}$, given in equation 5.1.

$$p' = dummy \oplus p \oplus \mathcal{S} \tag{5.1}$$

Example 5.1.1. *Faulty Motherhood Theory.*

$$mum(X,Y) \wedge mum(Z,\ Y) \implies X = Z \tag{R1}$$

$$\implies mum(lily,\ tina) \tag{A1}$$

$$\implies mum(lily,\ victor) \tag{A2}$$

$$\implies mum(anna,\ victor) \tag{A3}$$

$\mathcal{F}(\mathbb{PS}) = \{anna = lily,\ lily = anna\},\ \mathcal{T}(\mathbb{PS}) = \varnothing$

$$\cfrac{\cfrac{\cfrac{anna = lily \implies}{mum(anna,Y) \wedge mum(lily,Y) \implies} \quad mum(X,Y) \wedge mum(Z,Y) \implies X = Z}{mum(lily,victor) \implies} \quad \implies mum(anna,victor)}{\implies} \quad \implies mum(lily,victor)$$

Figure 5.1: The inference of the incompatibility of $anna = lily$ in Example 5.1.1[2].

The theory in Example 5.1.1 is incompatible due to the theorem $anna = lily$. Assume its the blue RS of $mum(anna,Y) \equiv mum(anna,\ victor)$ in Figure 5.1 to break, and the target axiom is $A3$, then $dummy \oplus p \oplus \mathcal{S}$ is $dummymum1$, as given in Example

5.1.2. The change is highlighted in red. If interpreting $dummymum1$ as step-mum, then the repaired theory concludes that Anna is Victor's step mother.

Example 5.1.2. *CR1: Repaired Motherhood Theory by Renaming mum.*

$$mum(X,Y) \wedge mum(Z,Y) \Longrightarrow X = Z \quad (R1)$$

$$\Longrightarrow mum(lily, tina) \quad (A1)$$

$$\Longrightarrow mum(lily, victor) \quad (A2)$$

$$\Longrightarrow dummymum1(anna, victor) \quad (A3)$$

$\mathcal{F}(\mathbb{PS}) = \{anna = lily,\ lily = anna\},\ \mathcal{T}(\mathbb{PS}) = \varnothing$

CR2. Break $p(\overrightarrow{S}^n) \equiv p(\overrightarrow{T}^n)$ by renaming a constant: $p(\overrightarrow{S}^n) \not\equiv p(\overrightarrow{T'}^n)$.

Instead of renaming the predicate, we break the RS by changing a constant in the arguments. Similarly, this repair plan can be applied to the target axiom which 1) on the right side of the unification; or 2) introduced the goal in a previous RS and the corresponding argument is a constant originally[3].

Due to the minimal change, only one argument T_i is renamed. Then the repaired arguments satisfy equation 5.2.

$$\exists i,\ T_i' \neq T_i;\ \forall 1 \leq j \leq n,\ j \neq i,\ T_j' = T_j \tag{5.2}$$

Similar to predicate renaming, dummy term is given as the new argument. The new argument can be written as:

$$T_i' = dummy \oplus T_i \oplus \mathcal{S} \tag{5.3}$$

In Example 5.1.1, assume its the green RS, $mum(lily, victor) \equiv mum(lily, victor)$, to break and the target axiom is $A2$, then T_i can be either $lily$ or $victor$. Their corresponding new constants are $dummylily1$ and $dummyvictor1$, shown in Example 5.1.3 and 5.1.4 respectively. The repaired theory is as the following. The former describes that $dummylily1$ is another name of $anna$ so they refer to one individual, while the latter denotes that $dummyvictor1$ is $lily$'s child, whose name was recorded incorrectly as victor in the original theory.

[3]In the case that argument originally is a variable and is bound to a constant during the resolutions, the repair plan is CR3 which weakens a variable to a constant, rather than CR2 which renames a constant.

Example 5.1.3. *CR2: Repaired Theory by Renaming lily.*

$$mum(X,Y) \wedge mum(Z, Y) \implies X = Z \quad (R1)$$
$$\implies mum(lily, tina) \quad (A1)$$
$$\implies mum(dummylily1, victor) \quad (A2)$$
$$\implies mum(anna, victor) \quad (A3)$$

$\mathcal{F}(\mathbb{PS}) = \{anna = lily,\ lily = anna\},\ \mathcal{T}(\mathbb{PS}) = \varnothing$

Example 5.1.4. *CR2: Repaired Theory by Renaming victor.*

$$mum(X,Y) \wedge mum(Z, Y) \implies X = Z \quad (R1)$$
$$\implies mum(lily, tina) \quad (A1)$$
$$\implies mum(lily, dummyvictor1) \quad (A2)$$
$$\implies mum(anna, victor) \quad (A3)$$

$\mathcal{F}(\mathbb{PS}) = \{anna = lily,\ lily = anna\},\ \mathcal{T}(\mathbb{PS}) = \varnothing$

CR3. Break $p(\vec{S}^n) \equiv p(\vec{T}^n)$ by weakening a variable X to c': $p(\vec{S}^n) \not\equiv p(\vec{T'}^n)$.

When X is on the right side, it is a local repair where the target axiom is the input axiom of this RS. When X is on the left, then trace-back is needed and the repair will be applied to the input axiom which originally introduced X. Notice that all the occurrences of X in the axiom should be replaced with c'.

The choice of the constant c' depends on how the target axiom R_X is involved in the wanted proofs of propositions in $\mathcal{T}(\mathbb{PS})$. These involvements are summarised as as the following cases. Recall that the target RS to break can be simplified as $X \equiv c$. The constants to which X is bound in essential proofs are called X's *essential constants*.

Case1. If R_X is essential w.r.t. $\mathcal{T}(\mathbb{PS})$, and c is X's unique essential constant, then this repair is not applicable.

Case2. If the targeted axiom R_X is not essential w.r.t. $\mathcal{T}(\mathbb{PS})$, then:

$$c' = \begin{cases} c_w & \exists c_w/X \in \mathbb{S}_w \\ dummy \oplus \mathcal{S} & \text{otherwise} \end{cases} \quad (5.4)$$

where $\mathbb{S}_w$ is the set of substitutions of X in wanted proofs.

The repair of weakening a variable to a constant is not applicable in the first case, because weakening X to any c', where $c' \neq c$, will introduce new insufficiency.

Example 5.1.5. *Faulty Motherhood Theory with Types.*

$$\begin{aligned}
mum(X,Y,\ W) \wedge mum(Z,\ Y,\ W) &\implies X = Z && \text{(R1)}\\
&\implies mum(lily,\ tina,\ birth) && \text{(A1)}\\
&\implies mum(lily,\ victor,\ step) && \text{(A2)}\\
&\implies mum(anna,\ victor,\ step) && \text{(A3)}
\end{aligned}$$

$\mathcal{F}(\mathbb{PS}) = \{lily = anna,\ anna = lily\},\ \mathcal{T}(\mathbb{PS}) = \varnothing$

The theory in Example 5.1.5 is incompatible because it derives $anna = lily$ and $lily = anna$. Semantically, $R1$ represents that all types of mother are unique, but combining A2, A3 and $\mathbb{PS}$, it can be concluded that step mother may be not unique. The unwanted proof of $anna = lily$ is shown in Figure 5.2.

$$\dfrac{\dfrac{\dfrac{anna = lily \implies \quad mum(X,Y,W) \wedge mum(Z,Y,W) \implies X = Z}{mum(anna,Y,W) \wedge mum(lily,Y,W) \implies} \quad \implies mum(anna,victor,step)}{mum(lily,victor,step) \implies} \quad \implies mum(lily,victor,step)}{\implies}$$

Figure 5.2: The inference of the incompatibility of $anna = lily$ in Example 5.1.5.

Assuming that the targeted unification is between W and $step$, then the targeted input axiom is $R1$. As $R1$ is not essential, so c' is $dummy1$ according to Equation 5.4. By weakening W as $dummy1$, the repaired theory is as the following, where only mothers of $dummy1$ type are unique.

Example 5.1.6. *CR3: Repaired Motherhood Theory with Types.*

$$\begin{aligned}
mum(X,Y,\ dummy1) \wedge mum(Z,\ Y,\ dummy1) &\implies X = Z && \text{(R1)}\\
&\implies mum(lily,\ tina,\ birth) && \text{(A1)}\\
&\implies mum(lily,\ victor,\ step) && \text{(A2)}\\
&\implies mum(anna,\ victor,\ step) && \text{(A3)}
\end{aligned}$$

$\mathcal{F}(\mathbb{PS}) = \{lily = anna,\ anna = lily\},\ \mathcal{T}(\mathbb{PS}) = \varnothing$

These dummy terms can be either repaired to something more meaningful on subsequent repairs or renamed to something with meanings by the user. For example, rule $R1$ will cause a lot of insufficiencies until $dummy1$ is instantiated to $birth$, and then the corresponding axioms can be interpreted as birth mothers.

CR4. Break $p(\vec{S}^n) \equiv p(\vec{T}^n)$ by adding new argument: $p(\vec{S}^n, c_1) \not\equiv p(\vec{T}^n, c_2)$.

Because $\mathbb{PS}$ is given by the user, and it is not allowed to change in this thesis, therefore, this repair plan could be generated only if p does not occur in $\mathbb{PS}$.

The targeted RS is broken by adding a new argument but with different constants to the end of the target propositions on both side of the unification. Let the serial number $\mathcal{S} = s$, then the constants to be added are:

$$c_1 = dummy \oplus s \tag{5.5}$$

$$c_2 = dummy \oplus (s+1) \tag{5.6}$$

Due to SL-Resolution, $p(\vec{T}^n)$ is guaranteed to be an original proposition from an input axiom. However, $p(\vec{S}^n)$ could be a result of substitutions. Therefore, we need to trace the proof back to find the other target proposition where the precondition $p(\vec{S}^n)$ comes from.

$$p(\vec{S'}^n) \cdot \sigma = p(\vec{S}^n) \tag{5.7}$$

In equation 5.7, $p(\vec{S'}^n)$ is the target precondition and σ represents the substitutions. Then c_1 is added to the target precondition $p(\vec{S'}^n)$ and c_2 are added to $p(\vec{T}^n)$, alternatively, c_2 to $p(\vec{S'}^n)$ and c_1 to $p(\vec{T}^n)$, where $c_1 \neq c_2$ and c_1 is the default value which will be propagated to other instances of the predicate.

Newly added constants repair an incompatibility by defining types of the predicate. For example, the bird theory $\mathbb{T}_B$ below is incompatible, of which the inference is shown in Figure 5.3.

Example 5.1.7. *Bird Theory $\mathbb{T}_B$.*

$$bird(X) \implies fly(X) \tag{R1}$$

$$penguin(Y) \implies bird(Y) \tag{R2}$$

$$\implies penguin(lucy) \tag{A1}$$

$\mathcal{F}(\mathbb{PS}) = \{fly(lucy)\}$, $\mathcal{T}(\mathbb{PS}) = \varnothing$

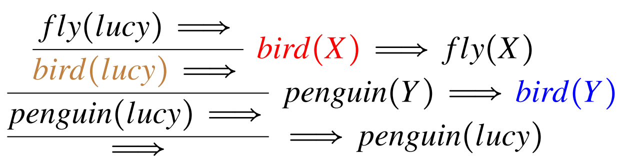


Figure 5.3: The inference of the incompatibility of $fly(lucy)$ in Example 5.1.7.

Assume the RS of $bird(lucy)$ and $bird(Y)$ is the one to break in Figure 5.3. The right side of the unification is the blue $bird(Y)$ in $R2$. The brown $bird(lucy)$ is the left side of the unification which is not an original proposition. Tracing the proof back, it can be seen that the brown $bird(lucy)$ is inherited from the red $bird(X)$ in $R1$. To repair the incompatibility, new arguments are added to the propositions with predicate $bird$ in these two original rules. The corresponding repaired theory is $\mathbb{T}'_B$.

Example 5.1.8. *CR4: Repaired Bird Theory* $\mathbb{T}'_B$.

$$\begin{aligned} bird(X, dummy1) &\Longrightarrow fly(X) && (R1)\\ penguin(Y) &\Longrightarrow bird(Y, dummy2) && (R2)\\ &\Longrightarrow penguin(lucy) && (A1) \end{aligned}$$

$\mathcal{F}(\mathbb{PS}) = \{fly(lucy)\}$, $\mathcal{T}(\mathbb{PS}) = \varnothing$

The interpretation of this repair is that $dummy1$ represents the flyable type and $dummy2$ non-flyable type. According to the repaired theory, penguin is a non-flyable bird.

Heuristic 1 (Mirror Avoidance)**.** *Only switch the value of* $dummy1$ *and* $dummy2$ *when the theory has another assertion containing the predicate, and that assertion is not involved in the target unification.*

Without the mirror avoidance heuristic, there would be another repaired bird theory given in Example 5.1.9, which semantically equals to $\mathbb{T}'_B$ by interpreting $dummy2$ as the flyable and $dummy1$ as the non-flyable. Therefore, it is a redundant mirror of $\mathbb{T}'_B$. This kind of redundancy is avoided by the mirror avoidance heuristic[4]. Heuristic 1 does not bring new repairs but contributes to reducing the search space of final solutions by avoiding redundant search branches which are syntactically different but semantically identical, e.g., Example 5.1.8 and Example 5.1.9. By employing Heuristic 1, the repair in Example 5.1.9 will be avoided as the redundant because it is produced after the other in our implementation. Since they are identical in semantics, it does not matter which is avoided.

[4]Similar heuristics can be applied to CR1 and CR2, e.g., do not switch the side of the unification on which the dummy term is used when the predicate or the constant symbol only occurs twice in the theory.

Example 5.1.9. *Redundancy: Mirror Bird Theory.*

$$
\begin{aligned}
bird(X, dummy2) &\Longrightarrow fly(X) && \text{(R1)}\\
penguin(Y) &\Longrightarrow bird(Y, dummy1) && \text{(R2)}\\
&\Longrightarrow penguin(lucy) && \text{(A1)}
\end{aligned}
$$

$\mathcal{F}(\mathbb{PS}) = \{fly(lucy)\}$, $\mathcal{T}(\mathbb{PS}) = \varnothing$

Because overloading predicates is not allowed in Datalog. Therefore, arity increment has to be propagated to all the occurrences of that predicate whose arity is increased. Before discussing the propagation, the independent variable and passive propositions are defined.

- Independent Variable: *a new variable that does not exist in that rule.*
- Passive Propositions: *all the propositions written in the predicate whose arity will be increased, except the two involved in the target RS which triggers the arity increment.*

For automatically propagating arity increment, the propagation postulates addressing the two challenges, are proposed in Postulate 2. The first three postulates are proposed based on the idea of minimal change: the original axioms are considered as correct unless they are involved in an unwanted proof. The last postulate makes the repaired theory satisfy Datalog's grammar.

Postulate 2 (Propagation of Arity Increment)**.** *To increase the arity of a predicate, the propagation of that arity increment follows the steps below, where c_1 or c_2 are the arguments newly added to the target propositions*[5].

1. *If the axiom of a target proposition also contains passive propositions, then add the same constant to these passive propositions.*
2. *Add the default constant c_1 to the passive propositions of assertions.*
3. *Add an independent variable to those passive propositions of rules.*
4. *If a passive proposition is the head of a rule which does not contain the target proposition, and the added new variable has not been added into the body of that rule, then the new variable will be added into at least one predicate in the body of that rule.*

[5]Between the pair of new constants, the one with the smaller serial number is the default constant.

5. *New argument is added to the end of old arguments.*

In the original theory, all instances of the predicate can be considered as of the same default type which was omitted in the representation. Then the process of adding new arguments to break an unwanted proof can be seen as a unique type is revealed by the unwanted proof, which is represented by c_2 and the previously omitted type is represented by c_1. Similarly, different preconditions with the target predicate in a rule is considered to be of same type before the arity increment, so keeping them consistent by adding same constant argument to the target rule according to the first postulate, or adding an independent variable to passive rules according to the third postulate. The independent variable represents that the rule is applicable to all types. Rule A' in Example 5.3.3 on page 109 is an example of adding an independent variable to a rule.

Example 5.1.10. *CR4: Semi-Repaired Theory before Propagation.*

$$mum(X,Y) \wedge mum(Z, Y, dummy1) \implies X = Z \quad \text{(R1)}$$
$$\implies mum(lily, tina) \quad \text{(A1)}$$
$$\implies mum(lily, victor, dummy2) \quad \text{(A2)}$$
$$\implies mum(anna, victor) \quad \text{(A3)}$$

$\mathcal{F}(\mathbb{PS}) = \{anna = lily,\ lily = anna\},\ \mathcal{T}(\mathbb{PS}) = \varnothing$

Taken the incompatible motherhood theory in Example 5.1.1 as the example, assume that the green RS is targeted to break in the inference shown in Figure 5.1 on page 69. Then $dummy2$ is added as the new argument of predicate mum to the input axiom $A2$ and $dummy1$ to the second precondition in $R1$, where the goal $mum(lily, victor)$ inherits from. The corresponding semi-repaired theory before propagating the arity increment is shown in Example 5.1.10.

Example 5.1.11. *CR4: Repaired Theory $\mathbb{T}'_M$ after Propagation.*

$$mum(X,Y, dummy1) \wedge mum(Z, Y, dummy1) \implies X = Z \quad \text{(R1)}$$
$$\implies mum(lily, tina, dummy1) \quad \text{(A1)}$$
$$\implies mum(lily, victor, dummy2) \quad \text{(A2)}$$
$$\implies mum(anna, victor, dummy1) \quad \text{(A3)}$$

$\mathcal{F}(\mathbb{PS}) = \{anna = lily,\ lily = anna\},\ \mathcal{T}(\mathbb{PS}) = \varnothing$

According to the propagation postulates, $dummy1$ is added as the new argument to other propositions of *mum*, then the repaired theory($\mathbb{T}'_M$) is fault-free, shown in Example 5.1.11.

The interpretation of the repair is that the original two arguments of $mum/2$ are not enough to represent all the components of the concept of mother. We need a new argument to describe the type of mother. In the repaired theory, $mum(lily, victor, dummy2)$ is of the special type. Lily could be Victor's stepmother, or godmother or another type of mother, which is unknown without extra information. On the other hand, the default constant and the independent variable represent the other axioms' neutrality in the new argument. The neutrality could be because an axiom's feature represented by the new argument is still unknown, or the feature is the most common and basic one, e.g., birth-mum. These dummy terms can be either repaired to something more meaningful on subsequent repairs or renamed to something with meanings by users. For example, if the user renames $dummy1$ as *birth*, then the corresponding axioms can be interpreted as birth mothers.

Example 5.1.12 shows another possible repair by switching the new constants in the targeted unification: add $dummy2$ to $R1$'s second precondition and $dummy1$ to $A2$. Then according to the first propagation postulate, the first precondition of $R1$ will be added with $dummy2$ to keep preconditions of one rule in same kind. The Propagation of adding $dummy1$ to $A1$ and $A3$ remains the same. As *mum* occurs more than twice, switching the assignment won't result in a mirror theory. In this case, the special type ($dummy2$) of mum is unique, while the other type of mum are the majority.

Example 5.1.12. *CR4: Another Arity Increment of the Motherhood Theory.*

$$mum(X,Y, dummy2) \wedge mum(Z, Y, dummy2) \implies X = Z \quad \text{(R1)}$$

$$\implies mum(lily,tina,dummy1) \quad \text{(A1)}$$

$$\implies mum(lily,victor,dummy1) \quad \text{(A2)}$$

$$\implies mum(anna,victor,dummy1) \quad \text{(A3)}$$

$\mathcal{F}(\mathbb{PS}) = \{anna = lily,\ lily = anna\},\ \mathcal{T}(\mathbb{PS}) = \varnothing$

By interpreting $dummy1$ as stepmother, this repaired theory says that *victor* has two stepmothers.

CR5. Break $p(\overrightarrow{S}^n) \equiv p(\overrightarrow{T}^n)$ by deleting the axiom where $p(\overrightarrow{T}^n)$ comes from.

This is a local repair where neither trace-back nor propagation is needed. The target axiom where $p(\overrightarrow{T}^n)$ comes from is the input clause in the current RS. To avoid introducing insufficiency, this repair is applicable when the targeted axiom is not essential to $\mathcal{T}(\mathbb{PS})$.

By deleting any of $A2$ or $A3$ or $R1$ from the original theory in Example 5.1.1, its incompatibilities can be fixed. One of the fault-free theory is given by Example 5.1.13.

Example 5.1.13. *CR5: Repaired Motherhood Theory by Deleting A2.*

$$mum(X,Y) \wedge mum(Z,Y) \implies X = Z \qquad \text{(R1)}$$
$$\implies mum(lily, tina) \qquad \text{(A1)}$$
$$\cancel{\implies mum(lily, victor)} \qquad \text{(A2)}$$
$$\implies mum(anna, victor) \qquad \text{(A3)}$$

$\mathcal{F}(\mathbb{PS}) = \{anna = lily,\ lily = anna\}$, $\mathcal{T}(\mathbb{PS}) = \varnothing$

CR6. Break $p(\overrightarrow{S^n}) \equiv p(\overrightarrow{T^n})$ by adding an unprovable precondition $q(\overrightarrow{Z^w})$.

This repair does not need trace-back, nor propagation. The key problem to address is how to find the target precondition $q(\overrightarrow{Z^w})$, which blocks the unwanted proof by adding a new restriction.

Postulate 3 (Unprovable Precondition)**.** *Let R be the input axiom where $p(\overrightarrow{T^n})$ comes from, and the repaired axiom R' is generated by adding $q(\overrightarrow{Z^w})$ to R. The corresponding theory of R and R' are $\mathbb{T}$ and $\mathbb{T}'$ respectively. Then $q(\overrightarrow{Z^w})$ should make them have the following properties.*

1. *The repair does not introduce new insufficiencies: $\mathbb{T}' \vdash P$, if $\mathbb{T} \vdash P$, $P \in \mathcal{T}(\mathbb{PS})$.*
2. *The targeted proof is blocked by adding $q(\overrightarrow{Z^w})$ to R.*
3. *The new precondition is relevant to R in the way of sharing at least one variable in its arguments $\overrightarrow{Z^w}$ with other propositions in R. That variable is called the shared variable.*

The first two properties reflect the correctness of the repair and the last one corresponds to the reasonableness and the generalisation of the repaired rule.

To denote why the shared variable mentioned in the last property is important, Example 5.1.14 gives relevant rules of PhD awards, where *RB* is the basic rule to repair which says that one is awarded with doctor degree if she/he writes a thesis. But the fact is that she/he has to pass the viva to get the degree.

To repair rule *RB* shown below, a new precondition $viva(lucy, pass)$ with no variable but constants is added in *RI*1, which makes it incorrect because it says that whether an author of a thesis can get the doctor degree relies on the condition that there is a Lucy who passes a viva. The new precondition fails in restricting to the same individual as the other propositions do, because without a variable, it is too specific to summarise general cases. Usually, the more variables $q(\overrightarrow{Z^w})$ contains, the more general the cases it represents.

Example 5.1.14. *Rules for PhD Awards.*

$$thesis(X) \wedge author(Y,X) \implies degree(Y,doctor) \quad \text{(RB)}$$

$$thesis(X) \wedge author(Y,X) \wedge viva(lucy,pass) \implies degree(Y,doctor) \quad \text{(RI1)}$$

$$thesis(X) \wedge author(Y,X) \wedge viva(Z,pass) \implies degree(Y,doctor) \quad \text{(RI2)}$$

$$thesis(X) \wedge author(Y,X) \wedge viva(Y,pass) \implies degree(Y,doctor) \quad \text{(RC)}$$

If the new precondition contains a variable, but it is different from all the others, as $viva(Z, pass)$ in *RI*2 does, then the resulting rule is still incorrect because it concludes that as long as there is one person who passes a viva, an author of a thesis can get the degree, even if they are different individuals. The fault is caused by the mismatch of variables, so the new precondition cannot restricts on the target individual as the others do.

By adding a new precondition $viva(Y, pass)$, *RC* is formalised which is a correct rule describing that the author of a thesis should also pass the viva to get a doctor degree. Here X is the shared variable, which enable all preconditions to restrict on the same individual. Therefore, the shared variable lets the new precondition work together with the others, so the third desired property of $q(\overrightarrow{Z^w})$ is claimed.

However, if it is $viva(X, pass)$ not $viva(Y, pass)$ added to *RB*, then it is a madly incorrect rule which requires a thesis to pass a viva in an unsorted logic[6]. That mistake can be avoided by checking the theorems of the theory, which would contain the fact that PhD students have passed viva while no theorem of a thesis passing viva. Therefore, the theorems of the input theory play an important role in the computation of the new precondition.

In the rest of the discussion, Example 5.1.15 is used to illustrate the computation of unprovable preconditions. It can be seen that the swan theory is incompatible w.r.t. $white(bruce)$ and insufficient w.r.t. $black(bruce)$. Only incompatibility is repaired in

[6]In a sorted logic, it would be an ill-formed formula.

this section and the other in the next section by analogical abduction[7].

Example 5.1.15. *Swan Theory.*

$swan(X) \Longrightarrow white(X)$ (R1)	$german(X) \Longrightarrow european(X)$ (R2)
$\Longrightarrow swan(lily)$ (A1)	$\Longrightarrow german(lily)$ (A4)
$\Longrightarrow swan(lucy)$ (A2)	$\Longrightarrow european(lucy)$ (A5)
$\Longrightarrow swan(bruce)$ (A3)	$\Longrightarrow australian(bruce)$ (A6)

$$\mathcal{T}(\mathbb{PS}) = \{black(bruce), white(lily), white(lucy)\}$$
$$\mathcal{F}(\mathbb{PS}) = \{white(bruce), black(lily), black(lucy)\}$$

As rule $R1$ is involved in the incompatibility in the example, it is the target rule which will be augmented with a new precondition. To calculate the precondition $q(\overrightarrow{Z^w})$ which has the desired three properties, the following data is needed, where R is the target rule.

- A predicate q/w in the signature;
- Theorem information of the target predicate q based on $\mathbb{T}_2$: $\mathbb{TQ}(q)$, where $\mathbb{T}_2 = \mathbb{T} - \{R\}$; the set of the arguments of these theorems $\mathbb{A}(q)$ and the domain of the ith argument of q/w in $\mathbb{T}_2$: $\mathcal{D}(q, i, \mathbb{T}_2)$, where $1 \le i \le w$.

$$\mathbb{TQ}(q) = \{q(\overrightarrow{ca^w}) | \ \mathbb{T}_2 \vdash q(\overrightarrow{ca^w})\} \tag{5.8}$$
$$\mathbb{A}(q) = \{\overrightarrow{ca^w} | \overrightarrow{ca^w} = (ca_1, ca_2, ...ca_w), \ q(\overrightarrow{ca^w}) \in \mathbb{TQ}(q)\} \tag{5.9}$$
$$\mathcal{D}(q, i, \mathbb{T}_2) = \{ca_i | \overrightarrow{ca^w} = (ca_1, ca_2, ...ca_w), \ q(\overrightarrow{ca^w}) \in \mathbb{TQ}(q)\} \tag{5.10}$$

- The set of the wanted proofs is: $\mathbb{P}(\mathbb{T}, \mathcal{T}(\mathbb{PS}))$, and the unwanted proof to block is: $P_{block}(\mathbb{T}, \beta)$, where $\beta \in \mathcal{F}(\mathbb{PS}) \wedge R \dot{\in} P_{block}$. $R \dot{\in} P_{block}$ denotes that R is involved in the proof P_{block}.
- $\gamma(R, P)_k$: the substitutions of the variables in the target rule R in the proof P.

$$\forall 1 \le k \le L, \ \gamma(R, P)_k = \begin{cases} \{c_x/X, \ c_y/Y, ...\}, \ R \dot{\in} P \\ \varnothing, \ Otherwise \end{cases} \tag{5.11}$$

where X, Y... are the variables in R and c_x, c_y... are the constants which are substituted for these variables respectively; L is the number of R being used in P.

[7] Axiom $A6$ provides essential information for the abduction of the rule that Australian swans are black in Example 5.2.2 in the next section.

Example 5.1.16. *CR6: Unprovable Precondition Computation 1.*

The set of theorems of the input theory:
$\{european(lily),\ white(lily),\ white(lucy),\ white(bruce)\}$
The following predicate candidates are taken as examples to discuss:
$european/1,\ australian/1,\ german/1.$
The proofs are simplified with only its input axioms: $R1, A1, A2, A3$. The following P_1, P_2, P are the proofs of $white(lily)$, $white(lucy)$ and $white(bruce)$ respectively.
Sufficiency proof: $\mathbb{P}(\mathbb{T}, \mathcal{T}(\mathbb{PS})) = \{P_1,\ P_2\}$, where $P_1 = \{R1, A1\}$, $P_2 = \{R1, A2\}$.
Incompatibility proof to block: $P_{block}(\mathbb{T}, white(bruce)) = \{R1, A3\}$.
The corresponding $\gamma(R1, P_i)_k$ are:
$\gamma(R1, P_1)_1 = \{lily/X\}$, $\gamma(R1, P_2)_1 = \{lucy/X\}$, $\gamma(R1, P_{block})_1 = \{bruce/X\}$.

The theorems information w.r.t. each predicate candidate is:

- $\mathbb{TQ}(european) = \{european(lily),\ european(lucy)\}$;
  $\mathbb{A}(european) = \{(lily),\ (lucy)\}$
  $\mathcal{D}(european, 1, \mathbb{T}) = \{lily,\ lucy\}$

- $\mathbb{TQ}(australian) = \{australian(bruce)\}$;
  $\mathbb{A}(australian) = \{(bruce)\}$
  $\mathcal{D}(australian, 1, \mathbb{T}) = \{bruce,\}$

- $\mathbb{TQ}(german) = \{german(lily)\}$;
  $\mathbb{A}(german) = \{(lily)\}$
  $\mathcal{D}(german, 1, \mathbb{T}) = \{lily,\}$

When there is a sequence of substitutions w.r.t. one variable, only the final result is represented, e.g., $Y/X \cdot Z/Y \cdot c/Z$ is written as c/X.

Based on γ_i, the set of the constants which substitute variable X in R in all the wanted proofs is $\mathbb{SC}(R, X)$, called the *bound constants* of variable X.

$$\mathbb{SC}(R, X) = \{c_{xi} | \forall P_i \in \mathbb{P}(\mathbb{T}, \mathcal{T}(\mathbb{PS})), c_{xi}/X \in \gamma(R,\ P_i)\} \tag{5.12}$$

Now the candidates of the argument of the precondition can be calculated by the equation below, where v_i is the ith argument of $q(\overrightarrow{Z^w})$. If the set of bound constants of variable X is a subset of the domain of the ith argument of the predicate $\mathcal{D}(q, i, \mathbb{T})$, then $v_i = X$. Otherwise, write v_i as a constant in its domain.

$$v_i = \begin{cases} X, \ \mathbb{SC}(R,X) \subseteq \mathcal{D}(q,i,\mathbb{T}) \\ c, \ otherwise, \forall c \in \mathcal{D}(q,i,\mathbb{T}) \end{cases} \tag{5.13}$$

where $\forall 1 \leq i \leq w$.

To make the new precondition resolvable in wanted proofs, its argument $\overrightarrow{Z^w}$ should satisfy the resolvable condition in 5.14, which guarantees that the assigned arguments $\overrightarrow{Z^w}$ match at least one argument of a theorem of p/w under the substitutions of each wanted proofs. Meanwhile, after bounding $\overrightarrow{Z^w}$ with the substitutions of the proof to block $\gamma(R, P_{block})$, the resulting argument cannot be unified with any theorem's argument, so that the corresponding precondition is irresolvable under P_{block}. The irresolvable condition is written in 5.15, where R is $P_1 \wedge P_2, ..., \wedge P_m \implies p(\overrightarrow{T^n})$

$$\forall P_i \in \mathbb{P}(\mathbb{T}, \mathcal{T}(\mathbb{PS})), \exists \overrightarrow{ca^w} \in \mathbb{A}(q) \text{ let } \overrightarrow{Z^w} \cdot \gamma(R, P_i) \equiv \overrightarrow{ca^w} \tag{5.14}$$

$$(q(\overrightarrow{Z^w}) \cdot \gamma(R, P_{block}) \not\equiv p(\overrightarrow{T^n})) \wedge (\nexists \, \overrightarrow{ca^w} \in \mathbb{A}, \overrightarrow{Z^w} \cdot \gamma(R, P_{block}) \equiv \overrightarrow{ca^w}) \tag{5.15}$$

The following definition summarises the computation of the unprovable precondition to add to R.

Definition 5.1.1 (Unprovable Precondition Computation)**.** *Get one predicate q/w, where q/w exists in the signature but not in R. Based on Equation 5.13, calculate the argument candidates $\overrightarrow{Z^w}$s. If one candidate $\overrightarrow{Z^w}$ 1) contains at least one variable, and 2) satisfies the resolvable condition 5.14 and 3) the irresolvable condition 5.15, then the unprovable precondition $q(\overrightarrow{Z^w})$ satisfies Postulate 3.*

Based on the above definition, the faulty swan theory is fixed by adding the unprovable precondition $european(X)$ to $R1$. Then the unwanted proof of $white(bruce)$ is blocked.

It can be seen that the precondition calculated by Definition 5.1.1 is defined to have the first property in Postulate 3. If the following equations in Theorem 5.1.1 are true, then that precondition also has the other two properties.

Example 5.1.17. *CR6: Unprovable Precondition Computation 2.*

$\mathbb{SC}(R1,X) = \{lily,\ lucy\}$. The precondition candidate of each predicate is:

- $q = european$:
  $\mathcal{D}(q,1,\mathbb{T}) = \{lily, lucy\}$; so $\mathbb{SC}(R1,X) \subseteq \mathcal{D}(european,1,\mathbb{T})$.
  Thus, $v_1 = X$, so the candidate precondition is $european(X)$.

- $q = australian$:
  $\mathcal{D}(q,1,\mathbb{T}) = \{bruce\}$; so $\mathbb{SC}(R1,X) \nsubseteq \mathcal{D}(australian,1,\mathbb{T})$.
  Thus, $v_1 = bruce$, so the candidate precondition is $australian(bruce)$.

- $q = german$:
  $\mathcal{D}(q,1,\mathbb{T}) = \{lily\}$ so $\mathbb{SC}(R1,X) \nsubseteq \mathcal{D}(german,1,\mathbb{T})$.
  Thus, $v_1 = lily$, so the candidate precondition is $german(lily)$.

Due to the last two candidates do not contain variables, they are dropped, and then only the first one is the new precondition to add. The semi-repaired theory is:

$$european(X) \wedge swan(X) \implies white(X) \qquad (R1')$$
$$german(X) \implies european(X) \qquad (R2)$$

$$\implies swan(lily) \quad (A1) \qquad \implies german(lily) \quad (A4)$$
$$\implies swan(lucy) \quad (A2) \qquad \implies european(lucy) \quad (A5)$$
$$\implies swan(bruce) \quad (A3) \qquad \implies australian(bruce) \quad (A6)$$

Theorem 5.1.1 (Correctness of Definition 5.1.1)**.** *If the following equations are true, then the precondition found by Definition 5.1.1 has the desired properties given by Postulates 3. Equation 5.16 corresponds to the first property, and 5.17 corresponds to the second property.*

$$\forall P,\ P \in \mathcal{T}(\mathbb{PS}), \mathbb{T} \vdash P \implies \mathbb{T}' \vdash P \tag{5.16}$$
$$(\mathbb{A} \cup R') \nvdash \beta,\ \mathbb{A} = \{\alpha | \alpha \neq R,\ \alpha \dot{\in} P_{block}(\mathbb{T},\beta)\} \tag{5.17}$$

where R is the original rule, R' and $\mathbb{T}'$ are the repaired rule and the corresponding repaired theory: $\mathbb{T}' = \mathbb{T} \cup \{R'\} - \{R\}$; the proof to block is of $\mathbb{A} \cup \{R\} \vdash \beta$.

Proof of Equation 5.16.

1. **When R is not involved in the proofs of $\mathbb{T}\vdash P$.** Let R be the axiom where $p(\overrightarrow{T}^n)$ comes from, and $\mathbb{T}_2=\mathbb{T}-\{R\}$, then $\mathbb{T}_2\vdash P$.
   Due to $\mathbb{T}'=\mathbb{T}_2+\{R'\}$, then: $\mathbb{T}'\vdash P$ because $\mathbb{T}'$ is monotonic[8].

2. **When R is involved in the proofs of $\mathbb{T}\vdash P$.** In the original theory $\mathbb{T}$, $\forall P\in\mathbb{P}(\mathbb{T},\mathcal{T}(\mathbb{PS}))$, the RS in which R resolves with the goal clause in P can be written as the inference in 5.18. By applying the substitution $\gamma(R,P_i)$, all subgoals are resolved and the empty clause is resulted. Here the resolution is based on Equation 5.11, so (5.11) is written after the input theory to make it clear.

$$\dfrac{\dfrac{p(\overrightarrow{S}^n)\wedge G_1,\ldots\wedge G_m\Longrightarrow \qquad P_1,\ldots\wedge P_m\Longrightarrow p(\overrightarrow{T}^n)}{(P_1,\ldots\wedge P_m\wedge G_1\ldots\wedge G_m\Longrightarrow)\cdot\gamma(R,P_i)}\ \mathbb{T}(5.11)}{\Longrightarrow} \tag{5.18}$$

Replace R with R', the updated inference is shown below. The new precondition is highlighted in red. Based on Equation 5.14, $q(\overrightarrow{Z^w})\cdot\gamma(R,P)$ is rewritten as $q(\overrightarrow{ca}^w)$. According to Equation 5.8, $q(\overrightarrow{ca}^w)$ is a theorem of $\mathbb{T}_2$, so the subgoal can be resolved away. As for the remaining subgoals, they are resolvable based on the original theory $\mathbb{T}$ according to the inference above.

$$\dfrac{\dfrac{\dfrac{\dfrac{p(\overrightarrow{S}^n)\wedge G_1,\ldots\wedge G_m\Longrightarrow \qquad q(\overrightarrow{Z^w})\wedge P_1,\ldots\wedge P_m\Longrightarrow p(\overrightarrow{T}^n)}{(q(\overrightarrow{Z^w})\wedge P_1,\ldots\wedge P_m\wedge G_1\ldots\wedge G_m\Longrightarrow)\cdot\gamma(R,P)}\ (5.14)}{q(\overrightarrow{ca}^w)\wedge(P_1,\ldots\wedge P_m\wedge G_1\ldots\wedge G_m\Longrightarrow)\cdot\gamma(R,P)}\ \mathbb{T}_2(5.8)\star}{(P_1,\ldots\wedge P_m\wedge G_1\ldots\wedge G_m\Longrightarrow)\cdot\gamma(R,P_i)}\ \mathbb{T}\star}{\Longrightarrow} \tag{5.19}$$

For the repaired theory $\mathbb{T}'$, the starred RSs, the last two in inference 5.19, only succeeds when the input theory is monotonic. As $\mathbb{T}'$ is monotonic and $\mathbb{T}'=\mathbb{T}_2+\{R'\}$, all the theorems of $\mathbb{T}_2$ are also theorems of $\mathbb{T}'$.

$$\text{If } \mathbb{T}' \text{ is monotonic, then } \forall G,\mathbb{T}_2\vdash G\Longrightarrow\mathbb{T}'\vdash G. \tag{5.20}$$

Thus, the first starred RS succeeds with $\mathbb{T}'$. Same as the second starred RS, it succeeds with monotonic $\mathbb{T}'$. There are two cases of the second starred RS.

   1. When R is not involved in the process of resolving the remaining goal clause $(P_1,\ldots\wedge P_m\wedge G_1\ldots\wedge G_m\Longrightarrow)\cdot\gamma(R,P_i)$ in 5.18, then $\mathbb{T}_2$ resolves the goal clause. Based on 5.20, $\mathbb{T}'$ also resolves the goal clause.

[8]The UNAE introduced in §6.1 is the only exception of non-monotony. But the equalities will be computed in advance and will not be changed during the repair process, so the theory is monotonic during repair generation.

2. When R is involved, then there are extra $q(\overrightarrow{v_i^w})$ introduced to the goal clause. According to Equation 5.14, $\forall i$, the introduced $q(\overrightarrow{v_i^w})$ in a wanted proof are resolvable.

In summary, Equation 5.16 is true in the ABC Repair System.

Proof of Equation 5.17.
The proof to block can be written as the inference below where $\gamma(R,P_{block})_1$ is the first substitution of R.

$$\cfrac{\beta \Longrightarrow}{\cfrac{p(\overrightarrow{S}^n)\wedge G_2\ldots\wedge G_n \Longrightarrow}{\cfrac{(P_1,\ldots\wedge P_m\wedge G_2\ldots\wedge G_n)\cdot\gamma(R,P_{block}) \Longrightarrow}{\Longrightarrow}\;\mathbb{A}\cup\{R\}} \quad P_1,\ldots\wedge P_m \Longrightarrow p(\overrightarrow{T}^n)}\;\mathbb{A}$$

By replacing R with R', the partial proof of Equation5.17 is as the following.

$$\cfrac{\beta \Longrightarrow}{\cfrac{p(\overrightarrow{S}^n)\wedge G_2\ldots\wedge G_n \Longrightarrow}{\cfrac{(q(\overrightarrow{Z^w})\wedge P_1,\ldots\wedge P_m\wedge G_2\ldots\wedge G_n)\cdot\gamma(R,P_{block}) \Longrightarrow}{q(\overrightarrow{Z^w})\cdot\gamma(R,P_{block}) \Longrightarrow}\;\mathbb{A}\cup\{R'\}} \quad q(\overrightarrow{Z^w})\wedge P_1,\ldots\wedge P_m \Longrightarrow p(\overrightarrow{T}^n)}\;\mathbb{A} \tag{5.21}$$

Based on the second half of the irresolvable condition 5.15 and Equation 5.9:

$$\nexists\, q(\overrightarrow{ca^w}),\ \mathbb{T}_2 \vdash q(\overrightarrow{ca^w}),\ q(\overrightarrow{Z^w})\cdot\gamma(R,P_{block}) \equiv q(\overrightarrow{ca^w})$$

Therefore, $q(\overrightarrow{Z^w})\cdot\gamma(R,P_{block})$ is irresolvable to $\mathbb{T}_2$.
Due to $\mathbb{A}\subseteq\mathbb{T}_2$, thus $q(\overrightarrow{Z^w})\cdot\gamma(R,P_{block})$ is irresolvable to $\mathbb{A}$.
Based on the first part of the irresolvable condition 5.15, $q(\overrightarrow{Z^w})\cdot\gamma(R,P_{block})$ does not resolve with the head of R'.
Then conclude that $q(\overrightarrow{Z^w})\cdot\gamma(R,P_{block})$ is irresolvable to $\mathbb{A}\cup\{R'\}$. So Equation 5.17 is true whose failed inference is 5.21 and its goal clause cannot be fully resolved away due to $q(\overrightarrow{Z^w})$.

All of the repair algorithm's incompatibility repair operations have been discussed based on each repair plan. There can be new expressions introduced to the repaired theory. However, they lack meanings, which could be addressed by using background knowledge, which is a part of future work, introduced as point 4 in §9.1. Currently, it needs the user to interpret.

5.1.2 Insufficiency Repair

For insufficiency detection, each preferred proposition is negated as the initial goal which would be resolved by the axioms in the theory. When a goal cannot be resolved, the corresponding insufficiency is detected with its partial proof as the evidence. An evidence can be completed into a proof by building all necessary RSs. Table 5.2 gives RPs for building a targeted RS.

RS	Repair Plan	Technique
$p_1(\overrightarrow{S}^n) \not\equiv p_2(\overrightarrow{T}^m)$	SR1. Reform $p_2(\overrightarrow{T}^m)$ or reform $p_1(\overrightarrow{S}^n)$	Reformation
	SR2. If $s_i \neq t_i$ and $p_2(\overrightarrow{T}^m)$ is from a rule, extend constant t_i to a variable Z.	
	SR3. Add the assertion $p_1(\overrightarrow{S}^n)$.	Abduction
	SR4. Add a rule which proves $p_1(\overrightarrow{S}^n)$.	
	SR5. Delete $p_1(\overrightarrow{S}^n)$, a precondition from its original input axiom.	Abduction (variant)

Table 5.2: Insufficiency repair plans: unblock a proof by building the targeted RSs.

Similarly, all symbols in Table 5.2 mean the same as in Table 4.2: $p_1(\overrightarrow{S}^n)$ is an irresolvable remaining goal in the evidence of a partial proof, while $p_2(\overrightarrow{T}^m)$ is a theorem of the input theory.

In Table 5.2, $p_1(\overrightarrow{S}^n)$ represents the unprovable goal, and $p_2(\overrightarrow{T}^m)$ is a positive literal from the theory, which is either an assertion, or the head of a rule. The algorithm of applying each Repair Plan (RP) for repairing the insufficiency is discussed below, where $\mathcal{F}(\mathbb{PS})$ is default to be the empty set if it is not given explicitly in examples below for simplicity.

SR1. Fix the failed RS $p_1(\overrightarrow{S}^n) \not\equiv p_2(\overrightarrow{T}^m)$ by rewriting either side as the other.

The repair can be applied to either side of the unification but not when the left goal proposition is the negated $\mathbb{PS}$ member. To rewrite $p_2(\overrightarrow{T}^m)$ as $p_1(\overrightarrow{S}^n)$, the following changes need to be applied to the original input proposition where $p_2(\overrightarrow{T}^m)$ comes from.

1. If $p_1 \not\equiv p_2$, then rename p_2 to p_1.

2. If $\overrightarrow{S}^n \not\equiv \overrightarrow{T}^m$, then replace $\overrightarrow{T}^m$ with $\overrightarrow{T'}^n$ by following the operations below.

(a) If $n < m$, then $\forall i < n$, if $T_i \neq S_i$, give S_i as T_i', and delete the extra $m - n$ arguments in the end of $p_2(\overrightarrow{T'^m})$.

(b) If $n \geq m$, $\forall i < n, T_i' \neq S_i \vee \nexists T_i'$, give S_i as T_i'.

One special case of this repair is that the left goal proposition is the negated $\mathbb{PS}$ member which has the same predicate with an axiom in the theory, but their arities are different. Then the arity in the theory of that predicate will be updated, which means the argument changes will be propagated to all the instances of that predicate to avoid predicate overload.

In Example 5.1.18, the insufficiencies are caused by the mismatched arity of *mum* between $\mathbb{PS}$ and the theory. Here the problematic RSs include:

$$mum(diana, william) \not\equiv mum(diana, william, birth) \tag{5.22}$$

$$mum(camilla, william) \not\equiv mum(camilla, william, step) \tag{5.23}$$

Example 5.1.18. *SR1: The Input Theory.*

$$\implies mum(diana, william) \tag{A1}$$

$$\implies mum(camilla, william) \tag{A2}$$

$\mathcal{T}(\mathbb{PS}) = \{mum(diana, william, birth), mum(camilla, william, step)\}$

To repair the faults, the arity of *mum* is increased by giving *birth* as the new argument of $mum(diana, william)$, and *step* of $mum(camilla, william)$. The produced fault-free theory is shown below.

Example 5.1.19. *SR1: The Repaired Theory.*

$$\implies mum(diana, william, birth) \tag{A1}$$

$$\implies mum(camilla, william, step) \tag{A2}$$

SR2. Fix the failed RS $s_i \neq t_i$ by extending either to variable Z.

This is the reverse of CR3. By extending the constant s_i or t_i to a variable Z, the original failed RS becomes successful: $Z \equiv s_i$ or $Z \equiv t_i$.

The input theory in Example 5.1.20 is the semi-repaired theory from Example 5.1.24, where the proof of $female(kate)$ is failed due to $william \neq george$. By extending *william* to a variable Y, the insufficiency w.r.t. $female(kate)$ is fixed.

Example 5.1.20. *SR2: The Input.*

$$mum(X, william) \implies female(X)$$
$$\implies mum(diana, william)$$
$$\implies mum(kate, george)$$
$$\mathcal{T}(\mathbb{PS}) = \{female(kate)\}$$

Example 5.1.21. *SR2: The Repair.*

$$mum(X, Y) \implies female(X)$$
$$\implies mum(diana, william)$$
$$\implies mum(kate, george)$$
$$\mathcal{T}(\mathbb{PS}) = \{female(kate)\}$$

SR3. Fix the failed RS $p_1(\vec{S}^n) \nvdash p_2(\vec{T}^m)$ by adding the assertion: $p_1(\vec{S}^n)$.

In this repair, the proposition $p_1(\vec{S}^n)$ is added as an axiom to the theory. To maintain a Datalog theory, if the ith argument of $p_1(\vec{S}^n)$ is a variable, then the variable would be replaced by a *dummy* constant in the added axiom.

Example 5.1.22. *SR3: The Input.*

$$\implies mum(camilla, william)$$
$$\mathcal{T}(\mathbb{PS}) = \{mum(diana, william)\}$$

Example 5.1.23. *SR3: The Repair.*

$$\implies mum(camilla, william)$$
$$\implies mum(diana, william)$$

SR4. Fix the failed RS $p_1(\vec{S}^n) \nvdash p_2(\vec{T}^m)$ by adding the rule shown in (5.26).

This is a local repair with no trace-back or propagation needed. The rule to add should contain at least one variable, otherwise, it does not summarise different cases. The formalisation of the new rule follows three steps.

1. When $p_2(\vec{T}^m)$ is a theorem $p_2(\vec{c}^m)$ of the input theory as the candidate of the precondition of the rule, which has at least one argument that can unified with an argument of the goal clause $p_1(\vec{S}^n)$.

$$\exists c_i \in \vec{c}^m,\ S_j \in \vec{S}^n,\ c_i = S_j \tag{5.24}$$

2. Formalise the rule candidate.

$$p_2(\vec{c}^m) \implies p_1(\vec{S}^n) \tag{5.25}$$

3. Generalise the candidate by replacing the unified arguments which occur on both sides of the implication with a variable, and leave the rest as constants.

$$(p_2(\vec{w}^m) \implies p_1(\vec{v}^n))\cdot\gamma \tag{5.26}$$

where $\gamma = \{S/c_i, S/S_j | \forall 1 \le i \le m, 1 \le j \le n,\ c_i \in \vec{c}^m,\ S_j \in \vec{S}^n,\ c_i = S_j\}$. Here different variables should be employed for different pairs of unified terms.

The selection of the precondition in the first step guarantees that there will be at least a pair of shared arguments c_i and X_i on both sides of the implication in the rule, which will be replaced with a variable in the last step. Replacing shared arguments with variables increases the general applicability of the new rule.

Except generating the rule in the above way, a new rule can be formalised by analogising an existing useful rule, which is discussed in the next section of analogical abduction.

Example 5.1.24 below is semi-repaired by adding a rule. The theorems of the input theory are the two axioms listed. If the goal is $female(diana)$, then the only candidate of precondition is $mum(diana, william)$ because the other $mum(kate, george)$ shares no argument with the goal proposition. The candidate rule is as the following.

$$mum(diana, william) \implies female(diana) \tag{5.27}$$

The above rule is generalised by replacing *diana* with variable X because *diana* occurs more than once. The corresponding theory is given below with the repair highlighted in red, which is sufficient w.r.t. $female(diana)$ but w.r.t. $female(kate)$. The remaining insufficiency can be further repaired by SR2 as in Example 5.1.20.

$$mum(X, william) \implies female(X) \tag{5.28}$$

Example 5.1.24. *SR4: The Input Theory.*	**Example 5.1.25.** *SR4: The Semi-Repair.*
$\implies mum(diana, william)$	$mum(X, william) \implies female(X)$
$\implies mum(kate, george)$	$\implies mum(diana, william)$
$\mathcal{T}(\mathbb{PS}) = \{female(diana), female(kate)\}$	$\implies mum(kate, george)$

SR5. Fix the failed RS $p_1(\overrightarrow{S}^n) \nsucceq p_2(\overrightarrow{T}^m)$ by deleting the precondition $p_1(\overrightarrow{S}^n)$.

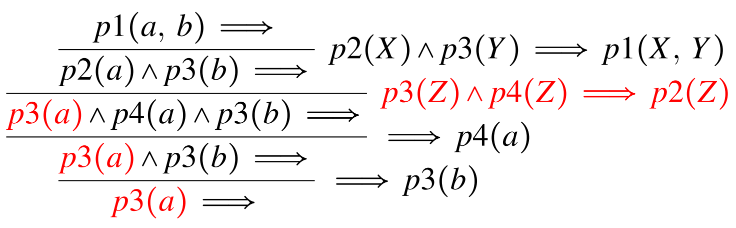
The goal $p_1(\overrightarrow{S}^n)$ comes from either a preferred proposition in $\mathbb{PS}$ or a precondition of a rule in the theory. If $p_1(\overrightarrow{S}^n)$ is a precondition, then the insufficiency could be repaired by deleting the original precondition $p_1(\overrightarrow{S'}^n)$ from its input axiom where $p_1(\overrightarrow{S}^n)$ is inherited from $p_1(\overrightarrow{S'}^n)$ by substitutions.

Taken the theory and $\mathbb{PS}$ below as an example, Figure 5.4 depicts the inference of the insufficiency.

Example 5.1.26. *SR5: The Input Theory.*

$$
\begin{array}{rcll}
p2(X)\wedge p3(Y) & \implies & p1(X,Y) & \text{(R1)}\\
p3(Z)\wedge p4(Z) & \implies & p2(Z) & \text{(R2)}\\
 & \implies & p3(b) & \text{(A2)}\\
 & \implies & p4(a) & \text{(A3)}
\end{array}
$$

$\mathcal{T}(\mathbb{PS}) = \{p1(a,b)\}$



$p1(a,b) \implies$
$p2(X)\wedge p3(Y) \implies p1(X,Y)$
$p2(a)\wedge p3(b) \implies$
$p3(Z)\wedge p4(Z) \implies p2(Z)$
$p3(a)\wedge p4(a)\wedge p3(b) \implies$
$\implies p4(a)$
$p3(a)\wedge p3(b) \implies$
$\implies p3(b)$
$p3(a) \implies$

Figure 5.4: Insufficiency caused by the unprovable precondition $p3(a)$, originally from the rule: $p3(Z)\wedge p4(Z) \implies p2(Z)$.

In this example, the original precondition is $p3(Z)$; the substitution is $\{a/Z\}$, and the resulting unprovable subgoal is $p3(a)$. The unprovable subgoal and the targeted rule are highlighted in red. By deleting the original precondition $p3(Z)$, the repaired theory is sufficient shown by Figure 5.5.

Example 5.1.27. *SR5: The Repaired Theory.*

$$
\begin{array}{rcll}
p2(X)\wedge p3(Y) & \implies & p1(X,Y) & \text{(R1)}\\
p4(Z) & \implies & p2(Z) & \text{(R2')}\\
 & \implies & p3(b) & \text{(A2)}\\
 & \implies & p4(a) & \text{(A3)}
\end{array}
$$

$\mathcal{T}(\mathbb{PS}) = \{p1(a,b)\}$

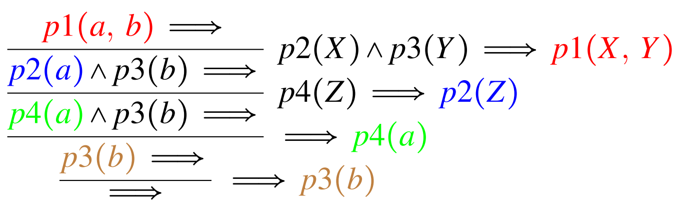


Figure 5.5: The repaired theory is sufficient after deleting precondition $p3(Z)$ from rule $p3(Z)\wedge p4(Z) \implies p2(Z)$.

In the above discussion, the details of each repair plan are given for repairing insufficiency. Different RPs may be suitable for different scenarios. Without background knowledge, it is unknown to the repair mechanism which ones are more suitable than the others. Thus, all generated RPs are applied to the theory respectively. If some of them are worse than the others in the aspect of remaining fewer faults, they will be pruned by the sub-optimal pruning mechanism discussed in §6.4.

5.2 Repairs by Analogical Abduction

Analogical abduction seeks for an explanation of a given phenomenon by analogically formalising a rule based on existing rules (Schurz, 2008; Haig, 2013). Different from (Schurz, 2008), which discovers both new rules and new concepts, we allow the analogical abduction to only produce a new rule. Consequently, our mechanism can support the scenario where a concept is already in the theory, but some of its properties remain unknown. Then when the rule-like phenomenon is observed, our mechanism can analogically formalise the rule which describes the property revealed by the phenomenon[9].

The main materials of analogical abduction include a relevant existing rule to analogise, the phenomenon to explain, and the new rule to abduce. The existing rule is called *the source rule*, which does not explain the targeted phenomenon directly, but it describes parts of the nature of that phenomenon's causes. Therefore, conceiving a new rule based on it allows the formalisation of the relational structures which explains the targeted phenomenon. That new rule is called *the target rule*. Here the targeted phenomenon are the base of the abduction, which are restricted to be written in ground assertions.

Example 5.2.1 describes the analogical abduction whose inputs are the source rule about the reflection of water waves and the phenomenon of the reflection of sound, and the output is the target rule about sound waves.

[9]The abduction of adding assertions or rules based on the failed RS without analogising existing axioms has been discussed in the last section as SR3 and SR4 respectively.

Example 5.2.1. *Sound Waves.*

Source Rule: The law of reflection of water waves.

Targeted Phenomenon to be explained: The reflection of sound.

Target Rule: Sound is consists of sound waves which analogise water waves and follow the reflection law too.

5.2.1 Algorithm of an Analogical Abduction

As far as we know, there was no automated method of analogical abduction. This section gives such an algorithm which implements analogical abduction. To conduct analogical abduction, the essential computing strategies to develop include:

1. *The selection method* which finds the relevant and useful source rules.
2. *The search method* which formalises the candidate target rules w.r.t. each source rule.
3. *The evaluation* which makes plausibility judgements about target rules.

Note that our algorithm is formalised based on the $\mathbb{PS}$.

1. The selection method.
First of all, the usefulness of a rule is defined, which corresponds to the definition of the preference entrenchment of an axiom in §6.2.1.

Specification 1 (Usefulness)**.** *A rule is considered to be useful if and only if it contributes to proving at least one proposition in the true set of* $\mathbb{PS}$.

All the proofs of the propositions in the true set of $\mathbb{PS}$ are considered useful. When a rule is involved in such a proof, we say this rule is useful.

In an existing rule, its preconditions are not guaranteed to be relevant to the targeted ground proposition and only the relevant ones are valuable for analogy.

Specification 2 (Relevant Precondition w.r.t. a Goal)**.** *In a logical theory* $\mathbb{T}$*, R is a rule in the form of* $p_1(\vec{X_1}) \wedge p_2(\vec{X_2}) \wedge \ldots \wedge p_m(\vec{X_m}) \implies q(\vec{Y})$*. Given the ground proposition goal* $g(\vec{c})$*, a precondition* $p(\vec{X})$ *in R is a relevant precondition w.r.t.* $g(\vec{c})$*, written as* $p(\vec{X}) \sim g(\vec{c})$*, if and only if it satisfies Equation 5.29, where* μ *is a substitution.*

$$p(\vec{X}) \sim g(\vec{c}) \textit{ iff } \exists \mu,\ p(\vec{X})\mu \equiv p(\vec{c_1}),\ \mathbb{T} \vDash p(\vec{c_1}),\ \vec{c_1} \cap \vec{c} \neq \varnothing \tag{5.29}$$

Based on Specification 2, a precondition is relevant when the theory has a theorem $\Longrightarrow p(\vec{c_1})$ which resolves that precondition and shares at least one constant argument with the targeted ground proposition goal $g(\vec{c})$. Thus, by adjusting the head and the variables in the rule properly, relevant preconditions can be resolved away after resolving the goal with the adjusted rule.

Recall the semi-repaired swan theory by Example 5.2.2[10], which is insufficient w.r.t. $black(bruce)$. For the goal proposition $black(bruce)$, the relevant precondition is $swan(X)$ in $R1$ because $A3$ resolves the precondition $swan(X)$ and shares the constant argument $bruce$ with the goal proposition. If the goal proposition is $white(lily)$ or $black(lily)$, then both preconditions in $R1$ are relevant preconditions.

Example 5.2.2. *The Semi-Repaired Swan Theory.*

$$european(X) \wedge swan(X) \Longrightarrow white(X) \quad \text{(R1)}$$
$$german(X) \Longrightarrow european(X) \quad \text{(R2)}$$
$$\Longrightarrow swan(lily) \quad \text{(A1)}$$
$$\Longrightarrow swan(lucy) \quad \text{(A2)}$$
$$\Longrightarrow swan(bruce) \quad \text{(A3)}$$
$$\Longrightarrow german(lily) \quad \text{(A4)}$$
$$\Longrightarrow european(lucy) \quad \text{(A5)}$$
$$\Longrightarrow australian(bruce) \quad \text{(A6)}$$
$$\mathcal{T}(\mathbb{PS}) = \{black(bruce), white(lily), white(lucy)\}$$
$$\mathcal{F}(\mathbb{PS}) = \{white(bruce), black(lily), black(lucy)\}$$

From the example, it can be seen the defined relevant preconditions describe some feature of the entity in the goal proposition, e.g., $bruce$ is the entity when $black(bruce)$ is the goal proposition, and the relevant precondition $swan(X)$ describes the feature that $bruce$ is a swan.

Based on the relevant preconditions, the relevancy degree of a rule w.r.t. a targeted ground proposition is defined as the following.

Definition 5.2.1 (Relevancy Degree of a Rule)**.** *In the logical theory $\mathbb{T}$, rule R is $p_1(\vec{X_1}) \wedge p_2(\vec{X_2}) \wedge ... \wedge p_m(\vec{X_m}) \Longrightarrow q(\vec{Y})$. The targeted ground proposition G is $g(\vec{c})$. Then the relevance degree of R w.r.t. $g(\vec{c})$ is a 2-tuple $r(R,G) = (t_1,t_2)$, which is calculated by the below equations.*

[10]It is originally from Example 5.1.15.

$$r(R,\ G) = (t_1,\ t_2),\ where$$

$$t_1 = \begin{cases} 1,\ q = g \\ 0,\ q \neq g \end{cases}$$

$$t_2 = |\mathbb{S}|,\ \ where\ \mathbb{S} = \{p_i(\vec{X_i})|\ \forall 1 \leq i \leq m,\ p_i(\vec{X_i}) \sim g(\vec{c})\}$$

In Example 5.2.2, $r(R1,\ white(lily)) = (1,2)$ where $t_1 = 1$ because the head predicate is the same as the goal proposition, and $t_2 = 2$ because the two preconditions of A1 can be resolved by A2 and A4 respectively. On the other hand, $r(A1,\ black(bruce)) = (0,1)$ because the head predicate does not match with the target predicate and only the precondition $swan(X)$ can be resolved by A3 which has an overlap constant $bruce$ with the target $black(bruce)$.

A rule's relevancy is higher than another's if its 2-tuple is lexicographically bigger than the other's, e.g., $(1,0) > (0,3)$ and $(1,2) > (1,1)$.

Theorem 5.2.1 (The Transitivity of the Relevancy Degree)**.** *The relevancy degree of rules is transitive, written as the following equation where $r(R1,G)$, $r(R2,G)$, $r(R3,G)$ be the relevancy degrees of rules $R1, R2$ and $R3$ w.r.t. goal G respectively.*

$$r(R1,G) \geq r(R2,G) \wedge r(R2,G) \geq r(R3,G) \implies r(R1,G) \geq r(R3,G) \tag{5.30}$$

Based on the transitivity of $\geq$, it can be seen that Theorem 5.2.1 is true.

Specification 3 (The selection method)**.** *The selection method γ takes the proofs of efficiencies P and the ground goal proposition G as the inputs, and outputs the rules which are involved in P with the highest relevancy w.r.t. G.*

$$\gamma(P,G) = \{R|\forall R' \in P,\ r(R',\ G) \leq r(R,\ G)\} \tag{5.31}$$

In Example 5.2.2, $\gamma(\mathbb{T}, black(bruce)) = \{R1\}$, because $r(R1,\ black(bruce)) = (0,1) > r(R2,\ black(bruce)) = (0,0)$. Therefore, $R1$ is the source rule by analogising which target rule is formalised.

1. The search method.

The search method is the key of an analogical abduction which formalises the target rule that explains the targeted ground proposition. To cover all possible cases, the method below has to do multiple tasks. The swan theory will continue to be the example discussed along with the definition of the method, where some steps are not involved in its repairing process. A more complicated example is the game theory which is given in the next section, where the search for the target rule is described step by step following the below definition.

Specification 4 (The search method)**.** *The search method ζ takes the input theory $\mathbb{T}$, the selected source rule R and the targeted ground proposition G as its inputs and formalises a set of target rules $\mathbb{R}$.*

$$\zeta(\mathbb{T},\ R,\ G) = \mathbb{R} \tag{5.32}$$

where $\forall R' \in \mathbb{R}, \mathbb{T} \cup R' \vDash G,\ \mathbb{T} \nvDash G$.

Let the parent R be $p_1(\vec{X_1}) \wedge p_2(\vec{X_2}) \wedge \ldots \wedge p_m(\vec{X_m}) \implies q(\vec{Y})$ and G be $g(\vec{c})$. The procedure of ζ consists of four steps.

Step 1. Duplicate R and reform the duplicate's head based on the equations below. According to the relation between the goal and R, there are four possible cases. The resulting rule is R_2. Notice that the source rule R is not changed. In $\vec{Y'}$, the mismatched arguments are replaced by independent variables.

$$R_2 = \begin{cases} p_1(\vec{X_1}) \wedge p_2(\vec{X_2}) \wedge \ldots \wedge p_m(\vec{X_m}) \implies g(\vec{Y'}), & \textit{iff } q \neq g \ \wedge\ \nexists \mu,\ \vec{Y} \cdot \mu \equiv \vec{c} \\ p_1(\vec{X_1}) \wedge p_2(\vec{X_2}) \wedge \ldots \wedge p_m(\vec{X_m}) \implies g(\vec{Y}), & \textit{iff } q \neq g \ \wedge\ \exists \mu,\ \vec{Y} \cdot \mu \equiv \vec{c} \\ p_1(\vec{X_1}) \wedge p_2(\vec{X_2}) \wedge \ldots \wedge p_m(\vec{X_m}) \implies g(\vec{Y'}), & \textit{iff } q = g \ \wedge\ \nexists \mu,\ \vec{Y} \cdot \mu \equiv \vec{c} \\ R, & \textit{iff } q = g \wedge\ \exists \mu,\ \vec{Y} \cdot \mu \equiv \vec{c} \end{cases}$$

Step 2. Delete all irrelevant preconditions ($\mathbb{I}$) from R2, which is represented by function $\dot{-}$, and then get the instantiated rule R_3 based on the target $g(\vec{c})$, where μ is from the last step, and $k \leq m$.

$$R_3 = (R_2 \dot{-} \mathbb{I}) \cdot \mu = p_1(\vec{X_1}) \wedge p_2(\vec{X_2}) \wedge \ldots \wedge p_k(\vec{X_k}) \implies g(\vec{Y}) \tag{5.33}$$

$$\mathbb{I} = \{p_i(\vec{X_i}) |\ \forall i, 1 \leq i \leq m,\ p_i(\vec{X_i}) \not\rightsquigarrow g(\vec{c})\} \tag{5.34}$$

Step 3. Get the set of irresolvable preconditions $\mathbb{S}(R_3)$ based on Equation 5.36, where ν is a substitution.

$$g(\vec{Y}) \cdot \gamma \equiv g(\vec{c}) \tag{5.35}$$

$$\mathbb{S}(R_3) = \{p_i(\vec{V_i}) | \forall 1 \leq i \leq k,\ \forall \mathbb{T} \vdash p_i(\vec{c}), \nexists \nu, p_i(\vec{c}) \equiv p_i(\vec{V_i}) \cdot \nu\} \tag{5.36}$$

The inference is as below with irresolvable preconditions left in the final goal clause, where $n \leq k$:

$$\cfrac{\cfrac{g(\vec{c}) \implies \qquad p_1(\vec{X_1}) \wedge p_2(\vec{X_2}) \wedge \ldots \wedge p_k(\vec{X_k}) \implies g(\vec{Y})}{(p_1(\vec{X_1}) \wedge p_2(\vec{X_2}) \wedge \ldots \wedge p_k(\vec{X_k})) \cdot \gamma \implies \qquad \mathbb{T}}}{p_1(\vec{V_1}) \wedge p_2(\vec{V_2}) \wedge \ldots \wedge p_n(\vec{V_n}) \implies} \tag{5.37}$$

Step 4. *Get the minimal argument mismatches $\mathbb{M}_m(R)$ between irresolvable preconditions and their corresponding theorems: 1) Equation 5.38 calculates the argument mismatches $\mathbb{M}(p(\vec{c}))$ between the instantiated precondition $p(\vec{V_i})$ and the theorem $p(\vec{c})$; 2) Equation 5.39 gets the minimal mismatch set; 3) Equation 5.40 collects the set of all minimal mismatches for all preconditions in R_3.*

$$\forall \mathbb{T} \vdash p(\vec{c}),\ \mathbb{M}(p(\vec{c})) = \{(V_j,\ c_j) | \forall c_j \in \vec{c}, p(\vec{V}) \in \mathbb{S}(R), V_j \in \vec{V}, c_j \neq V_j\} \tag{5.38}$$

$$\mathbb{M}_{min}(p(\vec{V})) = \mathbb{M}(p(\vec{c_n})),\ \forall p(\vec{c_l}) \neq p(\vec{c_n}),\ |\mathbb{M}(p(\vec{c_n}))| \leq |\mathbb{M}(p(\vec{c_l}))| \tag{5.39}$$

$$\mathbb{M}_m(R_3) = \{(v,c) | \forall p_i(\vec{V_i})) \in \mathbb{S}(R_3), (v,c) \in \mathbb{M}_{min}(p_i(\vec{V_i})))\} \tag{5.40}$$

Step 5. *Get $R4$ by replacing the mismatched argument in precondition $p_i(\vec{V_i})$ with its paired argument based on $\mathbb{M}_{min}(p(\vec{V}))$, where $\mathcal{R}(p_i(\vec{V_i}), p_i(\vec{C_i}), R_3)$ replaces $p_i(\vec{V_i})$ with $p_i(\vec{C_i})$ in R_3.*

$$R_4 = \mathcal{R}(p_i(\vec{V_i}), p_i(\vec{C_i}), R_3), \forall 1 \leq i \leq n \tag{5.41}$$

$$C_{ij} = \begin{cases} c,\ (V_{ij}, c) \in \mathbb{M}_m(R_3) \\ V_{ij},\ otherwise \end{cases} \tag{5.42}$$

where $\forall 1 \leq j \leq |\vec{V_i}|$, $V_{ij} \in \vec{V_i}$, $C_{ij} \in \vec{C_i}$.

The remaining goal clause in 5.37 can be resolved as the following, where the original irresolvable goal clause become resolvable after replacing their mismatched arguments.

$$\cfrac{\cfrac{p_1(\vec{V_1}) \wedge p_2(\vec{V_2}) \wedge \ldots \wedge p_n(\vec{V_n}) \implies \qquad \mathcal{R}(p_i(\vec{V_i}), p_i(\vec{C_i}), R_3)}{p_1(\vec{C_1}) \wedge p_2(\vec{C_2}) \wedge \ldots \wedge p_n(\vec{C_n}) \implies \qquad \mathbb{T}}}{\implies} \tag{5.43}$$

Step 6. *Get $R5$ by adding the introduction preconditions in $\mathbb{I}_{min}$ to R_4 to link the mismatching arguments to their partners, which bound new arguments to certain* *theorems. Introduction preconditions are considered to represent the semantic relations between mismatched pairs.* $\mathbb{I}_{min}$ *is the minimal subset of* $\mathbb{I}$ *which satisfies the condition 5.45.*

$$\mathbb{I} = \{q(\vec{u}) | \exists (c_1, c_2) \in \mathbb{M}_m(R_3), \mathbb{T} \vdash q(\vec{u}), c_1 \in \vec{u}, c_2 \in \vec{u}\} \tag{5.44}$$

$$\mathbb{I}_{min} \subseteq \mathbb{I},\ \forall (c_1, c_2) \in \mathbb{M}_c(R), \exists q(\vec{u}) \in \mathbb{I}_{min}, c_1 \in \vec{u}, c_2 \in \vec{u} \tag{5.45}$$

Analogise R1 to Prove $black(bruce)$ ***in Swan Theory (Example 5.2.2).***

$g(\vec{c}) = black(bruce)$ *and* $R1$ *is:* $european(X) \wedge swan(X) \implies white(X)$. *The resulting rules of each search step is given below.*

Step 1. *R2: reform the head:* $european(X) \wedge swan(X) \implies black(X)$

Step 2. *R3: delete the irrelevant precondition* $european(X)$ *from R2:*
$swan(bruce) \implies black(bruce)$ *(R3)*

Step 3. *- Step 6. No irresolvable precondition in* $R3$ *so no change to make in these step:* $R5 = R4 = R3$.

Step7a. *Generalisation:* R_5*: because* $\mathcal{O}(bruce, R_5) = 2 > 1$*, Therefore,* $R_c = \mathcal{R}(bruce,\ IVbruce,\ R_5)$*, where IVbruce is the independent variable arising from the constant bruce:*
$swan(IVbruce) \implies black(IVbruce)$ *(R_c)*

Step7b. *Faults caused: incompatibilities of* $black(lily)$ *and* $black(lucy)$.

Step7c. *Adjustment: the only remaining relevant theorem of* $black(bruce)$ *is* $australian(bruce)$*. Add it to* R_5*:*
$australian(bruce) \wedge swan(bruce) \implies black(bruce)$

Finish. *Repeat Step 7 on the above rule and terminate with the targeted rule below.*
$australian(IVbruce) \wedge swan(IVbruce) \implies black(IVbruce)$ *(TR)*

Step 7. *Generalise R5 and check if it causes incompatibilities. If yes, apply adjustment. Otherwise, terminate.*

(a) **Generalisation:** *If a constant c occurs more than once in the resulting rule, replace all the occurrences of that constant with an independent variable*

IVc arising from the constant c. In Equation 5.46, function $\mathcal{O}(c, R_5)$ returns the number of the occurrences of constant c in rule R_5.

$$R_c = R_5 \cdot \gamma \tag{5.46}$$

$$\gamma = \{IVc/c \mid \forall c,\ \mathcal{O}(c, R_5) > 1\} \tag{5.47}$$

(b) **Fault Check:** *If R_c causes an incompatibility computed by Equation 5.48, continue to the step of Adjustment. If no incompatibility is caused, terminate with R_c being the target rule abduced.*

$$\exists \alpha \in \mathcal{F}(\mathbb{PS}),\ \mathbb{T} \nvDash \alpha,\ \mathbb{T} \cup R_c \vDash \alpha \tag{5.48}$$

(c) **Adjustment:** *Add an adjustment precondition $P(\vec{C_p})$ to rule R_5, where the adjustment precondition has to be a relevant theorem w.r.t. the goal proposition, and does not cause duplicates. Exhaustively get a set of all possible adjusted rules, and generalise each of them, which results in a set of candidates of the target rules $\mathbb{S}_c$. Then the set of the abduced target rules $\mathbb{R}_C$ is one from $\mathbb{S}_c$ of which each target rule does not cause incompatibilities.*

$$\mathbb{R}_C = \{R_c \mid \forall \alpha \in \mathcal{F}(\mathbb{PS}),\ R_c \in \mathbb{S}_c, \mathbb{T} \cup R_c \nvDash \alpha\}$$

Example 5.2.2 The Repaired Swan Theory.

$$european(X) \wedge swan(X) \Longrightarrow white(X) \tag{R1}$$

$$australian(IVbruce) \wedge swan(IVbruce) \Longrightarrow black(IVbruce) \tag{TR}$$

$$german(X) \Longrightarrow european(X) \tag{R2}$$

$$\Longrightarrow swan(lily) \tag{A1}$$

$$\Longrightarrow swan(lucy) \tag{A2}$$

$$\Longrightarrow swan(bruce) \tag{A3}$$

$$\Longrightarrow german(lily) \tag{A4}$$

$$\Longrightarrow european(lucy) \tag{A5}$$

$$\Longrightarrow australian(bruce) \tag{A6}$$

$$\mathcal{T}(\mathbb{PS}) = \{black(bruce), white(lily), white(lucy)\}$$

$$\mathcal{F}(\mathbb{PS}) = \{white(bruce), black(lily), black(lucy)\}$$

It is possible that the target rule is not unique, or does not exist. The former can be evaluated by the sub-optimal repair pruning mechanism described in the next chapter, and the repair plan of analogising an existing rule will not be applied in the latter case. By applying the analogical abduction, TR is formalised and added to the swan theory in Example 5.2.2, which fixes the insufficiency w.r.t. $black(bruce)$. In commonsense reasoning we might appeal to the fact that Australian and European both refer to continents. If there is such background knowledge available, the search space of the precondition candidates can be reduced to the relevant domain, which can be done in future work, discussed as point 7 in §9.1.

The underlying principles of the above searching function are:

1. The preconditions in the target rule should be relevant to the entity given in the goal proposition;
2. Retain as much source rule as possible in the target rule;
3. Do not add new preconditions unless there is a reason for it.

Therefore, preconditions are deleted if they are not relevant to the entity of the goal proposition and new preconditions are added only when incompatibility is caused.

3. The evaluation function.
The last step of the search method includes the fault check and the corresponding adjustment, which can be seen as an evaluation of the correctness of the target rule. On the other hand, all the preconditions that are newly added to the rule are from theorems of the theory so they can be resolved by reversing the substitutions in the generalisation step.

Theorem 5.2.2. *The target rule formalised in the search method in Specification 4 proves the goal.*

Proof. Let source rule $R1$ be $p_1(\vec{X_1}) \wedge p_2(\vec{X_2}) \wedge ... \wedge p_m(\vec{X_m}) \implies q(\vec{Y})$ and goal: $g(\vec{c}) \implies$.
Let target rule TR be $q_1(\vec{X_1}) \wedge q_2(\vec{X_2}) \wedge ...q_n(\vec{X_n}) \wedge p_1(\vec{X_1}) \wedge ...p_k(\vec{X_k}) \implies g(\vec{X_g})$, where $q_1(\vec{X_1}) \wedge q_2(\vec{X_2}) \wedge ...q_n(\vec{X_n})$ are the newly added preconditions, and $p_1(\vec{X_1}) \wedge ...p_k(\vec{X_k})$ are the remaining preconditions from $R1$.
Resolve the goal clause with TR: $g(\vec{c}) \equiv g(\vec{X_g}) \cdot \mu$.
Based on Step 7a: $g(\vec{c}) \cdot \gamma \equiv g(\vec{X_g})$; Therefore, $\forall c/X \in \mu$, $\exists X/c \in \gamma$.

Therefore, there exist a substitution θ:

$$\exists\theta,\ \vec{X'}\cdot\mu\cdot\theta = \vec{C'},\ \text{ where } \vec{C'}\cdot\gamma \equiv \vec{X'} \tag{5.49}$$

In 5.51:
$q_1(\vec{V_1}) \wedge ... \wedge q_n(\vec{V_n}) \wedge p_1(\vec{V_1}) \wedge ... \wedge p_k(\vec{V_k}) \equiv (q_1(\vec{X_1}) \wedge ... \wedge q_n(\vec{X_n}) \wedge p_1(\vec{X_1}) \wedge ... \wedge p_k(\vec{X_k}))\cdot\mu$.
Based on Step 7a:
$q_1(\vec{X_1}) \wedge ... \wedge q_n(\vec{X_n}) \wedge p_1(\vec{X_1}) \wedge ... \wedge p_k(\vec{X_k})) \equiv (q_1(\vec{C_1}) \wedge ... \wedge q_n(\vec{C_n}) \wedge p_1(\vec{C_1}) \wedge ... \wedge p_k(\vec{C_k}))\cdot\gamma$,
Then substitution θ in 5.49 can be applied to 5.51:

$$q_1(\vec{C_1}) \wedge ...q_n(\vec{C_n}) \wedge p_1(\vec{C_1}) \wedge ...p_k(\vec{C_k}) \equiv p_1(\vec{V_1}) \wedge p_2(\vec{V_2}) \wedge ... \wedge p_m(\vec{V_m})\cdot\theta \tag{5.50}$$

The resulting goal clause $q_1(\vec{C_1}) \wedge ...q_n(\vec{C_n}) \wedge p_1(\vec{C_1}) \wedge ...p_k(\vec{C_k})$ is the R_5 in Step 6, where each subgoal is a theorem in the theory, because the mismatches variables have been rewritten according to theorems in step 5 and only theorems are added as preconditions in step 6. Therefore, the goal clause can be resolved with the input theory.

$$\cfrac{\cfrac{\cfrac{g(\vec{c}) \Longrightarrow}{q_1(\vec{V_1}) \wedge ...q_n(\vec{V_n}) \wedge p_1(\vec{V_1}) \wedge ...p_k(\vec{V_k}) \Longrightarrow}\ TR}{q_1(\vec{C_1}) \wedge ...q_n(\vec{C_n}) \wedge p_1(\vec{C_1}) \wedge ... \wedge p_m(\vec{C_m}) \Longrightarrow}\ \theta}{\Longrightarrow}\ \mathbb{T} \tag{5.51}$$

□

In this section, we defined analogical abduction by defining the selection method, the search method and the evaluation of the formalised target rule. As a result, by analogising an existing useful rule, a target rule can be formalised. Adding the target rule to the input theory unblocks the proof of the goal, which was unprovable previously.

5.2.2 Repair Game Theory by Analogical Abduction

In this section, the example of Analogically abducing target rules is given to illustrate the strategy of selecting the source rule and searching for the target rule.

In the virtual bargaining game (Bundy et al., 2020), there are two players: the sender and the receiver, and three boxes of two kinds: the helpful and the harmful.

The sender knows all the boxes' kinds and marks one of them to guide the receiver to choose a helpful box. On the other hand, the receiver only knows which box is marked, and then aims to select as many helpful boxes as possible. Two rounds of the game are taken as our example. In the first round (g1), only box 1 (b1) is helpful and boxes b2 and b3 are harmful while the opposite is true in the second round (g2): b2 and b3 are helpful and b1 is harmful.

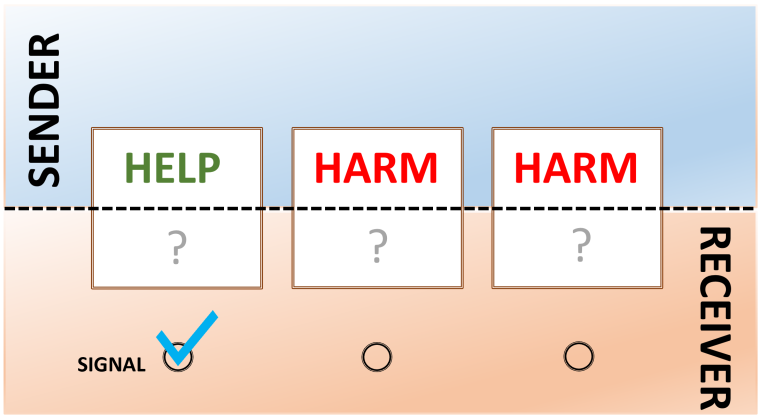


Figure 5.6: The first round of the game $(g1)$: *$b1$ is the first box; the sender has marked the only helpful box $b1$; the players win if the receiver selects $b1$.*

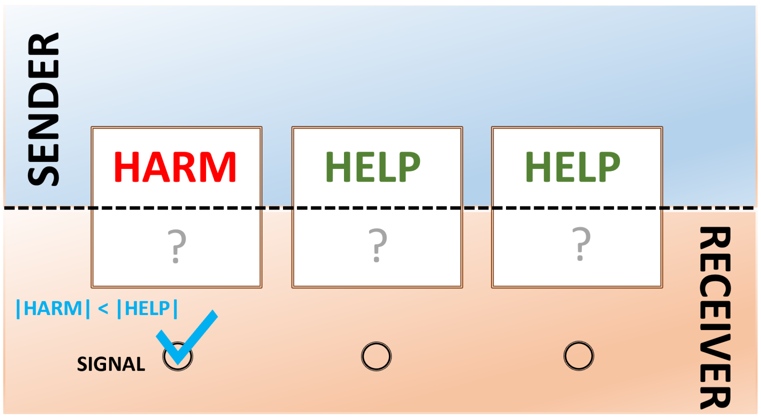


Figure 5.7: The second round of the game $(g2)$: *the sender has marked the harmful box $b1$; the players lose if the receiver selects $b1$.*

The setup and the marks of both games are formalised in the theory in Example 5.2.3, where g1's winning strategy is represented by rule A1. Assertion $<(g1, hp, hm)$ means that in game1, there are more harmful boxes than the helpful ones, while in game2 the opposite is true (A3).

$$\dfrac{\dfrac{select(g1,b1) \Longrightarrow \qquad mark(X,\,Y) \Longrightarrow select(X,\,Y)}{mark(g1,b1) \Longrightarrow \qquad \Longrightarrow mark(g1,b1)}}{\Longrightarrow}$$

Figure 5.8: The proof of $select(g1,b1)$ in Example 5.2.3.

$$\dfrac{\dfrac{select(g2,b2) \Longrightarrow \qquad mark(X,\,Y) \Longrightarrow select(X,\,Y)}{mark(g2,b2) \Longrightarrow \qquad \Longrightarrow mark(g2,b1)}}{\Longrightarrow}$$

Figure 5.9: The broken proof of $select(g2,b2)$ in Example 5.2.3: *the RS in blue is broken.*

Example 5.2.3. *Virtual Bargaining Game Theory.*

$$\begin{aligned}
mark(X,\,Y) &\Longrightarrow select(X,Y) && \text{(A1)}\\
&\Longrightarrow <(g1,hp,hm) && \text{(A2)}\\
&\Longrightarrow <(g2,hm,hp) && \text{(A3)}\\
&\Longrightarrow mark(g1,b1) && \text{(A4)}\\
&\Longrightarrow mark(g2,b1) && \text{(A5)}\\
\Longrightarrow b1 \neq b2 \qquad \Longrightarrow b1 \neq b3 \qquad &\Longrightarrow b2 \neq b3 && \text{(A6-8)}
\end{aligned}$$

$\mathcal{T}(\mathbb{PS}) = \{select(g1,b1),\ select(g2,b2),\ select(g2,b3)\}$
$\mathcal{F}(\mathbb{PS}) = \{select(g1,b2), select(g1,b3),\ select(g2,b1)\}$

$\mathbb{PS}$ depicts the result of each possible selection, where the choices in $\mathcal{T}(\mathbb{PS})$ win the game while the ones in $\mathcal{F}(\mathbb{PS})$ fail. Now the task is to automatically repair the theory to win both g1 and g2, which corresponds to repairing the insufficiencies caused by $select(g2,\ b2)$ and $select(g2,\ b3)$ and the incompatibility caused by $select(g2,\ b1)$.

The insufficiency is repaired first by the analogical abduction. Let the goal proposition be $select(g2,b2)$. Although there is only one rule in the theory, it still needs to be checked whether it is qualified as a source rule. It can be concluded that rule $A1$ is useful, because $select(g1,b1)$ is from $\mathcal{T}(\mathbb{PS})$ and rule $A1$ contributes to proving $select(g1,b1)$, of which the proof is shown in Figure (5.8). Meanwhile, the relevancy degree of $A1$ w.r.t. the target $select(g2,b2)$ is (1, 2) according to Definition 5.2.1. Therefore, $R1$ is the source rule for the rest abduction process.

Similar to the last example, the search of the target rule is given following the steps in Definition 4.

Step 1. Duplicate rule A1. As the head predicate is the same as the target $select(g2,b2)$, no further change is needed in this step. The duplicated rule is $R2$.

$$mark(X,Y) \implies select(X,Y) \tag{R2}$$

Step 2. Because no irrelevant precondition w.r.t. the targeted $select(g2,b2)$ occurs in $R2$, no deletion is made in this step. Get the instantiated rule $R3$ based on the target $select(g2,b2)$.

$$mark(g2,\ b2) \implies select(g2,\ b2) \tag{R3}$$

Step 3. The precondition $mark(g2,\ b2)$ is irresolvable due to the mismatch of $b2$ and $b1$.

$$\mathbb{S}(R_3) = \{mark(g2,\ b2)\} \tag{5.52}$$

Step 4. The minimal argument mismatches are selected based on two theorems of $mark/2$: $mark(g1,b1)$ and $mark(g2,b1)$, where the latter is the minimal one.

$$\mathbb{M}_{min}(mark(g2,\ b2)) = \mathbb{M}(mark(g2,b1)) = \{(b2,b1)\} \tag{5.53}$$

Step 5. Get $R4$ by replacing the mismatched argument $b2$ with $b1$ according to $\mathbb{M}_{min}$. Here the replacement is $\mathcal{R}(mark(g2,\ b2), mark(g2,\ b1), R3)$. The change made in this step is highlighted in red.

$$mark(g2,\ b1) \implies select(g2,\ b2) \tag{R4}$$

Step 6. Add introduction preconditions. The set of all candidates of the introduction preconditions is $\mathbb{I}$, where the argument of each candidate contains the pair of the mismatches: $b1$ and $b2$. As there is only one candidate $b1 \neq b2$, it is added to the rule.

$$\mathbb{I}_{min} = \mathbb{I} = \{b1 \neq b2\} \tag{5.54}$$

$$b1 \neq b2 \wedge mark(g2,\ b1) \implies select(g2,\ b2) \tag{R5}$$

Step 7. Generalisation, check and adjustment.

(a) Generalisation: replace $b1$ with an independent variable $IVb1$ because $b1$ occurs more than once in $R5$. The generalised rule is $R6$.

$$IVb1 \neq IVb2 \wedge mark(IVg2, IVb1) \implies select(IVg2, IVb2)$$

The rule can be simplified by renaming variables.

$$X \neq Y \wedge mark(Z, X) \implies select(Z, Y) \tag{R6}$$

(b) Incompatibility check: fails due to the theorems of $select(g1, b2)$ and $select(g1, b3)$.

(c) Adjustment: add an adjust precondition to $R6$. The relevant preconditions w.r.t. $select(g2, b2)$ that will not cause duplicates include $b1 \neq b2$, $b2 \neq b3$ and $<(g2, hm, hp)$. Adding them to $R6$ resultes in the following rules respectively.

$$\begin{aligned} b1 \neq b3 \wedge b1 \neq b2 \wedge mark(g2, b1) &\implies select(g2, b2) \\ b2 \neq b3 \wedge b1 \neq b2 \wedge mark(g2, b1) &\implies select(g2, b2) \\ <(g2, hm, hp) \wedge b1 \neq b2 \wedge mark(g2, b1) &\implies select(g2, b2) \end{aligned}$$

Generalise the adjusted rules. The following candidates of the target rules are generated.

$$\begin{aligned} X \neq b3 \wedge X \neq Y \wedge mark(Z, X) &\implies select(Z, Y) \\ Y \neq b3 \wedge X \neq Y \wedge mark(Z, X) &\implies select(Z, Y) \\ <(Z, hm, hp) \wedge X \neq Y \wedge mark(Z, X) &\implies select(Z, Y) \end{aligned}$$

Only the last candidate does not cause any incompatibility so the search terminates with the following rule abduced as the target rule.

$$<(Z, hm, hp) \wedge X \neq Y \wedge mark(Z, X) \implies select(Z, Y) \tag{TR}$$

The revised theory is given in Example 5.2.4, where TR is the target rule. However, the theory is still incompatible because $select(g2, b1)$ is derivable due to $A1$.

Example 5.2.4. *Semi-Repaired Virtual Bargaining Game Theory.*

$$
\begin{aligned}
mark(X,\,Y) &\implies select(X,\,Y) && \text{(A1)}\\
<(Z,\,hm,\,hp) \wedge \neq(X,\,Y) \wedge mark(Z,\,X) &\implies select(Z,\,Y) && \text{(TR)}\\
&\implies <(g1,hp,hm) && \text{(A2)}\\
&\implies <(g2,hm,hp) && \text{(A3)}\\
&\implies mark(g1,b1) && \text{(A4)}\\
&\implies mark(g2,b1) && \text{(A5)}\\
\implies b1 \neq b2 \qquad \implies b1 \neq b3 \qquad &\implies b2 \neq b3 && \text{(A6-8)}
\end{aligned}
$$

$\mathcal{T}(\mathbb{PS}) = \{select(g1,b1),\ select(g1,b3),\ select(g2,b2)\}$
$\mathcal{F}(\mathbb{PS}) = \{select(g1,b2), select(g2,b1),\ select(g2,b3)\}$

By applying CR6 (on page 78), the variant belief revision adds an unprovable precondition to A1 to block the unwanted proofs. The repaired $A1$ is as the following.

$$<(X,\,hp,\,hm) \wedge mark(X,\,Y) \implies select(X,\,Y)$$

Now the repaired theory is fault-free, which is written in Example 5.2.5.

Example 5.2.5. *Repaired Virtual Bargaining Game Theory.*

$$
\begin{aligned}
<(X,\,hp,\,hm) \wedge mark(X,\,Y) &\implies select(X,\,Y) && \text{(A1')}\\
<(Z,\,hm,\,hp) \wedge \neq(X,\,Y) \wedge mark(Z,\,X) &\implies select(Z,\,Y) && \text{(TR)}\\
&\implies <(g1,hp,hm) && \text{(A2)}\\
&\implies <(g2,hm,hp) && \text{(A3)}\\
&\implies mark(g1,b1) && \text{(A4)}\\
&\implies mark(g2,b1) && \text{(A5)}\\
\implies b1 \neq b2 \qquad \implies b1 \neq b3 \qquad &\implies b2 \neq b3 && \text{(A6-8)}
\end{aligned}
$$

$\mathcal{T}(\mathbb{PS}) = \{select(g1,b1),\ select(g1,b3),\ select(g2,b2)\}$
$\mathcal{F}(\mathbb{PS}) = \{select(g1,b2),\ select(g2,b1),\ select(g2,b3)\}$

All of the repair algorithms have been introduced in §5.1 and §5.2. In many cases, only one repair may not be enough to achieve a fault-free theory. The recursion of the repair algorithm is discussed in the next section.

5.3 Combination of Techniques

The three candidate techniques are complementary: abduction and belief revision operate on axioms in opposite directions, adding and deleting; while reformation works by changing the signature. By combining them, the repair mechanism can fix faults with new repairs. In some scenarios, a combined repair is necessary to fix the faulty theory and deliver the accurate meaning, e.g., in Example 5.2.5, abduction and belief revision are combined to repair the game theory and formalise the winning strategy accurately.

It is fairly common that several faults appear in one theory. These faults are detected in turn: after repairing one fault, a new one is detected. By repairing each fault with all possible techniques exhaustively, different techniques can be combined in the final fault-free theories. Except for the original existing faults, there could be two kinds of introduced faults during a repair process:

1. New introduced faults. A repair may introduce an error which did not exist previously. For example, if a predicate in an unwanted proof is changed, and this predicate is also necessary for proving a preferred proposition, then this repair causes newly introduced insufficiency.

2. Recurring faults. 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, a heuristic is developed for avoiding looping, which is introduced in Section 6.5.

For each fault, we can get all possible repairs by applying the three candidate techniques in parallel. And then continue repairing the resulting theories until no fault remains or no repairs are available. In other words, all solutions are searched in parallel, and the ones with the fewest repairs are found first.

Each of the final repaired theories is the product of the combinations of the sequent repairs on its search branch, shown in Figure 5.10. For example, if reformation changes the signature of the theory for fixing fault $F1$, after which belief revision deletes an axiom to tackle a remaining fault $F2$, then reformation and belief revision are combined in this search branch.

The search branches of fault-free theories can be of different length. When a fault cannot be repaired, the search branch terminates with failure. The final set of repaired theories is output without redundancies.

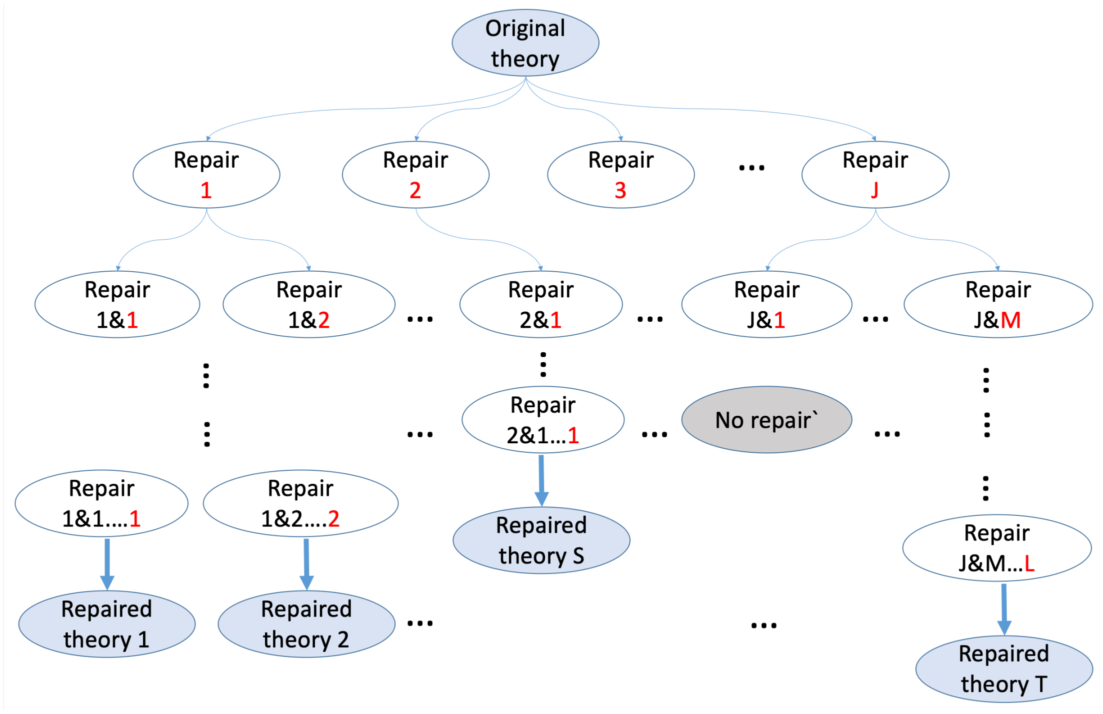


Figure 5.10: The original search space for fault-free theories: *the length of each search branch can be different; a search branch terminates with failure if there is no repair available to fix a detected fault, whose end is highlighted in grey.*

The game theory in the last section is repaired by combining analogical abduction and the variant of belief revision. Recall the faulty swan theory in Example 4.2.1, whose desired repair is given in Example 5.3.1, where reformation and abduction which adds assertions are combined.

The Faulty Swan Theory.

$$german(X) \implies european(X) \qquad (A1)$$

$$european(X) \wedge swan(X) \implies white(X) \qquad (A2)$$

$$\implies german(bruce) \qquad (A3)$$

$$\implies swan(bruce) \qquad (A4)$$

$$\mathcal{T}(\mathbb{PS}) = \{black(bruce)\}, \mathcal{F}(\mathbb{PS}) = \{white(bruce)\}$$

The problem could be caused by the fact that the concept of ‘European swan’ is ambiguous: it may mean that ‘the European variety of swans’ or ‘swans resident in Europe’. In the scenario that all swans of European variety are white, but Bruce is resident in Europe, the target theory is as below, where the desired repairs are highlighted in red.

Example 5.3.1. *The Desired Repaired Theory.*

$$
\begin{aligned}
german(X,Y) &\implies european(X,Y) && \text{(TA1)}\\
european(X,variety) \wedge swan(X) &\implies white(X) && \text{(TA2)}\\
&\implies german(bruce,resident) && \text{(TA3)}\\
&\implies swan(bruce) && \text{(TA4)}\\
&\implies black(bruce) && \text{(TA5)}
\end{aligned}
$$

In the rest of this section, we are going to discuss how to repair the swan theory based on the combination repair mechanism, and whether it can generate the targeted theory as we proposed. The order of solving incompatibility and insufficiency is unimportant. Firstly, incompatibility is repaired in multiple ways. The three repairs below are examples.

```
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, dummy2)  ∧ swan(X)   ⟹  white(X).
```

By deleting A4, the first repair blocks RS4, which is in blue in Figure 4.3. This repair is proper in the scenario that Bruce is another kind of a bird rather than a swan. The second repair blocks RS2, which is another way to distinguish resident from variety, where *europedash* means resident in Europe. The last one is quite close to the desired repairs, which applies concept revision by arity increment. As this repair adds a new argument to predicate *european*, it has to be propagated to all instances of the predicate *european*/1 following the propagation rules. Here *European* in A2 is seen as the targeted literal, which is assigned a unique constant *dummy2*, while the other instances are assigned the default constant *dummy1* or an independent free variable for rules. After propagation, the partially repaired theory is:

Example 5.3.2. *The Semi-Repaired Theory.*

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

$$
\begin{aligned}
german(X,Y) &\implies european(X,Y) && \text{(A1')}\\
european(X,dummy2) \wedge swan(X) &\implies white(X) && \text{(A2')}\\
&\implies german(bruce,dummy1) && \text{(A3)}\\
&\implies swan(bruce) && \text{(A4)}
\end{aligned}
$$

In this concept revision, A2 is the targeted axiom and all the others are passive axioms. According to Postulate 2, the default constant *dummy*1 and free variable Y are added as the new argument accordingly. Here the arity increment of *german* is because that Datalog requires that each variable in the head of rule $A1$ must also exist in its body.

$$\begin{array}{cl}
white(bruce) \implies & \\ \cline{1-1}
european(bruce,\ dummy2) \wedge swan(bruce) \implies & european(X,\ dummy2) \wedge swan(X) \implies white(X) \\ \cline{1-1}
german(bruce,\ dummy2) \wedge swan(bruce) \implies & german(X,\ Y) \implies european(X,\ Y) \\ \cline{1-1}
german(bruce, dummy2) \implies & \implies swan(bruce)
\end{array}$$

Figure 5.11: Evidence of the blocked incompatibility of the Swan theory in Example 5.3.2 by reformation which increases the arity of predicate *european* and *german*.

The incompatibility is blocked in the current theory. The evidence of the blocking is given by the Figure 5.11. It can be seen that the sub-goal *german(bruce, dummy2)* cannot unify the input literal *german(bruce, dummy1)*, neither other input literals. At this point, the incompatibility has been solved. We will continue to repair insufficiency. The best repair in our scenario is adding the preferred proposition as an axiom directly.

Example 5.3.3. *The Repaired Theory.*

$$german(X,Y) \implies european(X,Y) \qquad (A1')$$
$$european(X,dummy2) \wedge swan(X) \implies white(X) \qquad (A2')$$
$$\implies german(bruce,dummy1) \qquad (A3)$$
$$\implies swan(bruce) \qquad (A4)$$
$$\implies black(bruce) \qquad (A5)$$

The current theory is faithful concerning $\mathbb{PS}$. The repair solution is generated by combining reformation and abduction, and the solution satisfies the claimed repair postulates. By interpreting *dummy2* as variety and *dummy1* as resident, the produced theory is the desired one given at the beginning of this section. The semantic interpretation for dummy terms is something worthwhile to research in the future, discussed as point 4 in §9.1.

It can be seen that a single technique is not enough for generating the best repairs for a theory in some scenarios. Therefore, the combination repair mechanism works better than the individual techniques it combines.

5.4 Summary

Different repair techniques could be combined when dealing with multiple faults in one theory. By applying all possible repairs on each fault, a completely repaired theory is generated by combining a sequent repair techniques.

We have successfully applied our repair mechanism on several examples. From the result, it can be seen that new repair solutions with desired representations can be produced by combining different repair techniques. Our repair mechanism fills the gap of belief change, abduction and reformation, which operates at the levels of both the axioms and the signature of the input theory.

However, the repair algorithm is manifestly insufficient, because the repairs are generated purely from the view of logic. They are able to repair a faulty theory by blocking or unblocking proofs. However, it is not enough to generate a good repair, e.g., a repair makes the theory understandable, accurate, concise and easy to use. The quality of a logical theory is an open question, and the measurement of the quality of repairs is also a challenge. In Chapter 6, various refinements is developed to improve the performance of our repair generation, e.g., given all possible repairs, we hope to find the best ones while reducing the number of repairs.

Chapter 6

ABC Repair System: Refinements

Chapter 4-5 gives the basic framework of the ABC repair system and the algorithm for repairing insufficiency and incompatibility. In this chapter, refinements that improve the performance of the ABC repair system will be discussed. These refinements include simplifying the formalisation of a theory[1] by the unique name assumption with exceptions (UNAE) in §6.1; defining and quantifying the entrenchment of axioms and signature in §6.2; computing the maximal set of commutative repair plans so that commutative repair plans can be applied together in §6.3; pruning all sub-optimal repairs of a theory in §6.4; and developing heuristics to assist repair generation in §6.5. The summary of this chapter will be given in §6.6

6.1 Unique Name Assumption with Exceptions

In Datalog, inequality cannot be represented because negation is not permitted in its grammar. However, in lots of cases, syntactically non-identical constants are unequal and then inequality is an important ingredient of logic.

One way to express constants being unequal in Datalog is to employ the unique name assumption (UNA), which states that all distinct constants refer to different individuals (Poole and Mackworth, 2010). However, UNA is too strict in many applications, because it is fairly common that there exists one individual with two or more names.

Therefore, this section will develop a mechanism of unique name assumption with exceptions, which is based on UNA, but allows distinct constants to be equal. Symbol

[1]Recall that all predicate symbols and constants start with lowercase letters, and variables start with capital letters.

'$\neq$' will be used for representing inequality as a separate predicate constant rather than the negation of predicate '='.

First of all, the equality set $\mathbb{ES}$ is defined. Notice, that in a theory $\mathbb{T}$, there is only one equality set. We will use c_1 and c_2 to represent syntactically distinct constants.

Definition 6.1.1 (Equality Set ($\mathbb{ES}$)). *The equality set of a theory $\mathbb{T}$ is a set of subsets, where each subset contains syntactically distinct constants that are equal.* $c_1 = c_2$

$$\mathbb{ES} = \{\mathbb{E} | \forall c_1, c_2 \in \mathbb{E},\ \mathbb{T} \vdash c_1 = c_2\} \tag{6.1}$$

It can be seen that $\mathbb{ES}$ is derived based on the equality theorems in a Datalog theory. As in Example 6.1.1, there are five axioms, and '=' occurs in two of them. Based on the logical consequences of that theory, the equality set can be derived: $\mathbb{ES} = \{diana,\ lady\_di,\ camilla\}$. It can be seen that there are incompatibilities because of *camilla* being a member, which will be discussed in the end of this section.

Example 6.1.1. *Motherhood Theory.*

$$\Longrightarrow diana = lady\_di \qquad \Longrightarrow mum(diana, william)$$
$$\Longrightarrow mum(lady\_di, william) \qquad \Longrightarrow mum(camilla, william)$$
$$mum(X,Z) \wedge mum(Y,Z) \Longrightarrow (X = Y)$$
$$\mathcal{T}(\mathbb{PS}) = \varnothing, \qquad \mathcal{F}(\mathbb{PS}) = \{diana = camilla, lady\_di = camilla\}$$

Based on the equality set, *the inequality set includes any equation of inequality between constants not known to be equal according to* $\mathbb{ES}$. This formalisation is defined as UNAE in Definition 6.1.2.

Definition 6.1.2 (Unique Name Assumption with Exceptions (UNAE)). *Let $\mathbb{ES}$ be the equality set of a logical theory, then the UNAE conclusion of that theory is as the following.*

$$\begin{aligned} \exists \mathbb{E} \in \mathbb{ES}, c_1, c_2 \in \mathbb{E} &\iff \mathbb{T} \vdash c_1 = c_2 \\ \forall \mathbb{E} \in \mathbb{ES}, c_1 \notin \mathbb{E} \vee c_2 \notin \mathbb{E} &\iff \mathbb{T} \vdash c_1 \neq c_2 \end{aligned} \tag{6.2}$$

UNAE is a modified UNA by allowing equalities. The traditional UNA can be represented as a special case of UNAE, given by Equation 6.3, where $\mathbb{ES}$ is the equality set in the theory.

$$\forall \mathbb{E} \in \mathbb{ES}, \mathbb{E} = \varnothing \tag{6.3}$$

Based on Equation 6.2, the theory in Example 6.1.1 has the theorems below. Here $X = Y = Z$ is a shorthand for $X = Y \wedge X = Z \wedge Y = Z$.

$$\begin{gathered} diana = lady\_di = camilla \\ diana \neq william,\ lady\_di \neq william,\ camilla \neq william \end{gathered} \tag{6.4}$$

If all the above inequalities are written as axioms, the size of the theory will become much bigger than the current one formalised based on UNAE. Therefore, for a theory with inequalities, its representation can be made more succinct by applying the defined UNAE. Meanwhile, this succinct representation allows the existence of equalities.

Equation 6.4 conflicts with the given $\mathbb{PS}$ in Example 6.1.1 due to the equalities of $camilla$. The theorems derived based on UNAE are a part of the logical consequences of the input theory. Therefore, any fault caused by these theorems will be detected as the corresponding incompatibility/insufficiency of the input theory w.r.t. $\mathbb{PS}$. Following the algorithm given by Chapter 5, the repairs will be generated.

Example 6.1.2. *Repaired Motherhood Theory.*

$$\begin{aligned} &\implies diana = lady\_di \\ &\implies mum(diana, william, dummy1) \\ &\implies mum(lady\_di, william, dummy1) \\ &\implies mum(camilla, william, dummy2) \\ mum(X,Z,dummy1) \wedge mum(Y,Z,dummy1) &\implies (X = Y) \end{aligned}$$

$\mathcal{T}(\mathbb{PS}) = \varnothing,\ \mathcal{F}(\mathbb{PS}) = \{diana = camilla,\ lady\_di = camilla\}$

Example 6.1.2 gives one of the desired repairs of the faulty theory in Example 6.1.1, where the arity of *mum* is increased. By interpreting new argument $dummy1$ as $birth$ and $dummy2$ as $step$, the repaired theory says that $diana$ and $lady\_di$ are two names of $william$'s birth mother, while $camilla$ is $william$'s stepmother.

6.2 Entrenchment Based on a $\mathbb{PS}$

When there are multiple candidates to change for repairing a theory, the least important ones with least informational value should be selected. However, an unaided computer system cannot understand the semantics of theories so that it cannot directly judge which axioms, preconditions, predicates and constants own more overall informational value than the others. For example, an axiom from the paper published in the best journal in its field is usually more entrenched than the one published in an unknown workshop. But the repair system which does not have the relevant publication knowledge will not able to conclude that.

In belief revision, Epistemic Entrenchment (EE) is proposed for prioritising axioms (Gärdenfors, 1988). The more entrenched, the more valuable an axiom is, and the less inclined a system is to change it. However, the automatic evaluation of EE of an axiom is still an open question in the belief revision literature, because it is difficult to find the complete background information and domain knowledge, which can even be a challenge for domain experts sometimes. Although some properties to describe EE have been given by (Gärdenfors, 1988), it is less clear how one can best measure the epistemic entrenchment of an axiom quantitatively, e.g., with scores, by a domain-independent automatic system. More details are discussed in §3.4.2.

In our framework, $\mathbb{PS}$ can be seen as the formalisation of a piece of additional background knowledge. The propositions in the true set of a $\mathbb{PS}$ represents that the user is confident that those assertions are true, while the ones in the false set refer to something that the user is sure about its falsity. Although this background knowledge is incomplete because $\mathbb{PS}$ does not cover all possible expressions over the signature, it provides the basis of the quantitative measurement of the entrenchment of axioms in a theory. Based on $\mathbb{PS}$, a partial solution of evaluating epistemic entrenchment is given in this section.

Therefore, we will discuss the entrenchment based on $\mathbb{PS}$ in terms of the axioms, the preconditions and the signature of a logical theory respectively. When there are multiple RPs to fix a fault, the ones changing the least entrenched items will be preferred. Because a proof can be found automatically based on SL-Resolution[2], all of the calculations in this section can be done automatically without human interaction.

6.2.1 Axiom Entrenchment w.r.t. $\mathbb{PS}$

An axiom's entrenchment will be evaluated based on how much $\mathbb{PS}$ is supported by that axiom. A logical theory should fully follow its $\mathbb{PS}$.

The contribution of an axiom (a) to a goal (g) is the percentage of g's proofs in which a is involved, as given in Equation 6.5, where $\mathcal{N}(p_g)$ is the total number of the proofs of g, while $\mathcal{N}(p_{ga})$ returns the number of g's proofs in which axiom a is involved.

[2]As discussed in §4.3, in Datalog, proofs can always be detected, if there are any, based on SL-Resolution.

$$c(a,g) = \frac{\mathcal{N}(p_{ga})}{\mathcal{N}(p_g)} \tag{6.5}$$

Based on the contribution $c(a,g)$ of an axiom a to a goal g, the entrenchment based on a $\mathbb{PS}$ is defined as the *preference entrenchment* (PE) below.

Definition 6.2.1 (Preference Entrenchment ($\mathcal{PE}$)). *The* preference entrenchment *of an axiom $\mathcal{PE}(a) =< +\mathcal{C}_t(a),\ -\mathcal{C}_f(a) >$, is given by Equation 6.6 and 6.7, where the propositions in $\mathcal{T}(\mathbb{PS})$ and $\mathcal{F}(\mathbb{PS})$ are tp_i for $0 \le i \le n$ and fp_j for $0 \le j \le m$ respectively.*

$$\mathcal{C}_t(a) = (c(a,tp_1), c(a,tp_2), \ldots, c(a,tp_n)) \tag{6.6}$$

$$\mathcal{C}_f(a) = (c(a,fp_1), c(a,fp_2), \ldots, c(a,fp_m)) \tag{6.7}$$

Equation 6.6 gives the contribution of an axiom to the true set of $\mathbb{PS}$, while Equation 6.7 refers to its involvement to prove members of the false set.

Theorem 6.2.1. *For a fault-free theory $\mathbb{T}$, we have:*

$$\mathcal{IS}(\mathbb{T},\mathbb{PS}) = \varnothing, \iff \forall p_i \in \mathcal{T}(\mathbb{PS}), \exists a, a \in \mathbb{T}, c(a,\ p_i) \ne 0 \tag{6.8}$$

$$\mathcal{IC}(\mathbb{T},\mathbb{PS}) = \varnothing, \iff \forall a, a \in \mathbb{T},\ \mathcal{C}_f(a) = [\boldsymbol{0}] \tag{6.9}$$

A list whose members are all zeros is written as [**0**] and it is at the same length as the list it equals or compares.

Example 6.2.1. *Bird Theory.*

$$bird(X) \implies fly(X) \tag{A1}$$

$$bird(X) \implies feathered(X) \tag{A2}$$

$$penguin(Y) \implies bird(Y) \tag{A3}$$

$$\implies penguin(tweety) \tag{A4}$$

$$\implies bird(polly) \tag{A5}$$

$\mathcal{T}(\mathbb{PS}) = \{penguin(tweety), feathered(tweety), fly(polly)\}$
$\mathcal{F}(\mathbb{PS}) = \{fly(tweety)\}$

In Example 6.2.1, each proposition in $\mathcal{T}(\mathbb{PS})$ has one proof: $penguin(tweety)$'s proof is constituted by A4; $feathered(tweety)$ by $(A4,A3,A2)$, and $fly(polly)$ by $(A5,A1)$. The proposition $fly(tweety)$ in $\mathcal{F}(\mathbb{PS})$ is proved by axioms: $(A4,A3,A1)$. According to Definition 6.2.1, the preference entrenchment of each axiom is calculated as the following:

$$\begin{aligned}\mathcal{C}_t(A1) = (0,0,1); \mathcal{C}_f(A1) = (-1)\\ \mathcal{C}_t(A2) = (0,1,0); \mathcal{C}_f(A2) = (0)\\ \mathcal{C}_t(A3) = (0,1,0); \mathcal{C}_f(A3) = (-1)\\ \mathcal{C}_t(A4) = (1,1,0); \mathcal{C}_f(A4) = (-1)\\ \mathcal{C}_t(A5) = (0,0,1); \mathcal{C}_f(A5) = (0)\end{aligned} \tag{6.10}$$

The basic idea of preference entrenchment $\mathcal{PE}$ is that *the more respectful to* $\mathbb{PS}$ *an axiom is, the more entrenched that axiom is in the theory*. For an axiom, its single contribution to each of the propositions in $\mathbb{PS}$ plays a more important role than its overall contribution. Assume that $B1$ and $B2$ are two axioms whose preference entrenchments are shown by Equation 6.11 and 6.12, respectively. It can be seen that $B2$ is essential to all the proofs of the first proposition in its $\mathcal{T}(\mathbb{PS})$, shown by the red 1 in Equation 6.12. Without $B2$, the first proposition in $\mathcal{T}(\mathbb{PS})$ would be unprovable, while without $B1$, no new insufficiency would be introduced. Therefore, $B2$ should be more entrenched than $B1$, although the sum of $\mathcal{PE}(B1)$ is greater than $\mathcal{PE}(B2)$.

$$\mathcal{PE}(B1) =< [0.5,0.5,0.1],[0,0,0] > \tag{6.11}$$

$$\mathcal{PE}(B2) =< [1,0,0],[0,0,0] > \tag{6.12}$$

Therefore, the comparison of $\mathcal{PE}$ needs a list comparison function. We will write the length of a list A as $\mathcal{L}(A)$, and lists $A > B$ when A is lexicographic bigger than B.

Definition 6.2.2 (Sorted List Comparison Function)**.** *A list comparison function* γ *is used to compare two lists of the same length after sorting their elements from large to small. If the sorted lists of* $\mathcal{C}_1$ *and* $\mathcal{C}_2$ *are* $\mathcal{C}_{1s}$ *and* $\mathcal{C}_{2s}$ *respectively, then* γ *comparison results in:*

$$\begin{aligned}\mathcal{C}_1 >_\gamma \mathcal{C}_2 \textit{ iff } \mathcal{L}(\mathcal{C}_1) = \mathcal{L}(\mathcal{C}_2), \mathcal{C}_{1s} > \mathcal{C}_{2s}\\ \mathcal{C}_1 =_\gamma \mathcal{C}_2 \textit{ iff } \mathcal{C}_{1s} = \mathcal{C}_{2s}\end{aligned} \tag{6.13}$$

where $\mathcal{L}(\mathcal{C}_1) = \mathcal{L}(\mathcal{C}_2)$.

Based on the above comparison function, the sorted lists need to be computed first, e.g., $[1,1,0]$ is the sorted list of $\mathcal{C}_t(A4) = (1,1,0)$, and $[1,0,0]$ of $\mathcal{C}_t(A5) = (0,0,1)$. Then the entrenchment of axioms in Equation 6.10 are compared based on $\mathcal{T}(\mathbb{PS})$ and $\mathcal{F}(\mathbb{PS})$ respectively, shown in Equation 6.14.

$$\begin{aligned}\mathcal{C}_t(A4) =_\gamma [1,1,0] >_\gamma \mathcal{C}_t(A1) =_\gamma \mathcal{C}_t(A2) =_\gamma \mathcal{C}_t(A3) =_\gamma \mathcal{C}_t(A5) =_\gamma [1,0,0]\\ \mathcal{C}_f(A2) =_\gamma \mathcal{C}_f(A5) =_\gamma [0] >_\gamma \mathcal{C}_f(A1) =_\gamma \mathcal{C}_f(A3) =_\gamma \mathcal{C}_f(A4) =_\gamma [-1]\end{aligned} \tag{6.14}$$

Breaking a proof is much easier than building one because the latter needs to build all necessary proof steps, while the former can be done by breaking just one proof step. Therefore, $\mathcal{C}_t$ is seen as more important than $\mathcal{C}_f$ so that PEs can be compared based on lexicographic order.

In Equation 6.14, the most preference entrenched axioms are A2 and A5, while the least entrenched ones are A1 and A3, shown in Example 6.2.2.

$$\mathcal{PE}(A2) = \mathcal{PE}(A5) > \mathcal{PE}(A4) > \mathcal{PE}(A1) = \mathcal{PE}(A3) \tag{6.15}$$

Example 6.2.2. *Bird Theory: order axioms from the most entrenched to the least.*

$$bird(X) \implies feathered(X) \tag{A2}$$
$$\implies bird(polly) \tag{A5}$$
$$\implies penguin(tweety) \tag{A4}$$
$$bird(X) \implies fly(X) \tag{A1}$$
$$penguin(Y) \implies bird(Y) \tag{A3}$$

$\mathcal{T}(\mathbb{PS}) = \{penguin(tweety), feathered(tweety), fly(polly)\}$
$\mathcal{F}(\mathbb{PS}) = \{fly(tweety)\}$

Assuming that there are repairs which delete an axiom in Example 6.2.2, then the ones which delete A1 or A3 will be suggested because they changed the least entrenched axioms.

However, it can be seen that all $\mathcal{C}_t(A1) - \mathcal{C}_t(A5)$ have 1 in their lists, which means that all of axioms are necessary to prove the propositions in $\mathcal{T}(\mathbb{PS})$, thus deleting any of the axioms will introduce insufficiency. Therefore, reformation will provide a better repair as it changes the signature rather than axioms. The entrenchment of the signature will be discussed in the following section.

6.2.2 Entrenchment of Preconditions

In this section, the entrenchment of a precondition in a rule will be analysed based on $\mathbb{PS}$. In a logical theory, when there is an extra precondition in a rule axiom, some of the original theorems may become unprovable in the current theory. This theorem difference is defined as the impact of a precondition.

Definition 6.2.3 (Precondition Impact)**.** *In logical theory $\mathbb{T}$, if the rule axiom R's preconditions are $p_i(\vec{t_i})$s, $1 \leq i \leq n$, then the impact of precondition $p_i(\vec{t_i})$ of R is the theorems difference between $\mathbb{T}$ and $\mathbb{T}'$ (6.17). This impact is given by Equation 6.18.*

$$R' = R - p_i(\vec{t_i}) \tag{6.16}$$

$$\mathbb{T}' = (\mathbb{T} \dot{-} R) \dot{+} R', \text{ where } R \in \mathbb{T} \tag{6.17}$$

$$\mathcal{PI}(p_i(\vec{t_i})) = \{\alpha |\ \alpha \in \mathcal{C}(\mathbb{T}'),\ \alpha \notin \mathcal{C}(\mathbb{T})\} \tag{6.18}$$

In the above, $\mathbb{T}$ only contains axioms without theorems and R is one of its axioms; function $-$ removes a precondition from a rule; functions $\dot{-}$, $\dot{+}$ remove or add one axiom to a set of axioms respectively and $\mathcal{C}$ returns all of the theorems of a theory.

Theorem 6.2.2. *When the impact of a precondition is the empty set, that precondition can be removed from the rule without changing the logical consequences of the theory.*

$$\mathcal{PI}(p_i(\vec{t_i})) = \varnothing \iff \mathcal{C}(\mathbb{T}') = \mathcal{C}(\mathbb{T}) \tag{6.19}$$

Proof: If $\mathcal{PI}(p_i(\vec{t_i})) = \varnothing$, then according to Equation 6.18, $\nexists\, \alpha, \alpha \in \mathcal{C}(\mathbb{T}'),\ \alpha \notin \mathcal{C}(\mathbb{T})$. Thus, $\forall \alpha, \alpha \in \mathcal{C}(\mathbb{T}'),\ \alpha \in \mathcal{C}(\mathbb{T})$, which equals to

$$\mathcal{C}(\mathbb{T}') \subseteq \mathcal{C}(\mathbb{T}) \tag{6.20}$$

Meanwhile, all proofs that contain R remains after removing $p_i(\vec{t_i})$ from R and $\mathbb{T}'$ could have more theorems than $\mathbb{T}$ as there is one precondition less in R'.

$$\mathcal{C}(\mathbb{T}) \subseteq \mathcal{C}(\mathbb{T}') \tag{6.21}$$

Combining 6.20 and 6.21, Equation 6.19 is proved.

When the precondition impact overlaps with the propositions in $\mathbb{PS}$, the entrenchment of that precondition can be calculated as the following.

Definition 6.2.4 (Entrenchment of a Precondition (EoP))**.** *The entrenchment of a precondition is decided by its impact on $\mathbb{PS}$, calculated by Equation 6.22, where the rule R in $\mathbb{T}$ contains the precondition $p(\vec{t})$, but in $\mathbb{T}'$; n_{is} is the* negated *number of insufficiencies caused by the inclusion of $p(\vec{t})$ in rule R and n_{ic} is the number of the incompatibilities caused by the absence of $p(\vec{t})$ in rule R.*

$$\mathcal{E}(p(\vec{t}), R) = (n_{is},\ n_{ic}), \tag{6.22}$$

$$n_{is} = -|\{\alpha | \forall \alpha \in \mathcal{T}(\mathbb{PS}),\ \mathbb{T} \nvdash \alpha, \mathbb{T}' \vdash \alpha\}| \leq 0 \tag{6.23}$$

$$n_{ic} = |\alpha | \forall \alpha \in \mathcal{F}(\mathbb{PS}),\ \mathbb{T} \nvdash \alpha, \mathbb{T}' \vdash \alpha| \geq 0 \tag{6.24}$$

where $\mathbb{T}' = \mathbb{T} \dot{-} R \dot{+} R'$, $R' = R - p(\vec{t})$.

Based on Definition 4.3.1, Equation 6.23 and 6.24 can be rewritten as the following.

$$n_{is} = -|\mathcal{IS}(\mathbb{T},\mathbb{PS}) - \mathcal{IS}(\mathbb{T}',\mathbb{PS})| \tag{6.25}$$

$$n_{ic} = |\mathcal{IC}(\mathbb{T}',\mathbb{PS}) - \mathcal{IC}(\mathbb{T},\mathbb{PS})| \tag{6.26}$$

The more insufficiencies are caused by the inclusion of a precondition, the less entrenched that precondition should be. Therefore, it is a negative integer in Equation 6.23. On the other hand, the more incompatibilities are caused by the absence of a precondition, the more entrenched that precondition should be. Thus, it is a positive integer in Equation 6.24.

Specification 5 (Precondition Entrenchment Comparison)**.** *One precondition is more entrenched than another if and only if its corresponding theory dominates the other's.*

$$\begin{gathered} \textit{Let } \mathcal{E}(p_1(\vec{t_1}),R_1) = (n_{is1},\ n_{ic1}),\ \mathcal{E}(p_2(\vec{t_2}),R_2) = (n_{is2},\ n_{ic2}), \\ \textit{then } \mathcal{E}(p_1(\vec{t_1}),R_1) \geq \mathcal{E}(p_2(\vec{t_2}),R_2) \iff n_{is1} \geq^* n_{is2} \wedge n_{ic1} \geq^* n_{ic2} \end{gathered} \tag{6.27}$$

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

Similar to belief changes, the precondition changes include adding preconditions and/or deleting preconditions. Taking the theories in Definition 6.2.3 as an example, when the original theory is $\mathbb{T}$, and the repaired theory is $\mathbb{T}'$, the precondition change is a deletion. On the other hand, if the original theory is $\mathbb{T}'$, and the repaired theory is $\mathbb{T}$, the precondition change is an addition.

Based on the defined EoP, the precondition change which maximises the overall EoPs of a rule while repairing the faults will be conducted to repair the faulty theory, e.g., add the most entrenched and/or delete the least entrenched.

Based on the statistical set of faults given by Definition 4.3.1, the strictly dominated repair is defined as the following.

Definition 6.2.5 (Strictly Dominated Repair)**.** *Given two repairs ν_k and ν_j, ν_j is strictly dominated by ν_k iff:*

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

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

Then the aforementioned EoP maximisation corresponds to the selection of the repairs which are not dominated by others, which is Theorem 6.2.3.

Theorem 6.2.3. *The removal of a more entrenched precondition $p_1(\vec{t_1})$ is strictly dominated by the removal of a less entrenched precondition $p_2(\vec{t_2})$ from rule R.*

$$\begin{aligned}&\mathcal{E}(p_1(\vec{t_1}),R) \geq \mathcal{E}(p_2(\vec{t_2}),R) \iff \\ &|\mathcal{IS}(\mathbb{T}_1,\mathbb{PS})| \geq^* |\mathcal{IS}(\mathbb{T}_2,\mathbb{PS})| \wedge |\mathcal{IC}(\mathbb{T}_1,\mathbb{PS})| \geq^* |\mathcal{IC}(\mathbb{T}_2,\mathbb{PS})|\end{aligned}$$

where $\mathbb{T}_1 = \mathbb{T}\dot{-}R\dot{+}(R - p_1(\vec{t_1}))$, $\mathbb{T}_2 = \mathbb{T}\dot{-}R\dot{+}.(R - p_2(\vec{t_2}))$.

Proof. As $\mathcal{E}(p_1(\vec{t_1}),R) \geq \mathcal{E}(p_2(\vec{t_2}),R)$, bring Equation 6.23 and 6.24 to 6.27 so it can be concluded that:

$$\begin{aligned}&-|\mathcal{IS}(\mathbb{T},\mathbb{PS}) - \mathcal{IS}(\mathbb{T}_1,\mathbb{PS})| \geq^* -|\mathcal{IS}(\mathbb{T},\mathbb{PS}) - \mathcal{IS}(\mathbb{T}_2,\mathbb{PS})| \wedge \\ &|\mathcal{IC}(\mathbb{T}_1,\mathbb{PS}) - \mathcal{IC}(\mathbb{T},\mathbb{PS})| \geq^* |\mathcal{IC}(\mathbb{T}_2,\mathbb{PS}) - \mathcal{IC}(\mathbb{T},\mathbb{PS})|\end{aligned} \tag{6.28}$$

$\forall\alpha$, if $\mathbb{T} \vdash \alpha$, then $\mathbb{T}_1 \vdash \alpha \wedge \mathbb{T}_2 \vdash \alpha$, so it can be concluded that:

$$\begin{aligned}\mathcal{IS}(\mathbb{T}_1,\mathbb{PS}) \subseteq \mathcal{IS}(\mathbb{T},\mathbb{PS}) \wedge \mathcal{IC}(\mathbb{T},\mathbb{PS}) \subseteq \mathcal{IC}(\mathbb{T}_1,\mathbb{PS}) \\ \mathcal{IS}(\mathbb{T}_2,\mathbb{PS}) \subseteq \mathcal{IS}(\mathbb{T},\mathbb{PS}) \wedge \mathcal{IC}(\mathbb{T},\mathbb{PS}) \subseteq \mathcal{IC}(\mathbb{T}_2,\mathbb{PS})\end{aligned} \tag{6.29}$$

Then Equation 6.28 can be simplified as the following.

$$\begin{aligned}&-(|\mathcal{IS}(\mathbb{T},\mathbb{PS})| - |\mathcal{IS}(\mathbb{T}_1,\mathbb{PS})|) \geq^* -(|\mathcal{IS}(\mathbb{T},\mathbb{PS})| - |\mathcal{IS}(\mathbb{T}_2,\mathbb{PS})|) \wedge \\ &|\mathcal{IC}(\mathbb{T}_1,\mathbb{PS})| - |\mathcal{IC}(\mathbb{T},\mathbb{PS})| \geq^* |\mathcal{IC}(\mathbb{T}_2,\mathbb{PS})| - \mathcal{IC}(\mathbb{T},\mathbb{PS})|\end{aligned} \tag{6.30}$$

By removing $\mathcal{IS}(\mathbb{T},\mathbb{PS})$ and $\mathcal{IC}(\mathbb{T},\mathbb{PS})$ from both sides of the above inequalities, the following equation is derived.

$$|\mathcal{IS}(\mathbb{T}_1,\mathbb{PS})| \geq^* |\mathcal{IS}(\mathbb{T}_2,\mathbb{PS})| \wedge |\mathcal{IC}(\mathbb{T}_1,\mathbb{PS})| \geq^* |\mathcal{IC}(\mathbb{T}_2,\mathbb{PS})| \tag{6.31}$$

Thus, Theorem 6.2.3 is proved. □

Theorem 6.2.4. *The expansion of a more entrenched precondition $p_1(\vec{t_1})$ strictly dominates the expansion of a less entrenched precondition $p_2(\vec{t_2})$ to rule R.*

$$\begin{aligned}&\mathcal{E}(p_1(\vec{t_1}),R_1) \geq \mathcal{E}(p_2(\vec{t_2}),R_2) \iff \\ &|\mathcal{IS}(\mathbb{T}_1,\mathbb{PS})| \leq^* |\mathcal{IS}(\mathbb{T}_2,\mathbb{PS})| \wedge |\mathcal{IC}(\mathbb{T}_1,\mathbb{PS})| \leq^* |\mathcal{IC}(\mathbb{T}_2,\mathbb{PS})|\end{aligned}$$

where $R_1 = R + p_1(\vec{t_1})$, $R_2 = R + p_2(\vec{t_2})$, $\mathbb{T}_1 = \mathbb{T}\dot{-}R\dot{+}R_1$, $\mathbb{T}_2 = \mathbb{T}\dot{-}R\dot{+}R_2$.

Proof. Let $\mathbb{T}$ and $\mathbb{T}_1$in Theorem 6.2.4 be $\mathbb{T}'$ and $\mathbb{T}$ in Equation 6.23 and 6.24 respectively. Then:

$$\mathcal{E}(p_1(\vec{t_1}),R_1) = (-|\mathcal{IS}(\mathbb{T}_1,\mathbb{PS}) - \mathcal{IS}(\mathbb{T},\mathbb{PS})|, |\mathcal{IC}(\mathbb{T},\mathbb{PS}) - \mathcal{IC}(\mathbb{T}_1,\mathbb{PS})|)$$

Similarly, $\mathcal{E}(p_2(\vec{t_2}), R_1)$ can be written as the following.

$$\mathcal{E}(p_2(\vec{t_2}), R_2) = (-|\mathcal{IS}(\mathbb{T}_2, \mathbb{PS}) - \mathcal{IS}(\mathbb{T}, \mathbb{PS})|, |\mathcal{IC}(\mathbb{T}, \mathbb{PS}) - \mathcal{IC}(\mathbb{T}_2, \mathbb{PS})|)$$

As $\mathcal{E}(p_1(\vec{t_1}), R_1) \geq \mathcal{E}(p_2(\vec{t_2}), R_2)$, bring the above equations to Equation 6.27.

$$\begin{aligned} -|\mathcal{IS}(\mathbb{T}_1, \mathbb{PS}) - \mathcal{IS}(\mathbb{T}, \mathbb{PS})| \geq^* -|\mathcal{IS}(\mathbb{T}_2, \mathbb{PS}) - \mathcal{IS}(\mathbb{T}, \mathbb{PS})| \wedge \\ |\mathcal{IC}(\mathbb{T}, \mathbb{PS}) - \mathcal{IC}(\mathbb{T}_1, \mathbb{PS})| \geq^* |\mathcal{IC}(\mathbb{T}, \mathbb{PS}) - \mathcal{IC}(\mathbb{T}_2, \mathbb{PS})| \end{aligned} \tag{6.32}$$

$\forall\alpha$, if $\mathbb{T}_1 \implies \alpha$, then $\mathbb{T} \vdash \alpha$ and $\forall\beta$, if $\mathbb{T}_2 \implies \beta$ then $\mathbb{T} \vdash \beta$, so it can be concluded that:

$$\begin{aligned} \mathcal{IS}(\mathbb{T}, \mathbb{PS}) \subseteq \mathcal{IS}(\mathbb{T}_1, \mathbb{PS}) \wedge \mathcal{IC}(\mathbb{T}_1, \mathbb{PS}) \subseteq \mathcal{IC}(\mathbb{T}, \mathbb{PS}) \\ \mathcal{IS}(\mathbb{T}, \mathbb{PS}) \subseteq \mathcal{IS}(\mathbb{T}_2, \mathbb{PS}) \wedge \mathcal{IC}(\mathbb{T}_2, \mathbb{PS}) \subseteq \mathcal{IC}(\mathbb{T}, \mathbb{PS}) \end{aligned} \tag{6.33}$$

Then Equation 6.32 can be simplified as the following.

$$\begin{aligned} -(|\mathcal{IS}(\mathbb{T}_1, \mathbb{PS})| - |\mathcal{IS}(\mathbb{T}, \mathbb{PS})|) \geq^* -(|\mathcal{IS}(\mathbb{T}_2, \mathbb{PS})| - |\mathcal{IS}(\mathbb{T}, \mathbb{PS})|) \wedge \\ |\mathcal{IC}(\mathbb{T}, \mathbb{PS})| - |\mathcal{IC}(\mathbb{T}_1, \mathbb{PS})| \geq^* |\mathcal{IC}(\mathbb{T}, \mathbb{PS})| - |\mathcal{IC}(\mathbb{T}_2, \mathbb{PS})| \end{aligned} \tag{6.34}$$

By removing $\mathcal{IS}(\mathbb{T}, \mathbb{PS})$ and $\mathcal{IC}(\mathbb{T}, \mathbb{PS})$ from both sides of the above inequalities, the following is derived.

$$|\mathcal{IS}(\mathbb{T}_1, \mathbb{PS})| \leq^* |\mathcal{IS}(\mathbb{T}_2, \mathbb{PS})| \wedge |\mathcal{IC}(\mathbb{T}_1, \mathbb{PS})| \leq^* |\mathcal{IC}(\mathbb{T}_2, \mathbb{PS})| \tag{6.35}$$

Thus, Theorem 6.2.4 is proved. □

The defined EoP not only reflects the importance of a precondition w.r.t. $\mathbb{PS}$, but also details the exact faults it can cause, e.g., the proposition of an insufficiency which $\mathbb{T}$ fails in proving. Therefore, it is known whether a precondition change repairs a targeted fault. When there are multiple solutions of precondition changes to one fault, the one maximising the overall EoP of that rule will be applied.

In summary, EoP is based on the impact of the precondition on $\mathbb{PS}$, which is scored by a pair (n_{is}, n_{ic}), where $n_{is} \leq 0$ and $-n_{is}$ corresponds to the number of insufficiencies caused by having that precondition in its rule; $n_{ic} \geq 0$ where n_{ic} is the number of incompatibilities caused by the absence of that precondition in its rule. By applying lexicographical ordering, EoPs in a rule can be compared. Accordingly, when there are multiple faults, the precondition change which minimises the number of faults is applied. In addition, the candidates of precondition change that fix the targeted fault(s) are computed during the evaluation of EoPs. Therefore, ones maximising the overall EoPs of the rule are applied.

6.2.3 Entrenchment of Signature

Conceptual change works by adapting the signature in which a logical theory is written. Measuring the entrenchment of the signature helps to rank the repairs of conceptual change.

In this section, entrenchment will be defined and measured from the perspectives of different signature elements including predicate names and the arguments. An argument can be a variable or a constant.

First we will analyse signature elements which will help us to define signature entrenchment. Two signature elements can be independent of each other. For example, a predicate can be independent of another predicate; the arity of predicate $mum/2$ is independent of the arity of predicate $swan/1$. However, *when a predicate occurs in the head of a rule, it is linked to ones in the body of that rule and vice versa.* Example 6.2.3 is given to illustrate the possible linkages among signature elements in a logical theory.

Example 6.2.3. *Swan Theory.*

$$\begin{aligned} german(X) &\Longrightarrow european(X) && \text{(A1)} \\ european(X) \wedge swan(X) &\Longrightarrow white(X) && \text{(A2)} \\ &\Longrightarrow german(bruce) && \text{(A3)} \\ &\Longrightarrow swan(bruce) && \text{(A4)} \end{aligned}$$

The logical consequences of the theory include $european(bruce)$ and $white(bruce)$. If the predicate *german* is split, and the predicate in A3 is renamed to *germanResidence*, then the original logical consequences will become non-derivable. On the other hand, if it is the argument of $A3$ being renamed, e.g., from *bruce* to *liza*, then the logical consequences will include only $european(liza)$. In addition, if *european* has a new variable argument in $A1$, according to the Datalog safety requirement, then the same variable needs to be added into *german* in the body of A1 as well. Hence, the arity change of *european* results in the arity change of *german*.

It can be seen that when a predicate is involved in a rule, a change of it could result in the corresponding changes in its linked predicates, especially the ones on the other side of the implication in that rule. Therefore, rules play an important role in analysing the signature structure, especially from the perspective of changing a signature.

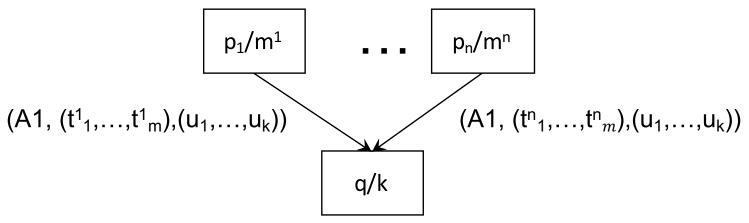
For describing the relation between predicates, a directed graph comes to our aid as a descriptive device.

Definition 6.2.6 (Theory Graph)**.** *A theory graph represents the links between predicates in a logical theory by a finite set of nodes and edges.*
Node: *each node corresponds to a predicate together with its arity.*
Edge: *each edge is an arrow that connects one or two nodes in the graph. The components of an edge are its direction and label.*
Path: *a path is the nodes connected by a sequence of edges in same direction.*

The axiom $A1$ in Equation 6.36, which has n propositions in its body and one proposition in its head, could be drawn as Figure 6.1 based on Definition 6.2.6.

$$A1.\ \bigwedge_{i=1}^{n} p_i(t_1^i,...,t_m^i) \implies q(u_1,...,u_k) \tag{6.36}$$

The *direction* of an edge is from a node in the body of the rule to the head of that rule. Meanwhile, each edge is labelled by a 3-tuple including the name of the axiom and the arguments of the tail node followed by the arguments of the head node. Each tail node corresponds to a proposition in the body of the rule and the head node represents the proposition in the head of the rule.



p1/m1
...
pn/mn
(A1, (t11,...,t1m),(u1,...,uk))
(A1, (tn1,...,tnm),(u1,...,uk))
q/k

Figure 6.1: Theory sub-graph of A1 in Equation 6.36.

An assertion can be seen as a rule without preconditions. In a theory graph, an assertion is drawn as a head node pointed by an edge with a special tail node of ‘true’.

$$A2.\ q(c_1,...,c_j) \tag{6.37}$$

The axiom $A2$ in Equation 6.37 is represented as in Figure 6.2.

Specification 6. *If predicate p/m is in the signature of a logical theory, then node p/m occurs exactly once in the theory’s corresponding theory graph.*

Specification 6 requires that no duplicates are allowed in a theory graph. When a predicate is involved in multiple axioms, there will be multiple edges corresponding

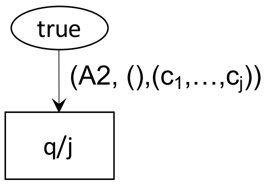


Figure 6.2: Theory sub-graph of A2 in Equation 6.37.

to these axioms. In each edge, the first element of its label, which is the name of the axiom e.g., A1 in Figure 6.1 and A2 in Figure 6.2, shows to which axiom the edge belongs. Hence different axioms can be represented correspondingly. When a predicate appears n times with different arguments in the body of a rule, there will be n edges from the predicate to that rule's head. Those edges should also be labelled by the rule number and the arguments accordingly, so their labels are different due to the distinct arguments. On the other hand, if a predicate appears in both the body and the head of a rule, then the labelled edge should go from the predicate to itself.

Theorem 6.2.5. *If there is no path from predicate p to predicate q in the theory's theory graph, then assertions of p cannot contribute to any proof of an assertion of q.*

Proof. In Theorem 6.2.5, no path from predicate p to predicate q in the theory's theory graph means that there is no rule connection between p and q. Therefore, assertions of p and q are independent from each other, so that assertions of p cannot contribute to any proof of an assertion of q. □

Theorem 6.2.5 decides whether adding a theorem of p has an impact of building the proofs of q's instances, which is important to the computation in §6.3.

Example 6.2.4. *A Mixed European Theory.*

$$\Longrightarrow german(bruce, residence) \quad (A1)$$

$$\Longrightarrow swan(bruce) \quad (A2)$$

$$german(X,Y) \Longrightarrow european(X,Y) \quad (A3)$$

$$european(X, variety) \wedge swan(X) \Longrightarrow white(X) \quad (A4)$$

$$flag(X,Y) \wedge white(Y) \Longrightarrow surrender(X) \quad (A5)$$

$$\begin{aligned} & surrender(X) \wedge war(X, worldWar2) \wedge \\ & european(X, party) \wedge leader(X, hitler) \Longrightarrow german(X, nazi) \quad (A6) \end{aligned}$$

Example 6.2.4 is a Datalog-like logical theory with six axioms. The first four axioms come from Example 6.2.3. The axiom A5 says that X surrenders if X gives a white flag Y. The last axiom describes that in the second world war, the surrendering European party led by Hitler is the Nazi party which is from Germany.

Figure 6.3 depicts a theory graph, where the 'true' node depicts the assertions in the theory[3] of Example 6.2.4. It can be seen that there are two loops in that theory graph. One loop is constituted of *german*/2, and *european*/2, and another is (*german*/2, *european*/2, *white*/1, *surrender*/1).

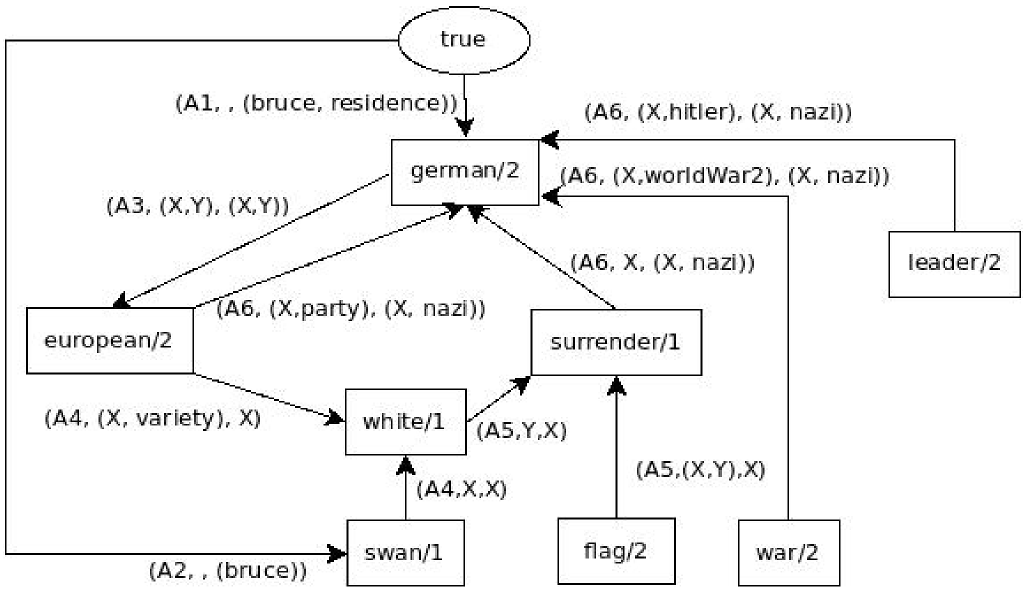


Figure 6.3: Theory graph of Example 5.3.

A theory graph shows how predicates are linked in a theory. For example, in Figure 6.3, predicate *german* is linked to *european* via A3 and A6. The change of a predicate could affect its linked ones. For example, if *white* is changed into black, then a flag in black would mean surrender. As a consequence, the meaning of a flag in terms of its colour is changed. It can be seen that the edges are essential for recovering a theory from its theory graph.

The entrenchment of predicate symbols will be evaluated based on the defined theory graph, following which the entrenchment of an argument will be evaluated based on its argument domain.

6.2.3.1 Predicate Entrenchment

Given a ℙ𝕊, all predicates that occur in ℙ𝕊, called *preferred predicates*, are fully trusted and most entrenched. Based on the preferred predicates, preferred distance is defined

[3]The 'true' node is not genuinely from the theory, and taking it into consideration will make analysis complicated without benefits. Therefore, we ignore it in our discussion.

to describe the predicate distance of a predicate symbol from the nearest preferred predicate.

Definition 6.2.7 (Preferred Distance)**.** *The preferred distance $d_p(p)$ of a predicate (p) is the is the number of edges on the shortest path from p to its nearest preferred predicate following the direction of each arrow.*

$$d_p(p) = \min_{\forall q \lhd \mathbb{PS}}(d(p,q)) \tag{6.38}$$

where $q \lhd \mathbb{PS}$ represents that q is a predicate and $\exists q(\vec{c}),\ q(\vec{c}) \in \mathcal{T}(\mathbb{PS}) \bigvee q(\vec{t}) \in \mathcal{F}(\mathbb{PS})$,

When a predicate has no path to any preferred predicates, it is called an *isolated predicate*. Here a path means a set of nodes connected by edges following one direction which is from the tail to the head of an edge. According to Definition 6.2.7, the preferred distance of an isolated predicate is infinity, and a preferred predicate's preferred distance is 0.

The motivation is that the further a predicate is from a preferred predicate p_p, the more problems could happen in terms of proving the statements of p_p. On the other hand, the nearer a predicate is from p_p, the more important it is for the statements of proving p_p. In other words, the nearer a predicate p is to preferred predicates, the more impact it has on the proofs of $\mathbb{PS}$ when changing p. Therefore, *the predicate which is nearer to a preferred predicate is more entrenched, and, therefore, is less likely to be changed.*

Specification 7. *Based on the preferred distance, the important properties that predicate entrenchment $e(p)$ should have are as follows, where p, p_1 and p_2 are predicates, and $\mathbb{S}_p$ is the set of predicates which occur in $\mathbb{PS}$ while $\mathbb{S}_t$ is the set of predicates which occur in the theory but not in $\mathbb{PS}$.*

1. $\forall p \in (\mathbb{S}_t \bigcup \mathbb{S}_p),\ e(p)$ *has exactly one value.*
   The entrenchment of a predicate should be just one value.

2. $0 \leq e(p) \leq 1$.
   The range of an entrenchment should be [0,1], where 0 means that a predicate is not trusted at all and 1 represents that the predicate is most entrenched and fully trusted[4].

[4]In Bayesian model and other probabilistic models (Gärdenfors, 1988), 1 is used as the value of the most entrenched. Although there was no explicit measurement of entrenchment, but only certain properties that entrenchment should have in the previous literature, it is good to be coherent with their work, which is to employ 1 representing the most entrenched here.

3. $\forall p_2 \in \mathbb{S}_p,\ e(p_2) = 1 \wedge \forall p_1 \in \mathbb{S}_t,\ 0 < e(p_1) < 1.$
   Because $\mathbb{PS}$ is more trusted than the theory, a preferred predicate is most entrenched whose entrenchment is 1. Any predicate appearing only in the theory is believed in some sense, but less than a preferred predicate. Meanwhile, any predicate that occurs in the theory is considered to convey some information. Therefore, its entrenchment should be bigger than 0 but smaller than 1.

4. $\forall p_1, p_2 \in \mathbb{S}_t,\ e(p_1) > e(p_2)$, *iff* $d_p(p_1) < d_p(p_2)$.
   When neither predicate occurs in $\mathbb{PS}$, p_1 is more entrenched than p_2 if and only if p_1 is closer to preferred predicates in terms of its preferred distance. The smaller $d_p(p_1)$ is, the more impact on $\mathbb{PS}$ changing p_1 will have. As preferred predicates should not be changed or effected, by assigning p_1 a bigger entrenchment value, it is less likely to be modified.

5. *If* $p_1 \in (\mathbb{S}_t \bigcup \mathbb{S}_p)$, *while* $p_2 \notin (\mathbb{S}_t \bigcup \mathbb{S}_p)$, *then* $e(p_1) > e(p_2)$.
   When p_2 is a predicate which neither appears in the theory nor $\mathbb{PS}$, it is an *unknown predicate*. Unknown predicate p_2 is considered to be less entrenched than the ones occur in the theory or $\mathbb{PS}$.

According to the above desired properties, Definition 6.2.1 is proposed as a measurement of predicate entrenchment $e(p)$ of a predicate p.

Definition 6.2.8 (Predicate Entrenchment)**.** *The entrenchment of a non-isolated predicate p_1 is given in terms of its preferred distance and the maximum preferred distance of all non-isolated predicates in the theory graph:*

$$\begin{aligned} e(p) &= 1 - \frac{d_p(p)}{d_{pMax} + 2} \\ \textit{where } d_{pMax} &= \max_{\forall q \in \mathbb{S}_t}(d_p(q)),\ q \textit{ is a non-isolated predicate.} \end{aligned} \tag{6.39}$$

On the other hand, the entrenchment of an isolated predicate p_2 is normalised as follows:

$$\begin{aligned} e(p_2) &= \frac{1}{d_{pMax} + 2} \\ \textit{where } d_{pMax} &= \max_{\forall q \in \mathbb{S}_t}(d_p(q)),\ q \textit{ is a non-isolated predicate.} \end{aligned} \tag{6.40}$$

If a predicate p_3 is an unknown predicate[5]*, which appears neither in the theory nor in*

[5]The zero entrenchment is defined to reflect that the existence of a predicate in the theory means that the predicate is trusted to have informational value.

$\mathbb{PS}$, *then its entrenchment is initialised as 0.*

$$e(p_3) = 0 \tag{6.41}$$

There are three disjoint kinds of predicates including non-isolated predicate(p_1), isolated predicate (p_2) and unknown predicate (p_3). A preferred predicate p is always a non-isolated predicate.

Assuming that in Example 6.2.4, $european/2$ and $white/2$ are the preferred predicates, which are highlighted in red in Figure 6.4, then the preferred distance and the entrenchment of each predicate can be measured. The results are shown in Table 6.1.

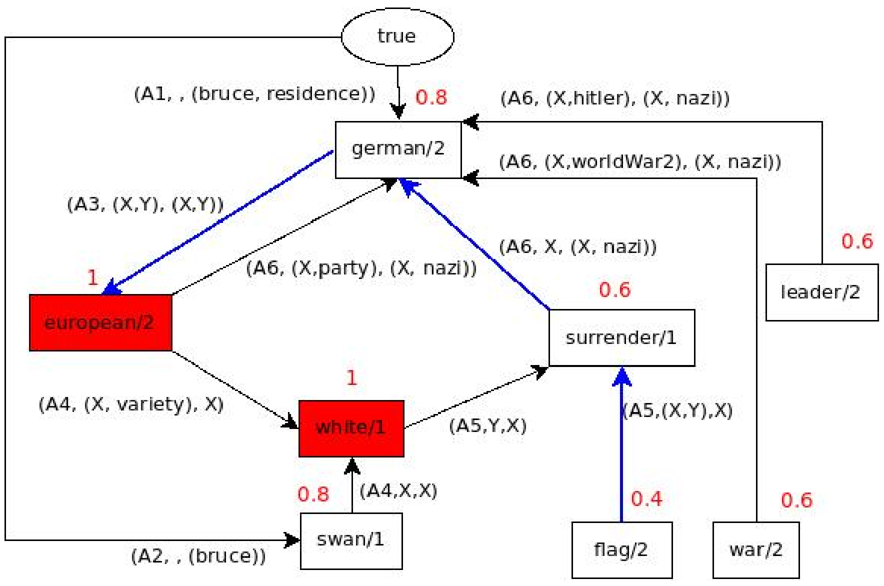


Figure 6.4: Theory graph of Example 5.3 with preferred predicates and the entrenchment of each predicate highlighted in red. The minimal path of $flag/2$ is highlighted in blue.

	european	*white*	*german*	*swan*	*war*	*leader*	*surrender*	*flag*
d_p	0	0	1	1	2	2	2	3
$e(p)$	1	1	0.8	0.8	0.6	0.6	0.6	0.4

Table 6.1: The preferred distance (d_p) and the entrenchment ($e(p)$) of each predicate in Example 6.2.4, where *european* and *white* are the preferred predicates.

As for Example 6.2.1, all predicates but $bird/1$ occur in $\mathbb{PS}$, therefore, $bird/1$ is the least entrenched predicate in that signature, as shown in 6.5.

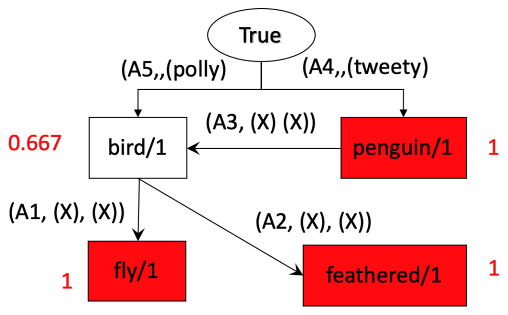


Figure 6.5: The theory graph of bird theory in Example 6.2.1: *the red nodes are the predicates in $\mathbb{PS}$, and the predicate entrenchment is attached beside each predicate in red.*

In the following paragraphs, we will prove that the defined measurement has the desired properties aforementioned.

Based on Equation 6.40 and 6.41, the entrenchment of an isolated predicate or an unknown predicate has one fixed value. Meanwhile, because the preferred distance of any non-isolated predicate has only one value, so that d_{pMax} is a fixed value in a given theory. Consequently, the entrenchment of a non-isolated predicate has only one value too. In summary, property 1 is held by our measurement.

The preferred distance of a non-isolated predicate p_1 satisfies:

$$0 \le d_p(p_1) \le d_{pMax} \tag{6.42}$$

According to Equation 6.39 and 6.42, the entrenchment of a non-isolated predicate p_1 is in the range $[\frac{2}{d_{pMax}+2}, 1]$:

$$\frac{2}{d_{pMax}+2} \le e(p_1) \le 1 \tag{6.43}$$

Based on Equation 6.43, 6.40 and 6.41, we can derive the following relation among the entrenchments of predicates in all different types, which proves property 2, the first half of property 3, the special case of property 4 and property 5.

$$0 = e(p_3) < \frac{1}{d_{pMax}+2} = e(p_2) < \frac{2}{d_{pMax}+2} \le e(p_1) \le 1 \tag{6.44}$$

When p is from $\mathbb{PS}$, then $d_p(p) = 0$. A preferred predicate p is a non-isolated predicate, whose entrenchment can be calculated according to Equation 6.39.

$$If\, p \in \mathbb{S}_p,\ then\ e(p) = 1 - \frac{d_p(p)}{d_{pMax}+2} = 1 - \frac{0}{d_{pMax}+2} = 1 \tag{6.45}$$

Combining Equation 6.44 and 6.45, it can be seen that the property 3 is proved.

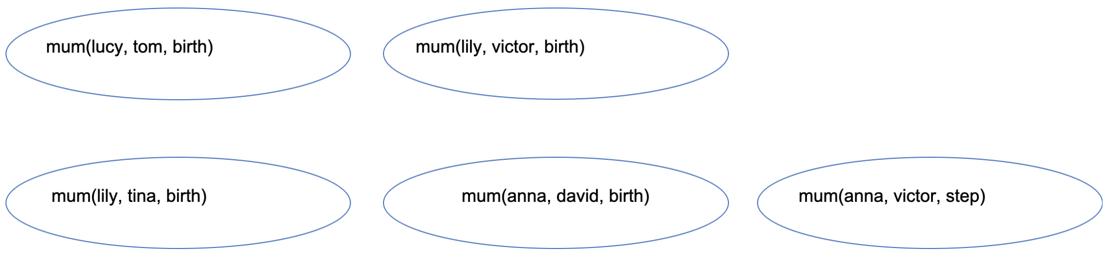


Figure 6.6: Theorems of predicate $mum/3$ in Example 4.4.2.

The difference of entrenchment between two non-isolated predicates p_{11} and p_{12} are given in the following equation.

$$e(p_{11}) - e(p_{12}) = 1 - \frac{d_p(p_{11})}{d_{pMax}+2} - (1 - \frac{d_p(p_{12})}{d_{pMax}+2}) = \frac{d_p(p_{12}) - d_p(p_{11})}{d_{pMax}+2} \tag{6.46}$$

Based on Equation 6.42 and 6.46, property 4 can be proved:

$$e(p_{11}) > e(p_{12}) \Leftrightarrow \frac{d_p(p_{12}) - d_p(p_{11})}{d_{pMax}+2} > 0 \Leftrightarrow d_p(p_{12}) > d_p(p_{11}) \tag{6.47}$$

In summary, the defined measurement of entrenchment has all the desired properties given by Specification 7.

6.2.3.2 Argument Entrenchment

Argument entrenchment is evaluated based on the argument domain, which is defined in Definition 4.4.1. The size of the domain of an argument reflects how diverse an argument is.

Figure 6.6 gives all the theorems of the theory in Example 4.4.2. The argument domains of predicate $mum/3$ are $\{lucy,\ lily,\ anna\}$, $\{tom,\ tina,\ victor,\ david\}$ and $\{birth, step\}$ respectively. The sizes of these domains are three, four and two respectively. Then the first two arguments with bigger domain are more diverse than the last argument: the former represents different persons while the latter only gives the type of mum, which is either $birth$ or $step$ in the current signature.

Assume that theorem $mum(lucy, tom, birth)$ is unknown but the domain of each argument, and each element in the domain is equi-probable. Then the probability of recovering that theorem from $mum(\_, tom, birth)$ is the reciprocal of the size of the missed first argument: $1/3$ and it is $1/4$ from $mum(lucy, \_, birth)$, $1/2$ from $mum(lucy, tom, \_)$.

Based on the known argument domain, the recovery probability of a lost argument reflects the informational value of that argument. We assume that *the smaller the* *recovery probability is, the more informational value that argument has.* As epistemic entrenchment is a notion describing the informational value of an element, the entrenchment of an argument can be evaluated based on its domain size.

Definition 6.2.9 (The Entrenchment of an argument)**.** *Let $\mathcal{E}_a(p,n)$ be the entrenchment of the nth argument of predicate p, $n \leq arity(p)$:*

$$\mathcal{E}_a(p,n) = \begin{cases} |\mathcal{D}(p,n)|, \ p \ntriangleleft \mathbb{PS} \\ Max(|\mathcal{D}(p',i)|) + 1, \ p \lhd \mathbb{PS}, p' \ntriangleleft \mathbb{PS}, i \leq arity(p') \end{cases} \tag{6.48}$$

where $p \lhd \mathbb{PS}$ represents that $\exists p(\vec{t}),\ p(\vec{t}) \in \mathcal{T}(\mathbb{PS}) \vee p(\vec{t}) \in \mathcal{F}(\mathbb{PS})$, and the function $\mathcal{D}(p,n)$ returns the argument domain of the nth argument of predicate p and Max returns the maximum value.

Therefore, in Figure 6.6, the third argument of $mum/3$ is the least entrenched.

Theorem 6.2.6. *The defined argument entrenchment is greater than one.*

$$\forall p,n,\ \mathcal{E}_a(p,n) \geq 1 \tag{6.49}$$

Proof. $\forall p,n,\ n \leq arity(p), |\mathcal{D}(p,n)| \geq 1$. So $Max(|\mathcal{D}(p,i)|) + 1 \geq 2$.
Therefore, $\forall p,n,\ \mathcal{E}_a(p,n) \geq 1$ based on Equation 6.48. □

Theorem 6.2.7. *A logical theory $\mathbb{T}$ can be simplified by omitting the nth argument of predicate p if p is not in $\mathbb{PS}$ and the entrenchment is one $\mathcal{E}_a(p,n) = 1$. If that argument is a variable in a rule, then weaken the remaining occurrences of that variable with the unique constant in the argument domain. Let the simplified theory be $\mathbb{T}'$ and p be the predicate of the omitted argument, then all theorems remains.*

$$\forall \alpha, \mathbb{T} \vdash \alpha, \mathbb{T}' \vdash \alpha* \tag{6.50}$$

where $\alpha = \alpha*$ if α is not a theorem with p, otherwise, $\alpha*$ is different from α by lacking of the omitted argument.

Proof. As $\mathcal{E}_a(p,n) = |\mathcal{D}(p,n)| = 1$, let $\mathcal{D}(p,n) = \{c\}$. Assume that R is the rule which contains $p(\vec{t})$ where $t_n = X$. Then in a proof which contains R, X can only be bound to c. After deleting that argument from all propositions of p, the remaining Xs in other propositions in R are replaced by c. These changes amount to binding X to c, so there is no effect to any proof which contains R.

Therefore, all theorems of the theory remain the same except the ones in which p lacks the omitted argument. □

From Theorem 6.2.7, it can be seen that the removal of the least entrenched argument is a minimal change under the defined argument entrenchment.

In summary, the entrenchment of axioms, preconditions and signature items are evaluated with scores, which is used to choose the repairs which change the least entrenched items. Notice that the entrenchments from different aspects cannot be compared, e.g., the entrenchment of a predicate and the entrenchment of an axiom. Therefore, different entrenchments are employed by the corresponding kinds of repair techniques independently, e.g., giving the proof of an incompatibility, the repair of axiom deletion will choose the least entrenched axioms in that proof, while the repairs of renaming a predicate will target at the least entrenched predicates which occur in that proof.

6.3 Maximal Set of Repair Plans (MSCR)

In a theory, there can be multiple faults which are caused by different axioms or signature items so that they can be repaired at the same time. In this case, their repairs commute and then the efficiency of the basic search strategy introduced in §4.4.2 is low because it cannot repair those faults at the same time. On the other hand, not all repairs of different faults can commute, e.g., two repairs targeting at one axiom.

In this section, a new search method is introduced, which computes maximal set of commutative repair plans (MSCR)s so that they can be applied together. As a result, fault-free theories can be found in a much shorter time. This method dramatically saves time especially when the number of independent faults are big.

6.3.1 Conditions of Combining Repair Plans

Each RP can have some side effects. Good side effects fix other faults simultaneously and bad effects introduce new faults. So it is possible that after applying one RP R_1, another RP R_2 won't be needed because the applied one also solves the fault which R_2 targets. On the other hand, R_1 may have changed the objects of R_2, e.g., R_1 merges predicate mother with mum, so all mother are replaced with mum; then if R_2 aims delete an axiom of $mother$, it cannot find it after R_1's application.

The condition of grouping RPs, e.g., R_1 and R_2, is that they *commute*: applying them in different order does not change the resulting theory. By employing function '$\cdot$' to represent the application of a RP to a theory, the commutation between R_1 and R_2

can be written as Equation 6.51.

$$\mathbb{T} \cdot R_1 \cdot R_2 = \mathbb{T} \cdot R_2 \cdot R_1 \tag{6.51}$$

To check whether RPs commute, the scope of which axioms RPs change is defined.

Definition 6.3.1 (Scope of a Repair Plan R)**.** *Let $\mathbb{T}$ be the original theory and $\mathbb{T}_r$ be the repaired theory by applying the RP R. Then the scope of R is $\mathbb{S}(R)$:*

$$\mathbb{S}(R) = \{Axiom | Axiom \in \mathbb{T} \wedge Axiom \notin \mathbb{T}_r\}$$

Accordingly, R_1 and R_2 commute when they satisfy the following conditions.

1. R_1 and R_2 do not repair the other's fault.

2. The scope of R_1 and R_2 do not overlap: $\mathbb{S}(R_1) \cap \mathbb{S}(R_2) = \varnothing$

The first condition guarantees that any grouped RP does fix at least one fault. Otherwise, if R_1 also fixes the fault which R_2 aims to fix, then R_2 is unnecessary, in which case, R_2 should not be applied after R_1 according to the minimal change postulate.

The second condition guarantees that the grouped RPs can be applied in any order. If there is any overlap between their scopes, then after applying one RP, the other may not be able to find its targeted axioms.

6.3.2 Compute the Maximal Sets of Commutative RPs

Given all RPs for all detected faults, we aim at computing maximal sets of commutative repair plan (MSCR).

Definition 6.3.2 (Maximal Set of Commutative RP)**.** *Given the whole set of all possible RPs for all detected faults, a maximal set of commutative repair plans is a collection of most commutative Repair Plans.*

Note that there could be multiple MSCRs and we compute all of them. Meanwhile, a RP can belong to more than one MSCRs. The above definition has the following properties.

The computation of MSCR is based on the following rules which check whether two RPs commute. The path mentioned below is a path in the theory graph[6] based on

[6]It is defined in Definition 6.2.6.

the new theory produced by applying the RP. According to Theorem 6.2.5, if there is no path from predicate p to predicate q in a theory graph, then an assertion of p will never be involved in a proof of an assertion of q. Thus, if no head predicate of an axiom is on a path to the predicate of an unprovable sub-goal, then any change on that axiom cannot make the unprovable goal of the insufficiency provable.

Theorem 6.3.1 (Commutation between RP of Incomp vs Incomp)**.** *Let R_1 and R_2 be two RPs for two incompatibilities, and $\mathbb{P}_1$, $\mathbb{P}_2$ the sets of input axioms which constitute the proofs that R_1 and R_2 block. Then R_1 and R_2 commute if $\mathbb{S}(R_1) \cap (\mathbb{P}_2 \cup \mathbb{S}(R_2)) = \mathbb{S}(R_2) \cap (\mathbb{P}_1 \cup \mathbb{S}(R_1)) = \varnothing$.*

Proof. As $\mathbb{S}(R_1) \cap \mathbb{P}_2 = \varnothing$, then $\forall \alpha \in \mathbb{P}_2 . \alpha \in \mathbb{T} \cdot R_1$. Thus, after applying R_1, the proof of $\mathbb{P}_2$ still exists, so R_2 will be generated by ABC to block the proof of $\mathbb{P}_2$.

Meanwhile, from $\mathbb{S}(R_1) \cap \mathbb{S}(R_2) = \varnothing$, it can be concluded that R_2 can be applied after R_1 because all the axioms it needs to change are not changed by R_1.

Similarly, if R_2 has been applied, R_1 will still be generated and it can be applied. Therefore, R_1 and R_2 commute because their generation and application do not affect each other. □

Theorem 6.3.2 (Commutation between RP of Insuff vs Insuff)**.** *Let R_1 and R_2 be two RPs for two insufficiencies, and $\mathbb{S}(R_1)$, $\mathbb{S}(R_2)$ be the scopes of R_1 and R_2, respectively. They commute if $\mathbb{S}(R_1) \cap \mathbb{S}(R_2) = \varnothing$, and no head predicate of an axiom in one's scope is on a path to the predicate of an unprovable sub-goal of the other insufficiency.*

Proof. $\forall p_1(\vec{t}_1) \wedge ... \wedge p_n(\vec{t}_n) \implies q(\vec{t}) \in \mathbb{S}(R_1)$, q is not on a *path* to the predicate of an unprovable sub-goal of the insufficiency targeted by R_2. Then any assertion of q will not unblock a proof of that insufficiency based on Theorem 6.2.5. Therefore, after applying R_1, R_2 will be generated to fix that insufficiency.

Again, from $\mathbb{S}(R_1) \cap \mathbb{S}(R_2) = \varnothing$, it can be concluded that R_2 can be applied after R_1 because all the axioms it needs to change are not changed by R_1.

Similarly, if R_2 has been applied, R_1 will still be generated and it can be applied. Therefore, R_1 and R_2 commute because their generation and application do not affect each other. □

Theorem 6.3.3 (Commutation between RP of Incomp vs Insuff)**.** *Let R_1 and R_2 be two RPs for an incompatibility and an insufficiency, and $\mathbb{S}(R_1)$, $\mathbb{S}(R_2)$ be the scopes of R_1 and R_2, respectively. They commute if $\mathbb{S}(R_1) \cap \mathbb{S}(R_2) = \varnothing$, and no head predicate of axioms in $R1$'s scope is on a path to the predicate of an unprovable sub-goal of the insufficiency.*

The proof of Theorem 6.3.3 can be formalised based on the ones of Theorem 6.3.1 and 6.3.2, so it is not given to avoid repetition.

Based on the above rules, MSCRs can be computed. The computation of MSCRs is similar to the computation of the minimal set given in 4.1.

By applying RPs in a MSCR together, the search space is reduced because the search branches of grouped RPs are merged into one. The comparison of search spaces are drawn in Figure 6.7 and 6.8.

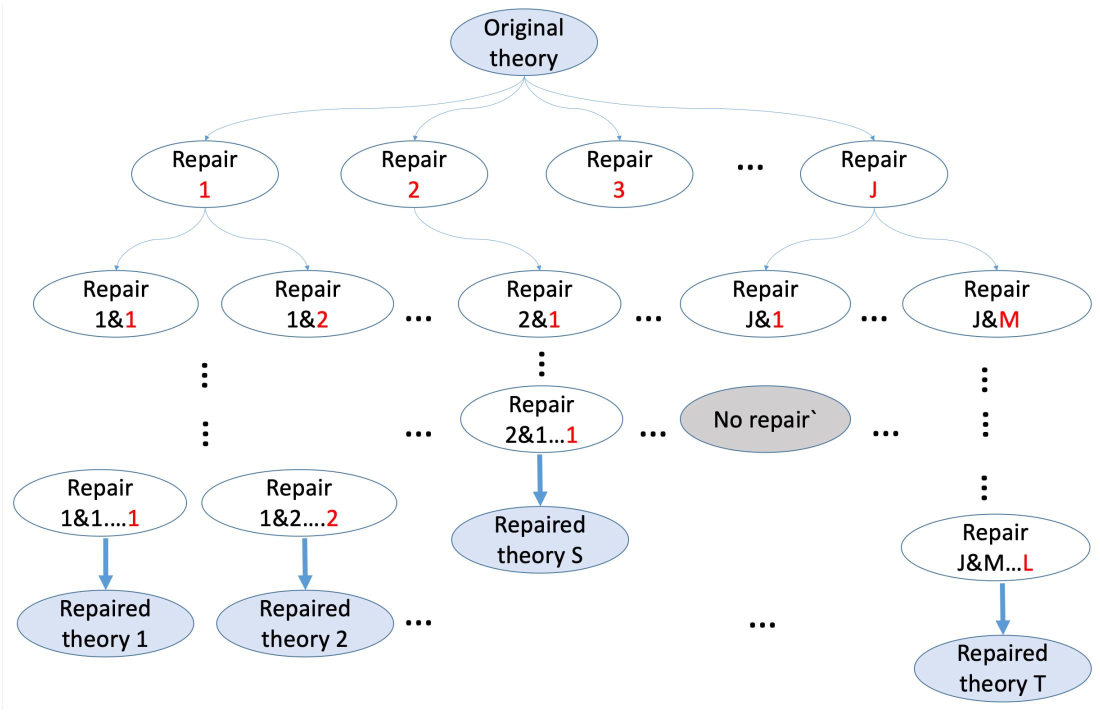


Figure 6.7: Recall The original search space for fault-free theories:*the length of each search branch can be different, and a search branch terminates with failure if there is no repair available to fix a detected fault.*

In summary, the MSCR is defined to merge search branches when their repair plans commute, which reduces the search space of fault-free theories in the ABC Repair System.

6.4 Pruning-out Sub-Optimal MSCRs

Similar to many repair techniques, our combination repair mechanism suffers the common issue of overproducing repairs: ABC suggests dozens of repaired theories for a faulty theory. This issue of overproduction damages the quality of ABC's output.

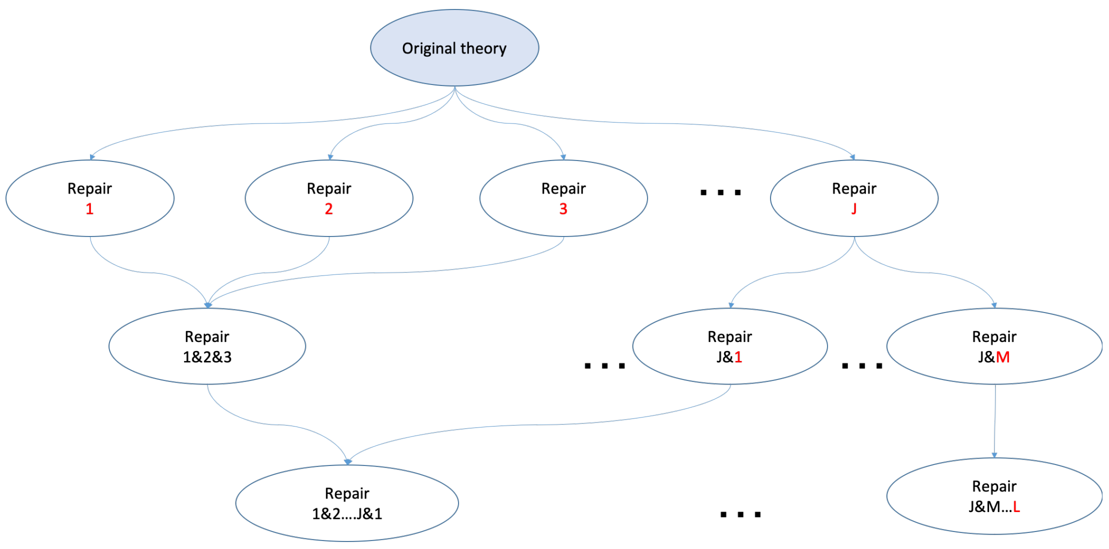


Figure 6.8: The reduced search space of grouping commutative RPs.

The bird theory in Example 6.4.1 is used to illustrate the overproduction of repairs, in which $\mathbb{PS}$ is formalised by a user's observation. This theory is incompatible because that its theorem $fly(tweety)$ is in the false set of $\mathbb{PS}$, whose proof is in 6.52. Since there is only one fault, the MSCRs of repairs are singleton sets. We will discuss these repairs directly for this example.

Example 6.4.1. *Bird Theory.*

$$bird(X) \implies fly(X) \qquad (A1)$$
$$bird(X) \implies feathered(X) \qquad (A2)$$
$$penguin(Y) \implies bird(Y) \qquad (A3)$$
$$\implies penguin(tweety) \qquad (A4)$$
$$\implies bird(polly) \qquad (A5)$$

$\mathcal{T}(\mathbb{PS}) = \{feathered(tweety), fly(polly)\}$
$\mathcal{F}(\mathbb{PS}) = \{fly(tweety)\}$

$$\dfrac{\dfrac{\dfrac{fly(tweety)\implies}{bird(tweety)\implies}\; bird(X)\implies fly(X)}{penguin(tweety)\implies}\; penguin(Y)\implies bird(Y)}{\implies}\; \implies penguin(tweety) \tag{6.52}$$

Abbr.	Repairs to the Input Axiom	ResStep 1	ResStep 2	ResStep 3
Ref1.	Rename its head predicate.	0	1	1
Ref2.	Increase the arity of its head predicate.	0	1	1
Ref3.	Change its variable into another constant.	1	1	0
BR1.	Delete the input axiom.	1	1	1
BR2.	Add unprovable precondition to the rule.	1	1	0

Table 6.2: The repairs of the bird theory in Example 6.4.1: *1 represents that the repair in that row is applicable to the input axiom in the resolution step (ResStep) in that column, while 0 represents that it is inapplicable. The red repair is the optimal one.*

Example 6.4.2. *Semi-Repaired Bird Theory $\mathbb{T}_{12}$: rename the head predicate of the input rule in ResStep 2.*

$$\begin{aligned} bird(X) &\Longrightarrow fly(X) && \text{(A1)}\\ bird(X) &\Longrightarrow feathered(X) && \text{(A2)}\\ penguin(Y) &\Longrightarrow birdDummy(Y) && \text{(A3')}\\ &\Longrightarrow penguin(tweety) && \text{(A4)}\\ &\Longrightarrow bird(polly) && \text{(A5)} \end{aligned}$$

$\mathcal{T}(\mathbb{PS}) = \{feathered(tweety), fly(polly)\}$
$\mathcal{F}(\mathbb{PS}) = \{fly(tweety)\}$

To repair the incompatibility, its proof, shown in 6.52, needs to be blocked by breaking any of its resolution steps. Table 6.2 lists all of the repairs w.r.t. this incompatibility suggested by the ABC system, among which the first three repairs are generated by reformation and the last two by belief revision. The head predicates of the first and the second resolution step are $fly/1$ and $bird/1$ respectively. Because the predicates that occur in $\mathbb{PS}$ are not allowed to change, so that the first and the second repairs in Table 6.2 cannot be applied to the first resolution step (ResStep1). On the other hand, since there are neither variables nor rules involved in the last resolution step, repairs Ref3. and BR2. are not applicable to the last resolution step (ResStep3).

The resulting theory of applying the first repair on the second resolution step is Bird Theory 12, given in Example 6.4.2 with the changed predicate highlighted in blue. It can be seen that this repair introduces an insufficiency because that $feathered(tweety)$ becomes unprovable.

In total, there are eleven repairs for fixing the incompatibility of $fly(tweety)$, among which, only the red 1 in Table 6.2 refers to a repair which does not introduce any new fault. Therefore, that repair is considered to be better than the others. The corresponding repaired theory is given in Example 6.4.3, where the new argument of predicate $bird/2$ in the third rule is assigned with a unique constant: dummy2. New constants were suggested by the observation that they often distinguish two kinds of the concept represented by their predicate, e.g., dummy1 represents for the normal birds that can fly and dummy2 for abnormal ones that cannot. Here the abnormal kind comes from the input axiom in the now broken resolution step.

Example 6.4.3. *Repaired Bird Theory $\mathbb{T}_{22}$: increase the head predicate's arity of the input rule in ResStep 2.*

$$\begin{aligned} bird(X,dummy1) &\Longrightarrow fly(X) && \text{(A1')}\\ bird(X,Y) &\Longrightarrow feathered(X) && \text{(A2')}\\ penguin(Y) &\Longrightarrow bird(Y,dummy2) && \text{(A3')}\\ &\Longrightarrow penguin(tweety) && \text{(A4)}\\ &\Longrightarrow bird(polly,dummy1) && \text{(A5')} \end{aligned}$$

$\mathcal{T}(\mathbb{PS}) = \{feathered(tweety), fly(polly)\}$
$\mathcal{F}(\mathbb{PS}) = \{fly(tweety)\}$

The overproduction of the repair system has been illustrated by Table 6.2. Based on the notion of the strict domination given by Definition 6.2.5, repairs which are not strictly dominated by others can be seen as the *optimal ones*.

Let the theories in Example 6.4.2 and 6.4.3 be $\mathbb{T}_{12}$ and $\mathbb{T}_{22}$ respectively. Then their fault sets are as the following. It can be seen that $\mathbb{T}_{22}$ is fault-free.

$$\begin{aligned} \mathcal{IS}(\mathbb{T}_{12},\mathbb{PS}) &= \{feathered(tweety)\}\\ \mathcal{IC}(\mathbb{T}_{12},\mathbb{PS}) = \mathcal{IS}(\mathbb{T}_{22},\mathbb{PS}) &= \mathcal{IC}(\mathbb{T}_{22},\mathbb{PS}) = \varnothing \end{aligned} \tag{6.53}$$

By comparing all repairs in Table 6.2, it can be concluded that $\mathbb{T}_{22}$ is the one produced by applying the only optimal repair plan.

Although the above discussion is based on repair plans rather than MSCRs, the issue of overproduction exists for MSCRs too. MSCRs reduce the search space by merging some of search branches into one branch, but they do not reduce the number of the fully repaired theories suggested by ABC.

In ABC, it is common to set a depth limit for the search of fault-free theories.

With a depth limit, more repaired theories could be suggested after applying MSCRs, because the length of a search branch could be reduced by MSCRs. For example, if it takes five rounds of fault detection and repair generation to produce a fault-free theory $\mathbb{T}_1$ without MSCRs, and all of $\mathbb{T}_1$'s repair plans commute, then $\mathbb{T}_1$ can be produced in one round by applying the MSCR. If the depth limit is three, then $\mathbb{T}_1$ can be produced only when MSCR is applied. Therefore, it is essential to have a mechanism which selects the optimal MSCRs while pruning the sub-optimal ones.

However, the strict domination given by Definition 6.2.5 cannot be directly applied to MSCRs. Because it is unfair to compare the remaining fault number when theory $\mathbb{T}'$ is generated based on a MSCR of three RPs while another $\mathbb{T}''$ is based on a MSCR of only one RP. In this example, if they are the only candidates to compare and have the same number of remaining faults, then both theories would be concluded as the optimal while $\mathbb{T}'$ should not be.

To evaluate the fitness of MSCRs, their estimated cost is defined as the following.

Definition 6.4.1 (Estimated cost of a MSCR)**.** *Let $\mathbb{M}$ be the set of repair plans combined by the MSCR, and $\mathbb{T}_g$ be the repaired theory after applying all RPs in $\mathbb{M}$. Then the estimated cost of the MSCR is $c(\mathbb{M})$:*

$$c(\mathbb{M}) = |\mathbb{M}| + \mathcal{N}_{insuff}(\mathbb{T}_g) + \mathcal{N}_{incomp}(\mathbb{T}_g)$$

where $\mathcal{N}_{insuff}(\mathbb{T}_g)$ and $\mathcal{N}_{incomp}(\mathbb{T}_g)$ are the number of inefficiencies and incompatibilities of $\mathbb{T}_g$ respectively.

Accordingly, the revised strict domination w.r.t. MSCRs is defined as the following.

Definition 6.4.2 (Strict Domination on MSCRs)**.** *A MSCR M_1 is strictly dominated by another M_2 if $c(\mathbb{M}_1) > c(\mathbb{M}_2)$.*

MSCRs which are not strictly dominated by others are optimal, and then all the sub-optimal ones which will be pruned from the repair process. Consequently, the search space of fault-free theories are reduced. This sub-optimal pruning can be seen as a simplified implementation of the entrenchment of axioms and preconditions by regarding insufficiencies and incompatibilities as equally important.

6.5 Other Specifications and Heuristics

A set of specifications and heuristics are developed developed to assist the ABC Repair System. Some of them have been given in the previous discussion. This section summarises the others that have not been introduced.

Specifications: desired behaviours of the ABC Repair System, especially for avoiding the repair process being broken.

Optional heuristics: restrict the repair generation to avoid undesired repairs.

Specifications are essential to the repair mechanism. They protect the repair mechanism by avoiding serious issues, e.g., loops. In contrast, heuristics are optional, which provide an opportunity of avoiding unwanted repairs from the user side. Consequently, the over production of repairs can be controlled and the rate of desired repairs to the total repairs can be increased. Neither of them invents new repairs, but they help to improve the general performance.

Specifications reflect the basic strategy of our repair framework, among which some prevent a previous repair from being reversed. All specifications are listed below where the reversal repairs are listed with 'vs' between them.

1. **Minimal Change**: do not change more than necessary. For example, only one extra argument will be added to break a unification by increasing the arity of the predicate, no more than that. Otherwise, the repair search space will be exploded and the quality of repairs will be predictably low.

2. **Preservation of $\mathbb{PS}$**: everything in $\mathbb{PS}$ is the most entrenched and is not changed in any case[7].

3. **No duplicates**: Same repaired theory could be generated by applying different combinations of MSCRs. Duplicates check is necessary before output.

4. **Axiom expansion vs deletion**: if an axiom is added by a previous repair, the identical axiom will not be deleted, and vice versa.

5. **Precondition expansion vs deletion**: if an precondition is added by a previous repair, the identical precondition will not be deleted, and vice versa.

[7]$\mathbb{PS}$ comes from the user, repair system cannot change it. Investigating the communication between a user and a repair system via $\mathbb{PS}$ will be a future work, discussed as bullet 6 in §9.1.

6. **Arity increment vs decrement**: if an argument is added by a previous repair, it will not be deleted, and vice versa.

7. **Predicate merging vs splitting**: if a predicate is merged by a previous repair, it will not be split, and vice versa.

The violation of any of the above specifications can be seen as an error of the repair mechanism. Heuristics are summarised below, each of which is followed by the sample reasons why a user would choose it.

1. **noAxiomDele**: no axiom will be deleted. It is desired when all of the axioms in the theory are considered to be useful, and then none of them should be deleted. Here belief revision is banned, e.g., in Example 6.4.1.

2. **noRuleDele**: no rule will be deleted, which is a special case of noAxiomDele. It can be chosen when the rules of a theory are essential, e.g., the rules of playing a game.

3. **noAxiomAdd**: no axiom will be added. It can be chosen when the size of a theory is limited or the current axioms are the target to modify. Here abduction is banned.

4. **noRuleAdd**: no rule will be added. If the theory will be used in a platform where only assertions are allowed, the user should employ this heuristic.

5. **protection list**: any element that is in the protection list will not be changed. Candidates of the elements include whole axioms, predicate names, the arity of a predicate, and constants. This list is different from the entrenchment evaluation. Protection list is specified by the user while the entrenchment evaluation is automatic based on the preferred structure. For example, when a predicate is isolated from the preferred structure, its entrenchment score will be low. However, if the user knows that the predicate should not be changed, e.g., ‘=’, then it can be given in the protection list.

6. **minArgDele**: when deleting arguments of a predicate, delete the ones whose domain is minimal in the size. The underlying idea is that the bigger the argument domain is, the more information that argument can provide.

When multiple optional heuristics are employed, the condition that dominates the rest of them will be applied, e.g., if both noRuleAdd and noAxiomAdd are employed, then noAxiomAdd will be the one applied.

Specifications are built-in by default with the repair mechanism, while a user can choose from optional heuristics according to what kinds of repairs or repaired theories are needed.

6.6 Summary

To achieve a succinct representation of inequalities, a mechanism of unique name assumption with exceptions has been developed, which makes it possible to describe how pairs of syntactically distinct constants are equal or unequal in the context of Datalog. UNAE retains the original UNA's advantage of having the inequalities by default for a simple representation. On the other hand, UNAE successfully avoids UNA's deficiency of refusing the fact that one individual can have two distinct names.

The traditional epistemic entrenchment proposes postulates but lacks a formalisation which can quantify the evaluation of it. Also, it only discusses axioms and not the signature and its elements. Based on $\mathbb{PS}$, we formalised the algorithms of evaluating entrenchment from the level of both axioms and the signature:

1. The entrenchment of an axiom is formalised based on how much it respects its $\mathbb{PS}$.
2. The entrenchment of a precondition is evaluated based on how big influence it has w.r.t. making the theory respect its $\mathbb{PS}$.
3. The entrenchment of the signature is formalised for the following elements:
    (a) Predicate names, whose entrenchment is calculated based on their distance to preferred predicates on a defined theory graph;
    (b) An argument of a predicate, whose entrenchment is evaluated based on its influence on making the theory respect its $\mathbb{PS}$.

Since there can be commutative RPs when the theory has multiple faults, a maximal set of commutative repair plans is defined and computed to apply RPs together. Accordingly, the sub-optimal pruning of MSCRs is developed based on their estimates cost.

A set of specifications and heuristics are developed: the former addresses basic issues so they are built by default, while the latter can be chosen by users to fit different requirements. They are important to make the repair generation effective and efficient.

Chapter 7

Implementation

The implementation is written in the logic programming language: Prolog. There are several reasons behind the choice. 1, Prolog is a logical programming language with roots in first-order logic which is sufficient for our project. 2, Prolog's free unification algorithm and depth-first search can significantly simplify our code. Although Prolog lacks the unification occurs check, it is not a problem in a Datalog theory, because Datalog has no functions so the occurs check is not needed. On the other hand, all searches in our project are exhaustive and finite, thus depth-first search does not suffer from non-termination. 3, Prolog is declarative so that we only need to express the logic of our computation, which makes our implementation more understandable than one written in non-declarative languages. In summary, Prolog allows a sound and complete implementation for our project which is succinct and easily readable.

In our implementation, an axiom of the input Datalog theory is written as a Horn clause and each Horn clause is given as the argument of the predicate $axiom/1$, e.g., there are three axioms in the motherhood theory below. A Horn clause is a list of literals among which the single positive one is its head and the negative ones constitute its body.

We use predicate '=' to represent the equality and $\neq$ inequality, while '$\equiv$' represents syntactically equal, whose negation is $\not\equiv$. The predicate $heuristics/1$ in the following Motherhood Example is for listing applied heuristics, e.g., 'noAxiomDele' means that the repair of deleting an axiom is banned.

Motherhood Example.

```
axiom([+mum(diana, william)])
axiom([+mum(camilla, william)])
axiom([-mum(X, Z), -mum(Y, Z), +(=(X, Y))])
heuristics([noAxiomDele]).
```

This chapter introduces the implementation of the work proposed in Chapter 4 and 6 first. And then the second half of this chapter discusses how we interact with the user and what heuristics have been built to boost the repairing performance.

7.1 Unique Name Assumption with Exceptions

Based on the UNAE, constants in one subset of a theory's equality set, $\mathbb{ES}$ in Definition 6.1.1, are equal, while ones in different subsets are unequal. This section discuss the implementation of UNAE from the aspects of an efficient representation of the subsets of the equality set, its calculation and the relevant inference processes.

Recall Example 6.1.1, the equality set can be derived from the axioms in a Datalog theory. Before deriving $\mathbb{ES}$, the properties of '=' need to be considered, which are listed below.

Reflexivity: $\forall x.\, x = x.$

Symmetry: $\forall x, y.\, x = y \implies y = x.$

Transitivity: $\forall x, y, z.\, x = y \wedge y = z \implies x = z.$

Monotonicity: $\forall \vec{x}, \vec{y}.\, \vec{x} = \vec{y} \implies [p(\vec{x}) \iff p(\vec{y})]$, where $\vec{x}$ is a vector of n variables. Here p is a predicate, because functions are not allowed in a Datalog signature.

Because of the monotonicity, the variable equality in a precondition of a rule axiom can be removed following the rules defined below. This removal needs to be done before any goal-proving task as it may influence the result of a derivation. Here $\uplus$ is *disjoint union*, i.e., $x \uplus y = x \cup y$ where $x \cap y = \varnothing$;

Definition 7.1.1 (Variable Equality Precondition Removal Rules)**.** *When precondition $v = t$ or $t = v$ is in a rule, it is removed by replacing all of the occurrences of v by t in that rule ($v \mapsto t$). Two rules are required to cover the symmetry of* $=$.

$$\frac{\mathbb{T} \uplus \{\bigwedge_{k=1}^{i-1} R_k \wedge v = t \wedge \bigwedge_{k=i+1}^{n} R_k \implies P\}}{\mathbb{T} \uplus \{(\bigwedge_{k=1}^{i-1} R_k \wedge \bigwedge_{k=i+1}^{n} R_k \implies P)\{v \mapsto t\}\}}\ VE_1$$

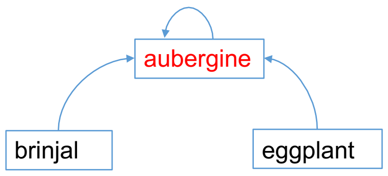


Figure 7.1: The tree structure of the subset of an equality set: $\mathbb{E} = \{aubergine, brinjal, eggplant\}$, where the representative 'aubergine' is highlighted.

$$\frac{\mathbb{T} \uplus \{\bigwedge_{k=1}^{i-1} R_k \wedge t = v \wedge \bigwedge_{k=i+1}^{n} R_k \implies P\} \qquad \mathbb{ES}}{\mathbb{T} \uplus \{(\bigwedge_{k=1}^{i-1} R_k \wedge \bigwedge_{k=i+1}^{n} R_k \implies P)\{v \mapsto t\}\} \qquad \mathbb{ES}} \; VE_2$$

Now the equality theorems can be derived from the resulting theory $\mathbb{T}'$ of variable equality preconditions removal[1]. Apply an exhaustive search with the goal clause $x = y \implies$ to $\mathbb{T}'$ and get all of ground answers which are in the form $c_i = c_j$. Based on the symmetry property, reorder these equalities so that c_i is alphabetically before c_j, after which duplicates are eliminated and then all the remaining equalities are ordered according to their first constant in lexicographical order. The resulting set of ordered equalities will be written as $\mathbb{E}_{ordered}$ in the following discussion.

Recall the properties of '='. They would need to be applied to the inference in which an equality is involved, and the transitivity and the monotonicity can make the inference quite difficult, sometimes even non-terminating. Therefore, a good representation of each subset of $\mathbb{ES}$ is needed for easing the inference with equalities. Thus, the equivalence class is defined, which organises $\mathbb{ES}$ by assigning each subset with a representative.

Definition 7.1.2 (Equivalence Class). *An equivalence class $\mathbb{E}_c$ summarises all the subsets of the equality set by enriching each subset $\mathbb{E}$ as a tree structure $\mathcal{T} =< r, \vec{\mathbb{A}}, \mathbb{E} >$, in which each node corresponds to one constant in $\mathbb{E}$, among which the root node in is its representative (r); arcs are arrows $\vec{\mathbb{A}}$, each of which points from a node to the root:*

$$\mathbb{E}_c = \{\mathcal{T} | \forall \mathbb{E}, \mathcal{T} =< r, \vec{\mathbb{A}}, \mathbb{E} >\}, where\ r \in \mathbb{E}; \forall \vec{a} \in \vec{\mathbb{A}}, \vec{a} = (c, r), c \in \mathbb{E} \tag{7.1}$$

According to Definition 7.1.2, the representative has its arrow pointing back to itself. An example of a equality set in tree structure is given in Figure 7.1, where *aubergine* is the representative.

[1]Here we are not aiming at the theorems inferred from equality's properties, but the ones directly derived by axioms in the theory.

It can be seen that given a representative, and a subset of the equality set, the arrows are uniquely decided, so the corresponding equality tree is.

$$\text{If } \mathcal{T}_1 =< r, \vec{\mathbb{A}}_1, \mathbb{E} > \bigwedge \mathcal{T}_2 =< r, \vec{\mathbb{A}}_2, \mathbb{E} >, \text{ then } \vec{\mathbb{A}}_1 = \vec{\mathbb{A}}_2 \wedge \mathcal{T}_1 = \mathcal{T}_2 \tag{7.2}$$

An equivalence class $\mathbb{E}_c$ can be calculated based on $\mathbb{E}_{ordered}$, given by Definition 7.1.3, where $\mathcal{N}$ is a function which returns the representative of a constant, e.g., $\mathcal{N}(eggplant) = aubergine$ based on the equality tree in Figure 7.1.

Definition 7.1.3 (Equivalence Classes Calculation)**.** *The equivalence class $\mathbb{E}_c$ is calculated based on the set of sorted equalities $\mathbb{E}_{ordered}$ following the processes below, where if $c_1 = c_2 \in \mathbb{E}_{ordered}$, then c_1 is lexicographic smaller than c_2.*

1. *Initialisation: initialise $\mathbb{E}_c$ by establishing a single-node tree $\mathcal{T}_i =< c_i, \vec{A}_i, \{c_i\} >$ for each constant c_i in the signature. Here $\mathcal{C}$ is the set of all constants in the signature of $\mathbb{T}$, $\vec{A}_i$ is a singleton set whose member $\vec{a_{ii}}$ is a vector whose arrow points from c_i back to itself.*

$$\mathbb{E}_c = \{\mathcal{T}_i | \forall c_i \in \mathcal{C},\ \mathcal{T}_i =< c_i, \vec{A}_i, \{c_i\} >\},\ \ where\ \vec{A}_i = \{\vec{a_{ii}}\},\ \vec{a_{ii}} = (c_i,\ c_i) \tag{7.3}$$

2. *Update: get the updated equivalence class from $\mathbb{E}_c$ to $\mathbb{E}_c'$ for each equality $c_1 = c_2 \in \mathbb{E}_{ordered}$. No changes are made in the update when $c1$ is already under the same representative with $c2$. Otherwise, merge the original equivalence classes of c_1 and c_2 into a new equivalence class whose representative is c_1's original representative.*

$$\mathbb{E}_c' = \begin{cases} \mathbb{E}_c,\ iff\, \mathcal{N}(c_1) = \mathcal{N}(c_2) \\ \mathbb{E}_c - \{< c_1, \vec{\mathbb{A}}_{c1}, \mathbb{E}_{c1} >,\ < c_2, \vec{\mathbb{A}}_{c2}, \mathbb{E}_{c2} >\} \uplus \{< c_1, \vec{\mathbb{A}}_{c12}, \mathbb{E}_{c1} \cup \mathbb{E}_{c2} >\} \end{cases} \tag{7.4}$$

Where $\vec{\mathbb{A}}_{c12} = \vec{\mathbb{A}}_{c1} \cup \{(c_i,\ c_{1rep}) | \forall c_i,\ (c_i,\ c_2) \in \vec{\mathbb{A}}_{c2},\ (c_1, c_{1rep}) \in \vec{\mathbb{A}}_{c1}\}$.

3. *Termination: when all equalities in $\mathbb{E}_{ordered}$ have been incorporated in $\mathbb{E}_c$, this process is done.*

To avoid the tediousness of equality's transitivity and monotonicity in inference, a normalisation of constants is pre-processed based on $\mathbb{E}_c$: all constants that occur in $\mathbb{E}_c$ will be replaced by their representatives in the theory $\mathbb{T}$ and the goal clause $\mathbb{G}$.

Definition 7.1.4 (Replacement of Constants with their Representatives)**.** *Let* $\sigma = \{c_1 \mapsto r'_1, \ldots c_n \mapsto r'_n\}$*, where* $\mathcal{N}(c_i) = r_i \in \mathbb{E}_c$*, then the replacement is:*

$$\frac{\mathbb{G} \quad\quad \mathbb{T} \quad\quad \mathbb{E}_c}{\mathbb{G}\sigma \quad\quad \mathbb{T}\sigma \quad\quad \mathbb{E}_c}\ R$$

Note that we have abused the notation for substitution by using it for replacement.

The correctness of this rule is justified by the monotonicity of =. By replacing all the constants by their representatives, the original 'equal' is translated to the 'identical'. Thus the properties of equality are not needed, except for reflexivity, during the following inference. The condition of this translation is that the equivalence classes calculation in Definition 7.1.3 and the replacement of constants in Definition 7.1.4 are pre-processed which should be done before any goal proving task and need to be updated when the theory or the goals are changed.

So far, the whole pre-process for UNAE is done and it is ready to undertake goal-proving tasks. When there is predicate = or ≠ in the subgoal, it can be resolved based on the two new inference rules defined in Definition 7.1.5 to supplement SL Resolution to remove ground equalities and ground inequalities. Notice that our resolution always selects the left most subgoal.

Definition 7.1.5 (UNAE Inference Rule.)**.** *After replacing all constants with their representative base on equivalence class, the subgoal* $c = c \implies$ *or* $c_1 \neq c_2 \implies$ *is resolved straight away, where* c_1 *and* c_2 *are syntactically distinct constants.*

$$\frac{c = c \implies \wedge\ \mathcal{G}}{\mathcal{G}} \tag{7.5}$$

$$\frac{c_1 \neq c_2 \implies \wedge\ \mathcal{G}}{\mathcal{G}} \tag{7.6}$$

Again, the above rules rely on the up-to-dated equivalence class, the theory and goals. Otherwise, rule 7.6 will be applied incorrectly.

7.2 Fault Detection

By fault detection, we mean detecting an input theory's insufficiency, incompatibility based on a preferred structure[2]. The main flow of fault detection is illustrated by the

[2]Relevant theoretical discussion is in §4.2 and 4.3.

pseudocode in Algorithm 1.

```
Algorithm 1: Fault Detection based on SL Resolution
  Result: Proofs or the insufficiency evidences.
  Input  : Input_theory, Preferred_structure
  if Preferred_structure is Empty then
      Insuff_set = Incomp_set = Proof_sets = Evidence_sets = Empty
      Terminate.
  else
      while Get the next Goal from the True Set in Preferred_structure do
          /* get all the proofs and the evidence of partial
             proofs.                                              */
          (ProofsInsuff, Evidences) = sl_Resolution(GoalInsuff, Input_theory).
          if Proof_Set is Empty. then
              Insuff_set = Insuff_set + GoalInsuff
              Evidence_sets = Evidence_sets + Evidences
              Continue.
          else
              Not a fault
              Proof_sets = Proof_sets + ProofsInsuff
              Continue.
          end
      end
      while Get the next Goal from the False Set in Preferred_structure do
          /* get all the proofs ignoring the partial proofs.      */
          (ProofsIncomp, _) = sl_Resolution(GoalIncomp, Input_theory).
          if Proof_Set is Empty. then
              Not a fault.
              Continue.
          else
              Incomp_set = Incomp_set + GoalIncomp
              Proof_sets = Proof_sets + ProofsIncomp
              Continue.
          end
      end
  end
```

In this section, Horn clauses are written in their disjunction forms sometimes.

- The disjunction form of the rule $P_1 \wedge \ldots \wedge P_n \implies Q$ is $-P_1 \vee \ldots \vee -P_n \vee +Q$.

- The disjunction form of the assertion $\Longrightarrow Q$ is $+Q$.

The rest of this section discusses the implementation of SL-Resolution in §7.2.1 and the proof representation in §7.2.2.

7.2.1 Selected Literal Resolution without Ancestor Resolution

Although Prolog has free resolution, we need to implement a resolution algorithm because: 1, we need the proof for generating repairs. 2, when repairing an insufficiency, we want to unblock a failed proof. Thus, unlike the common proof searching, we need the evidence of a failed proof as well as a complete proof. 3, we need to embed the new inference rules developed for UNAE.

In this thesis, SL Resolution is used to search for proofs of a goal. It works by refutation that the conjecture to be proved is negated and added to the axioms. If the empty clause is derived, then that conjecture is proved.

In this project, the conjecture comes from the preferred structure, which is for fault detection. As discussed in §4.3, to detect an insufficiency or an incompatibility, the original goal is always a negated ground proposition. Additionally, the input clauses are only Horn clauses, for which SL-resolution without ancestor resolution is complete.

Because we need all of the proofs and all of the evidence of incomplete proofs to support the repair generation, the Prolog search operator $findall$ is employed, whose search strategy is depth-first with chronological backtracking. In other words, the 'OR' choice in our resolution (Bundy, 1985) is made by iterating over all axioms. As for one search branch, which is the 'AND' choice, it is fixed by choosing the left most subgoal to resolve away. This fixed choice makes it easier for backtracking a proof than randomly choosing a subgoal to resolve.

Table 7.1 depicts the resolution algorithm in terms of AND choice: a search branch terminates with either a proof or an evidence of partial proof of the original goal. If a variable name occurs both in the selected input clause and the goal clause, the ones in the input clause will be renamed to avoid name collisions. For simplification, that variable renaming is not described in the table.

In our thesis, the input of SL Resolution includes a goal and the input theory, between which only the goal together with its derivation D is given in the first column in Table 7.1. The second column depicts different forms of the first, left most subgoal, and gives the condition of the resolution step. The condition is highlighted in blue, which requires that the theory has an axiom in the form of $(+P(\vec{t}) \vee -\vec{B}, S)$, whose head

Before	Condition	Check	After
$([],D)$	$--$	$--$	TER. Success. Proof =([], D).
$([G_1 \vee G_2 ... \vee G_n],D)$	$G_1 = -[c = c] \ \vee -[c_1 \neq c_2]$	$--$	Continue with $(GoalNew, DNew)$, where $GoalNew = \mu[G_2 ... \vee G_n]$, $DNew = ([G_1 ... \vee G_n], D, ([], unae))$.
	$G_1 = -[c_1 = c_2] \ \vee -[c \neq c]$	$--$	TER. Fail. $Evidence = ([G_1 \vee G_2 ... \vee G_n], D, [])$.
	$G_1 = -P(\vec{s})$, $\exists(+P(\vec{t}) \vee -\vec{B}, S) \in \mathbb{T}$, $P(\vec{s}) \equiv P(\vec{t}) \wedge \mu, \mu = \{\vec{s}/\vec{t}\}$	$\forall \vec{G}' \in D$, $\vec{G}'$ is a goal clause, $\vec{G}' \notin \mu[-\vec{B} \vee G_2 ... \vee G_n]$	Continue with $(GoalNew, DNew)$, where $GoalNew = \mu[-\vec{B} \vee G_2 ... \vee G_n]$, $DNew = ([G_1 ... \vee G_n], D, (+P(\vec{t}) \vee -\vec{B}, S))$.
		Otherwise, fail	Retry.
	Otherwise, fail	$--$	TER. Fail. $Evidence = ([G_1 \vee G_2 ... \vee G_n], D, [])$.

Table 7.1: The algorithm for SL-resolution: *The first column is the pair of the goal G and its derivation D: (G, D); The second column gives the condition of the derivation from 'Before' to 'After'; The third column is a check which fails the bad derivation which results in a new goal that contains a previous goal.*

can resolve with the goal. The third column is a check step in our resolution. If the resulting new goal clause is a superset of a previous goal clause, then this resolution step fails the check because it either makes the goal clause more complicated than before or reproduces a previous goal. In this case, the derivation reaches 'Retry' in the last column, and it will redo the resolution step by getting another axiom from the theory for resolution, which has been highlighted in blue in the table. In 'After' column, there are three different colours: the red ones terminates with a proof or an evidence; the blue one, as aforementioned, retrys another axiom to resolve the current subgoal $G_1 = -P(\vec{s})$; and the black 'continue' which updates the goal and the derivation and use them as the input of the next step of resolution.

7.2.2 Proof Representation

The proof and the evidence of partial proofs of a goal found by SL Resolution need to be recorded in a well formed representation, because they are the basics of repair generation. This section gives the format of a proof or the evidence of a partial proof. Our discussion will be based on evidence in the following.

An evidence ($\mathbb{P}_n$) is constituted by a sequence of resolution steps (RS), shown in 7.7, and each RS ($\mathbb{R}$) is represented by a five-tuple, shown in 7.8.

$$\mathbb{P}_n = [\mathbb{R}_1, \mathbb{R}_2, ..\mathbb{R}_n] \tag{7.7}$$

$$\mathbb{R} = (G_0,\ Inc,\ \gamma,\ G_1,\ (N,\ M)) \tag{7.8}$$

where some elements are local, which will not be changed along with its following RSs in the sequence; while some are global which updates when a new RS occurs.

- G_0 is local, representing the current goal clause in the form of $g_1(\vec{c_1}) \wedge g_1(\vec{c_2}) \wedge ...g_n(\vec{c_n}) \implies$, whose first, left-most subgoal $g_1(\vec{c_1})$ is resolved in this RS;
- Inc is local, representing the input clause, $Inc \in \mathbb{T}$, in the form of $p_1(\vec{X_1}) \wedge p_1(\vec{X_2}) \wedge ...p_m(\vec{X_n}) \implies q(\vec{Y})$, whose head $q(\vec{Y})$ resolves $g_1(\vec{c_1})$ is resolved in this RS.
- γ is global, which records the substitutions that have been applied on the input clause Inc. It can be expanded during the following sequence of RSs. The initialisation of γ is by the equation below.

$$g_1(\vec{c_1}) \equiv q(\vec{Y}) \cdot \gamma \tag{7.9}$$

- G_1 is local, representing the resulting goal clause in this RS, which will be the current goal for the next RS.

$$G_1 = (p_1(\vec{X_2}) \wedge ... p_m(\vec{X_n}) \wedge g_1(\vec{c_2}) \wedge ... g_n(\vec{c_n})) \cdot \gamma \Longrightarrow \tag{7.10}$$

- (N, M) is global, whose sum represents how many subgoals in the newest goal clause are originally from *Inc* in this RS. The initialisation of N is the size of the body of *Inc*, and M is zero. It can be seen that $|G_1| - |G_0| + 1 = N$, where 1 is caused by the resolved $g_1(\vec{c_1})$. M works on recording the reorder of subgoals when the goal is a mix of equalities/inequalities and non-equalities, which is discussed latter. As long as $N + M > 0$, substitutions need to be added to γ in this RS tuple.

G_0 in $\mathbb{R}_n$ can be different from G_1 in $\mathbb{R}_{n+1}$ when the order of the subgoals is changed. To reduce the search space, the subgoals with "=" or "≠" are resolved after all the non-equality subgoals, so that all variables in the remaining subgoals have be substituted if they can. Reordering reduces the search space especially when a variable occurs in the inequality subgoal, e.g., $c \neq X$, where X can be bound with all constants that are unequal to c. By moving equalities and inequalities to the end, subgoals like $c \neq X$ can be avoided when possible.

Based on the RS tuples, the operations in evidence $\mathbb{P}_n$ can be recorded by updating its RS tuples. These operations include reordering the subgoals and resolving the first subgoal, given as the following, where the existing evidence is $\mathbb{P}_{n-1} = [\mathbb{R}_1, \mathbb{R}_2, ... \mathbb{R}_{n-1}]$, and the current goal clause is G_n.

1. Resolution of non-equalities: when RS occurs, the first non-equality subgoal in G_n is resolved away, then the evidence needs to make the following changes:

   (a) update the existing $\mathbb{P}_{n-1}$ to $\mathbb{P}'_{n-1}$ by decreasing the last non-zero N in $\mathbb{P}_{n-1}$ by 1, representing the remaining subgoals from that input clause is 1 less.

   (b) append the updated existing evidence $\mathbb{P}'_{n-1}$ with a new RS tuple T, where G_n is the current goal clause from $\mathbb{P}_{n-1}$; Inc_n is the input clause which resolves the first sub-goal of G_n; γ_n is the applied substitution; G_{n+1} is the resulting new goal clause; N_n is the number of the newly introduced new sub-goals from Inc_n and M_n is zero.

$$\mathbb{P}_n = \mathbb{P}'_{n-1} + T \tag{7.11}$$

$$\mathbb{P}'_{n-1} = \mathbb{P}_{n-1} \ominus_N 1 \tag{7.12}$$

$$T = (G_n, Inc_n, \gamma_n, G_{n+1}, (N_n, M_n)) \tag{7.13}$$

where $\ominus_N$ takes the last[3] tuple $\mathbb{R}_i$ in the input list in which N is not zero, and then decreases N in $\mathbb{R}_i$ by 1.

(c) append the substitution to RS tuples whose numbers satisfies $N + M > 0$. If $N + M = 0$, e.g., $(G, Inc, \gamma, G', (0,0)) \in \mathbb{P}_{n-1}$, it means that none of the current subgoals are from Inc[4]. Thus the current substitution γ_n in 7.13 should not be applied to Inc. Alternatively, there are remaining subgoals from the input clause, so that new substitution is applied to that input clause.

2. Reorder: if $\exists (G_i, Inc_i, \gamma_i, G_{i+1}, (N_i, M_i)) \in \mathbb{P}_{n-1}$, $N_i \neq 0$, and the first subgoal is of the equality/inequality, then move that first subgoal to the end of the goal clause. Update the evidence by decreasing the last non-zero N in $\mathbb{P}_{n-1}$ by 1, and increasing its M by 1. Then the sum of all Ms in an evidence is the total number of the moved equality/inequality subgoals.

$$\mathbb{P}'_{n-1} = \mathbb{P}_{n-1} \ominus_N 1 \oplus_M 1 \tag{7.14}$$

where $\oplus_M$ takes the same tuple $\mathbb{R}_i$ which $\ominus_N$ operates, and then increase that tuple's M by 1.

3. Resolve the equality/inequality subgoals: if $\forall (G_i, Inc_i, \gamma_i, G_{i+1}, (N_i, M_i)) \in \mathbb{P}_{n-1}$, $N_i = 0$, then there are only equality or inequality subgoals in G_n. Resolve the first subgoal away, and the evidence needs to be updated by making the following changes:

(a) update the existing $\mathbb{P}_{n-1}$ to $\mathbb{P}'_{n-1}$ by decreasing the first non-zero M in $\mathbb{P}_{n-1}$ by 1.

(b) append the updated existing evidence $\mathbb{P}'_{n-1}$ with a new RS tuple T. If all of the newly introduced new sub-goals are of equality/inequality, then M_n

[3]The leftmost subgoal is one latest introduced, so that it is from the last input clause not the first.

[4]It happens when *Inc* is an assertion which does not introduce any subgoal, or *Inc* is a rule whose preconditions had been introduced as subgoals by *Inc*, but those subgoals have been resolved away in previous RSs.

is the number of them and $N_n = 0$. Alternatively, if new sub-goals contain non-equalities, then N_n is the number of them and $M_n = 0$.

$$\mathbb{P}_n = \mathbb{P}'_{n-1} + T \tag{7.15}$$

$$\mathbb{P}'_{n-1} = \mathbb{P}_{n-1} \ominus_M 1 \tag{7.16}$$

$$T = (G_n, Inc_n, \gamma_n, G_{n+1}, (N_n, M_n)) \tag{7.17}$$

where $\ominus_M$ takes the first[5] tuple $\mathbb{R}_i$ in the input list in which M is not zero, and then decrease M in $\mathbb{R}_i$ by 1.

(c) append the substitution to RS tuples whose numbers satisfies $N + M > 0$.

Theorem 7.2.1. *The evidence $\mathbb{P}_k = [\mathbb{R}_1, \mathbb{R}_2, ..., \mathbb{R}_k]$ recorded following the above operations has the below property, where $\mathbb{R}_i = (G_{i-1}, Inc_i, \gamma_i, G_i, (N_i, M_i))$, $1 \le i \le k$.*

$$|G_i| = |G_0| - 1 + \sum_{i=1}^{i=k} N_i + M_i \tag{7.18}$$

Proof. Because the reordering does not change the number of the subgoals, neither the sum of $N_i + M_i$, thus it is omitted in the proof.

Let i=1, if the first subgoal is not of equality, then $M = 0$, and the new goal is the original ones together with the body of the input clause without the one being resolved away, therefore the number of the subgoals in the new goal $|G_1|$ is as the following, which is the same as i=1 in 7.18.

$$|G_1| = M - 1 + N_1 + |G_0| - 1 \tag{7.19}$$

Let $i = j,\ j < k$, and assume that $|G_j| = |G_0| - 1 + \sum_{i=1}^{i=j} N_j + M_j$, then when a new RS occurs, one subgoal resolved away ($|G_j| - 1$) and new subgoals introduced by the input theory $N_{j+1} + M_{j+1}$.

$$|G_{j+1}| = |G_j| - 1 + N_{j+1} + M_{j+1} \tag{7.20}$$

$$= |G_0| - 1 + \sum_{i=1}^{i=j} N_j + M_j - 1 + N_{j+1} + M_{j+1} \tag{7.21}$$

Because the existing evidence has been updated by either 7.12 or 7.16. Thus the updated sum of $\mathbb{P}'_j$ is decreased by 1.

$$\sum_{i=1}^{i=j} N'_j + M'_j = \sum_{i=1}^{i=j} N_j + M_j - 1 \tag{7.22}$$

[5]Due to the reorder of equality/inequality subgoals, the leftmost equality/inequality subgoal is one oldest introduced, so that it is from the first input clause not the last.

Apply the updated evidence to 7.21. It can be seen that 7.18 is true for $j+1$.

$$\begin{aligned}
|G_{j+1}| &= |G_0| - 1 + (\sum_{i=1}^{i=j} N_j + M_j - 1) + N_{j+1} + M_{j+1} && (7.23)\\
&= |G_0| - 1 + \sum_{i=1}^{i=j} N'_j + M'_j + N_{j+1} + M_{j+1} && (7.24)\\
&= |G_0| - 1 + \sum_{i=1+1}^{i=j+1} N_j + M_j && (7.25)
\end{aligned}$$

In conclusion, the numbers in the evidence have the property written in 7.18.

□

The advantage of the defined representation are summarised as the following.

1. Given a proof, it is easy to get all input clauses involved in the proof.
2. Easy to trace-back. Specify one particular RS, $\mathbb{R}_k$, the input clause which introduced the *ith* remaining subgoal in $\mathbb{R}_k$ can be found by counting Ns and Ms.
3. Easy to know the substitutions applied to the input clause in the proof. It is γ given as the third element in each RS tuple.

For a proof, the fourth element in its last RS tuple is an empty clause. For a partial proof, the fourth element in its last RS tuple is the remaining goal clause that cannot be resolved.

7.3 Repair Generation

This section discusses the implementation of repair generation based on the repair algorithm given by Chapter 5 by incorporating the mechanism of pruning sub-optimal repairs in §6.4 and the heuristics in §6.5.

The proofs of propositions in the false set of the preferred structure are unwanted (P1), while the proofs of propositions in the true set are wanted (P2). On the other hand, the evidence of partial proofs of the propositions in the true set are potentially wanted (E1). The repairs of an incompatibility block P1 and the repairs of the insufficiency either unblock one of its partial proof in E1 or imitate a template proof in P2.

All of P1, P2 and E1 can be found by the Linear Resolution with Selection Function discussed in §7.2.1. These proofs and evidence together with the input theory and

Before	Condition	Repair Plan	After
(-G, prefStruc)	$--$	$--$	Termination.
([],$\mathbb{D}$)	$--$	$--$	Continue with $\mathbb{D}$.
$((-G_1 \vee -\vec{G}^n), \mathbb{D}, (A, S))$, where $G_1 = p(\vec{s}^m)$ $A = +p(\vec{t}^m) \vee -\vec{B}$	$p \notin Protected$ $S \neq abduction$	$rename(A, p)$	
	$G_1 \notin Protected$	$Arity\_inc(A, p)$	
	$t_i \in \vec{t}^m$ $s_i \in \vec{s}^m$ $t_i = s_i = c$ $c \notin Protected$	$Rename(A, p, s_i, cdummy)$	
	$t_i \in \vec{t}^m$ $s_i \in \vec{s}^m$ $s_i = c, t_i = X$	$Weaken(A, p, i, cdummy)$	
	$A \notin Protected$ $noAxiomDele \notin$ Heuristics	Delete axiom A.	

Table 7.2: Repair plan generator for blocking the proof of the incompatibility: *A is an input axiom whose source is S and body is $\vec{B}$; $G1$ is the left most subgoal which is resolved by A's head; $\vec{s}^m$ is a set of constants, while $\vec{t}^m$ is a set of terms which can be the mix of constants and variables; $Protected$ is a set of protected items and Heuristics is a set of chosen heuristics. This generator outputs all possible repairs and the details of applying each repair plan is given in §5.1.1 on page 68.*

heuristics are the input of repair generation. A cost limit is employed to terminate the repair process when its search is too deep or the applied RPs have been too many. Heuristics decide whether a repair plan is applicable or not. Thus, the strategy of searching for a fault-free theory is based on a depth-limited version of depth-first search with chronological backtracking.

Table 7.2 gives the repair generation of all possible repairs w.r.t. a proof of an incompatibility. A proof record starts from the end of a proof which is the empty clause together with the evidence of the empty clause, as shown in the second row in the table. The generator searches for repairs by backtracking the evidence and terminates when reaching the original goal that comes from the preferred structure, as shown in the first row of the table. In one resolution step, the unification is between the left most subgoal

Proof/Evidence	Condition	Rpair Plan
Proofs of sufficiency P	$\gamma(P,G) = R$ $noRuleAdd \notin Heuristics$	Analogise(R)
Evidence of a partial proof: $((-G_1 \vee -\vec{G}^n), \mathbb{D}))$	$A \in \mathbb{T} \wedge A \notin Protected$ $\mathbb{T} \vdash \vec{B}$, where A is $(+P \vee -\vec{B})$	Reform(AP)
	$noAxiomAdd \notin Heuristics$	Add(G_1)
	$noRuleAdd \notin Heuristics$ $\mathbb{T} \vdash +F$	Add($+G_1 \vee -F$)
	$(-G_1 \vee -\vec{G}^n) \dot{\sim}_{\mathbb{D}} (R,B)$	Delete(R,B)

Table 7.3: Repair plan generator for unblocking the proof of the insufficiency: *the output is a set of all possible repairs. All the symbols here are same with Table 7.2; F is a theorem of the theory; function $\dot{\sim}_{\mathbb{D}}$ returns true when the sub-goals on its left is originally from the rule R's preconditions $\vec{B}$ in the evidence $\mathbb{D}$.*

G_1 and the input axiom A. For each resolution step, there are five repair candidates. If a candidate does not violate the protected set nor the heuristics, it is generated as a repair plan.

Table 7.3 gives the repair generation based on a proof template or an evidence w.r.t. an insufficiency. There is no recursion in this generator. When a rule in a proof template is selected based on Equation 5.31 as the template of analogical abduction, that rule is analogised to unblock a proof. Discussion of this repair is in §5.2. Given an insufficiency evidence, the repair plans in Table 7.3 correspond to repair plans in Table 5.2, where the last repair plan is to delete unprovable preconditions from a rule.

7.4 Summary

The implementation of unique name assumption with exceptions is described in this chapter. By employing a tree structure, the equivalence class is defined. Accordingly constants are replaced with the representatives in their ECs in both the input theory and the preferred structure. Then the conflicts of the equality/inequality can be detected.

SL-resolution is the key component of the fault detection, of which the implementation is summarised in Table 7.1. Once a fault is detected, its proof or the evidence of partial proof is recorded in the structure given in §7.2.2. That structure makes it convenient to trace back the original input precondition from which a subgoal is from. Meanwhile, it is easy to calculate the substitutions applied to an input axiom in a proof. These advantages support repair generation.

Chapter 8

Evaluation

The purpose of this chapter is to provide evidence for the hypothesis outlined in the introduction:

Hypothesis: by combining abduction, belief revision and conceptual change via reformation, ABC is capable of repairing Datalog theories with the *best results* within the scope of the repair operations based on all the individual techniques.

Best results are defined below.

Definition 8.0.1 (Best Results)**.** *The best results are the repaired theories with the properties listed below.*

1. *they satisfy the preferred structure;*
2. *the applied repair operations on them are all necessary in terms of fault-repairing;*
3. *they embrace commonsense meanings.*

It can be seen that the first two properties are *objective*. Still, to go further, the heuristic measure of human judgement can check which repairs embrace commonsense meanings. And, in contrast to the objective measure, this is subjective.

Property 2 does not require a best repaired theory to be produced with the fewest repair operations, because it is possible that a theory with commonsense meanings needs a combination of more repair operations than the fewest ones. Therefore, as long as there is no redundant repair operation which does not fix any fault, the corresponding theory is not rejected to be best.

The evaluation criteria are defined as the *Silver Standard* and the *Gold Standard* (GS) based on the aforementioned properties.

Definition 8.0.2 (Silver Standard)**.** *The repaired theories that have Properties 1 and 2 in Definition 8.0.1.*

Definition 8.0.3 (Gold Standard)**.** *the repaired theories which are not only at the Silver Standard, but also have Property 3 in Definition 8.0.1.*

The following objectives are outlined for the project that would fulfil this hypothesis:

1. To develop an automatic mechanism for detecting insufficiency and incompatibility w.r.t. a preferred structure, and generating repairs that combine abduction, belief revision and conceptual change.

2. To provide a set of postulates that a repaired theory should satisfies. These postulates are the guidance of the repair generation.

3. To establish an algorithm which evaluates the fitness of repairs and prunes sub-optimal repairs.

4. To formalise heuristics to guide the repair process and enable the user to customise wanted repairs while avoiding unwanted ones.

5. To evaluate these abilities against a representative set of examples of Datalog theories with observations, formalised as the preferred structures, demonstrating that these abilities are useful and can be successfully performed.

Preliminary

The terminology that will be used in this section is summarised as the follows.

- **Semi-repaired theory**: a theory that has been repaired but is not fault-free, which needs further processing. It is not the output of ABC but an intermediate product.

- **Fully repaired theory**: a fault-free theory while respecting its preferred structure that has been found by ABC. Some of them may not be output as the final results due to the sub-optimal pruning.

- **Final repaired theory**: a fully repaired theory output by ABC. If the input theory is fault-free, then it is the output.

- **Best repairs (GS)**: the GS for the evaluation, which are a set of the fully repaired theories that are at the gold standard based on Definition 8. As all fully repaired theories are fault-free with fewest operations due to the sub-optimal pruning, the standard of GS is aiming higher to embrace commonsense meanings. The theories tested in this chapter are given in Appendix A, together with one of their best repaired theories (GS).

  Term GS is used when we discuss from the perspective of the whole set of the best repairs, while 'best repairs' are used when our discussion is more about each individual best repairs.

- **Produced repairs**: the repairs produced by ABC for resulting fully repaired theories. Although all the produced repairs correspond to fault-free theories, it is possible that not all of them are the best results mentioned in the hypothesis, e.g., when they cannot be interpreted well based on commonsense meanings.

- **Bad repairs**: the produced repairs that are not the best ones. These repairs correspond to the fault-free theories which violate commonsense or describe scenarios that are unintuitive to be true.

- **Repair length**: the repair length is the number of repair operations (ROs) applied to achieve a fully repaired theory. For example, if a fully repaired theory is produced by ROs: $\{expand(a), rename(b), delete(c)\}$, then its repair length is three.

- **Two non-terminations**: there are two kinds of non-terminations in this repair system. The *branch non-termination* is caused by just one search branch of ROs to achieve a fault-free theory. The search non-termination refers to the whole search over the space of all search branches to output some final repaired theories.

  The former is because a repair operation may introduce new faults and then a search branch could be endless. Without the sub-optimal pruning or the cost limit on repair length, a branch non-termination will result a search non-termination. On the other hand, an unbounded number of terminating search branches can also cause the search non-termination.

- **Non-Exhaustive Search (NES)**: when search non-termination occurs, the system cannot exhaustively explore the whole search space due to space or time limit, and then it is a NES case where not all possible fully repaired theories are produced by the system but only some of them.

- **Hybrid repair**: a hybrid repair is one which combines different repair techniques to achieve a fully repaired theory, e.g., abduction and reformation.

- **Execution time (E-time)**: the run time from the minimal set computation of the input theory to the time when all fully repaired theories are output. E-time is calculated with the equation below, where t_d, t_a are the time at point d and a respectively, which are drawn in the flow chart in Figure 8.1.

  $$\text{E-time} = t_d - t_a$$

  E-time reflects the overall time cost of ABC.

- **Single Fault Repair Time (SFR-Time)**: the average time cost for generating all possible ROs w.r.t. one fault for an input theory. Based on Figure 8.1, SFR-Time can be calculated with the equation below, where function $\mathcal{E}$ returns the average value of all members in its input set; n is the total number of faults; t_{ci} and t_{bi} are the time at the ith c point and its corresponding b point respectively.

  $$\text{SFR-Time} = \mathcal{E}(\{t_{ci} - t_{bi}, 1 \leq i \leq n\})$$

  Analysing SFR-Time helps to see whether NES cases are caused by the long time of repair generation for a particular fault.

8.1 Evaluation Methodology

This section outlines the methodology of our evaluation including the source of Datalog theories for test, GS and the aims of evaluation.

8.1.1 Data Source and GS

Theories adapted from the literature are the main source of data for our evaluation. We choose the theories which are originally problematic, e.g., inconsistent or incomplete w.r.t. essential information, in the literature. Based on these essential information, the formalisations of $\mathbb{PS}$s in our evaluation can be divided into the following kinds:

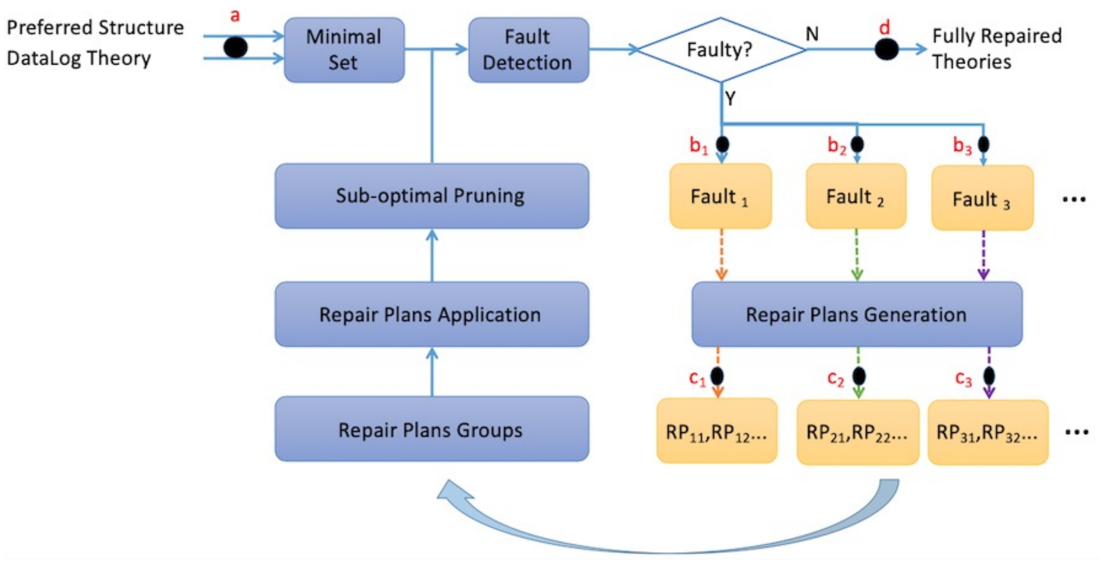


Figure 8.1: The flow chart of the ABC Repair System: *blue background represents the system operations while yellow the intermediate data including the detected faults and their corresponding repair plans, whose correspondence is depicted by arrows with dotted line in same colour; black dots are the time points used in calculating E-time or SFR-Time*.

- Represent explicit statements from the literature, e.g., the Super Penguin theory from Example 3.1 in (Gómez et al., 2010).
- Represent desired theorems from which new rules can be analogised or existing faulty rules can be repaired, e.g., the *Families*, the *Parent theory* and the *Married Women* theory adapted from Example 3.6 in (Gómez et al., 2010).

To make the evaluation objective and unbiased, all the formalisation follows the original literature and all the adaption respect commonsense meanings.

The key question of the evaluation is what is the appropriate GS for the GS. Because authors usually focus on the promising output of their work rather than all possibilities that make sense, it is not sufficient to only take the best repair originating from the literature as our GS. Meanwhile, GS for a given faulty theory is usually not unique for the following reasons.

- There could be different 'correct' scenarios for one faulty theory.
- One correct scenario can be represented by different signatures.

Because GS is still an open question in real theory repair tasks, e.g., correcting knowledge graphs, manual work is required to generate GS currently. Even human work is not perfect because they make mistakes naturally and different experts may give different answers due to personal preferences. Thus, it is hard to provide a set of best repairs as the unique GS.

In our evaluation, given a pair of a Datalog theory and its preferred structure, GS is decided based on Definition 8, which is a set of best repaired theories that describing diverse scenarios that follow the preferred structure and could happen in the real world. The scenarios in GS are various as there are dynamic possibilities in the real world. In general, our GSs may include original repairs given in the literature when they follow Definition 8; the repairs that interpret the same scenario with the original repairs but in different signatures; and the repairs represent novel scenarios which are not covered by the original repairs.

Although there is no comparison between ABC and human experts in this thesis, it is an interesting topic for future work, discussed as point 11 in §9.1.

8.1.2 Test Plans

The test plans of the evaluation are outlined in terms of what needs to be evaluated followed by the method of how to do it with tests. Here whether a fully repaired theory is fault-free is omitted. Because if not, the repair process will continue working on it until it is fault-free.

Superior Test. Are there hybrid repairs in the GS?

Method: flag the examples whose GS includes hybrid repairs [1].

As each member of GS is a representation for one scenario, and hybrids cannot be generated by the original belief revision, abduction or conceptual change. Thus, without ABC, the scenarios corresponding to the hybrids in GS would not be able to be represented.

Therefore, the occurrences of hybrid repairs in GS show the importance of the combination of repair techniques and the value of ABC.

Quality Test. How good are fully repaired theories in terms of the proportion of best repairs in them?

Method: the quality is tested by calculating the ratio of the best repairs against the total number of final repaired theories. Although the problem of counting the

[1]Because GS is open to be decidable, we cannot conclude how many best repairs the GS should contains. Thus, ratio is not used for representing how many hybrids are in GS.

best repairs in GS is undecidable, how many repaired theories from the outputted theories satisfy GS are countable. Therefore, a ratio can be calculated in this test.

Refinement Test. How good is the sub-optimal pruning? Can it improve the ratio of best repairs in the output?

Method: compare the total number of repaired theories and their quality with and without sub-optimal pruning. Also, highlight any case that any best repairs are unfortunately pruned.

Heuristics Test. How good are the heuristics? Can they improve the ratio of best repairs in the output?

Method: there are optional heuristics which can be chosen by the user. Conduct the quality test with and without heuristics respectively. The comparison of the results reflect whether the heuristics are useful.

The user chooses the heuristics by looking at the original theory before employing ABC. The chosen heuristics for each theory can be different according to its meaning. For example, there can be items which are most entrenched so they should not be rewritten. Then they can be added to a protected list to avoid being changed. As the most entrenched items for different theories are diverse, the protected lists are different. Similar to this protection list, the customised heuristics provide the user the opportunity of using some background knowledge.

Preferred Structure Test. How much do different but consistent preferred structures affect ABC applied to an input faulty theory?

Method: adapt the preferred structure of an example by consistently increasing the number of its propositions and collect the corresponding final repaired theories. Then check how the differences among preferred structures affect the quality of the final repaired theories.

Informational Loss (IL) Test. How much information is lost in final repaired theories?

Method: given an original faulty theory $\mathbb{T}$ and its preferred structure $\mathbb{PS}$, the fault reverting preferred structure $\mathbb{PS}'$ can be computed whose true set and false set are given by Definition 8.2.1 on page 175. Based on the defined $\mathbb{PS}'$, $\mathbb{T}$ is fault-free. Thus, taking each of $\mathbb{T}$'s best repaired theories $\mathbb{T}'$, and the $\mathbb{PS}'$ as the input of ABC, collect their output theories. Then compare the output

Test Plan	**Need GS?**	**Column in Table 8.2**	**Section**
Superior Test	Y	#H	§8.2.1.1
Quality Test	Y	Num. of FR-Theories	§8.2.1.2
Refinement Test	Y	Comparison of #P and #NoP	§8.2.1.2
Heuristics Test	Y	Comparison of #P and #NoH	§8.2.1.2
Preferred Structure Test	Y	#FN	§8.2.1.3
Running Time Test	N	Time	§8.2.1.3
Informational Loss Test	N	Revert	§8.2.1.4

Table 8.1: The general information about each experiment.

with the original theory $\mathbb{T}$. If $\mathbb{T}$ is contained in the output, then it means that the original input theory can be recovered from best repairs, which means that the informational loss (IL) in generating best repairs is considered minimal. Furthermore, if the original input theory $\mathbb{T}$ is the unique output, then IL is zero.

Running Time Test. How long it takes to repair a theory?

Method: get E-time and SFR-Time of ABC of the tests above. Then assess them, e.g., find average case complexity by plotting a family of examples parameterised by the size of the input theory and the size of faults against E-time, respectively.

The result of the above test plans is given in the following section.

8.2 Evaluation Results

First of all, the general information of each experiment is given in Table 8.1. The first column lists all the test plans given in the last section. The second column answers whether human work is needed to determine the GS of best repairs in each experiment. More details of some tests will be given in the following section in Table 8.2. The corresponding column in Table 8.2 which shows the result of each test plan is given in the third column of Table 8.1. In the end, column *Section* gives the section number where the discussion of a test plan is.

Theory Name	Size	#FN	No. of FR-Theories			#GP	#H	E-time	Rev.
			#P	#NoP	#NoH				
Families	3-2-0	1-0	3/3	NES	4/5	N	Y	12 ms	N
Tweety	5-3-1	0-1	1/1	NES	1/1	N	Y	9 ms	Y
Married Woman	5-1-1	1-1	1/3	NES	1/3	N	N	511 ms	Y
Researcher	4-1-1	1-1	1/3	NES	1/9	N	Y	200 ms	Y
Super Penguin	6-1-1	0-1	2/6	NES	2/6	N	Y	117 ms	Y
Buy Stock	8-1-1	0-1	2/7	NES	2/10	N	Y	26 ms	Y
Working Student	5-2-1	1-1	2/3	NES	2/3	N	Y	55 ms	Y
Parent	6-3-3	3-0	1/1	NES	1/4	N	Y	23 ms	Y
Missing parent	9-4-4	2-0	1/6	NES	1/8	N	Y	392 ms	Y
Load Car	11-2-3	2-0	0/6	NES	0/25	Y	Y	213 ms	Y

Table 8.2: Experimental results showing the performance of ABC: *More details of the test result are given in Appendix A.*

8.2.1 Experiment Results

From all test sets, ten faulty theories are selected to discuss, many of which were also tested in (Urbonas et al., 2020). As the repair algorithm and the pruning mechanism[2] are all different from the version in (Urbonas et al., 2020), the same test theories allow a direct comparison. Moreover, these theories are selected from the literature of belief revision, abduction, non-monotonic reasoning and inductive logic programming, which test our system with a broader perspective than theories from one field of literature.

The experimental results are given in Table 8.2, whose columns depicts the following statistics:

Theory Name: The names of the 10 selected faulty theories in our test set. The citation of each theory is as the following: *Families* (Bundy and Mitrovic, 2016); *Tweety* (Strasser and Antonelli, 2019b); *Married Woman*, *Super Penguin*, *Buy Stock* and *Working Student* (Gómez et al., 2010); *Researcher* (Rodler and Eichholzer, 2019); *Parent* and *Missing parent* (Muggleton, 2017), and *Load Car* (Svatoš et al., 2017). Adaption may be involved, e.g., to formalise the preferred

[2]We take the minimal sums of the heuristics scores instead of taking the Pareto optimal ones as (Urbonas et al., 2020) does.

structure and translate the original theory into Datalog.

Size: Figure is given in the form of X-Y-Z, where X is the number of axioms in the tested minimal set of each faulty theory; Y and Z are numbers of propositions in the true set and the false set of the preferred structure, respectively.

#FN: Figure is given in the form of X-Y, where X is the number of insufficiencies and Y is of incompatibilities.

No. of FR-Theories: The numbers of the final repaired theories output by the system under the conditions below respectively. Figure is given in the form of X/Y, where X is the number of best theories contained in the output and Y is the total number of the output final repaired theories. Test sets **#P.**, **#NoP.** and **#NoP.** are run to investigate the effects of the sub-optimal pruning and optional heuristics on ABC's performance respectively. Note that Silver repairs include the Gold ones according to their definitions.

- **#P.** With the sub-optimal pruning and a set of optional heuristics chosen for the best performance. Because the sub-optimal pruning guarantees the output theories are of non-redundant repair operations, the outputted theories are the fault-free ones with non-redundant repair operations, so they all satisfy the Silver Standard in Definition 8.0.2. Therefore, this ratio is the outputted theories at GS against the ones at the Silver Standard.
- **#NoP.** Without the sub-optimal pruning. This is the ratio of the outputted theories at GS against all of the fault-free repaired theories including ones that may not reach the Silver Standard.
- **#NoH.** Without optional heuristics chosen. This is also the ratio of the outputted theories at GS against the ones at the Silver Standard, but for a weaker system lacking optional heuristics.

#GP: Y if any best repairs are pruned as a sub-optimal one and N otherwise.

#H: Y if GS includes at least one best repair that is hybrid and N otherwise.

E-time: The E-time in milliseconds for each case in column #P, which employs the sub-optimal pruning and applied the most suitable heuristics. Average of 3 runs, computed on a single thread.

Rev: Check whether the original theory can be reverted from a best repaired theory. Y if the original input theory can be regenerated by taking any best repair and the fault reverting preferred structure $\mathbb{PS}'$, which is defined in Definition 8.2.1 on page 175 as the input and N if there is at least one best repaired theory which cannot recover the original one. If yes, then the original theory is recovered which means that IL of the repair process in that case is minimal.

More analysis of this table will be given in the following sections based on test plans one by one.

8.2.1.1 Superior Test

From Table 8.2, it can be seen that in our test cases, hybrid repairs are needed for all of the cases except the married women theory, which is depicts by column #H. Thus, without ABC, there is at least one best repair in GS that cannot be produced in each case.

For example, the combination of abduction and belief revision provides the repair of an insufficiency that finds a cause of the previously unprovable goal, while not resulting any incompatibility so no extra unwanted theorems are introduced. To avoid duplicates, examples will be discussed based on Example 8.2.1 and Example 8.2.2 given in following sections.

In the married women theory, reformation alone is capable of generating the best repair. Reformation is very powerful due to its ability to change the signature of the theory and modify the local representation in a targeted axiom. The former changes all occurrences of the repaired item and the later changes just one occurrence.

In general, the ability of combining abduction, belief revision and reformation allows the generation of a wider range of repairs which covers more best repairs than any individual repair technique it combines.

8.2.1.2 Quality, Refinement and Heuristics Tests

These three tests correspond to the column of *Num. of FR-Theories* in Table 8.2. The quality of the output repaired theories ($quality(\mathbb{T},\mathbb{FS})$) can be evaluated by the proportion of best repairs ($\text{best}(\mathbb{T},\mathbb{FS})$) contained by the final repaired theories ($\text{final}(\mathbb{T},\mathbb{FS})$).

$$quality(\mathbb{T},\mathbb{FS}) = \frac{|\text{best}(\mathbb{T},\mathbb{FS}) \cap \text{final}(\mathbb{T},\mathbb{FS})|}{|\text{final}(\mathbb{T},\mathbb{FS})|} \tag{8.1}$$

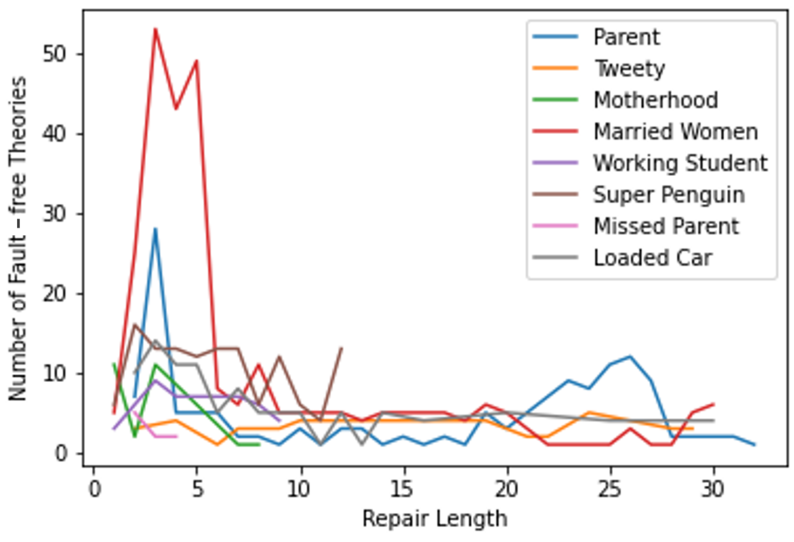


Figure 8.2: The number of fully repaired theories of different repair length found in NES cases, which is from #NoP column in Table 8.2.

The average quality of #P is 0.512 while the average quality of #NoH is 0.382. Note that all final repaired theories are fault-free so they are good in terms of respecting the preferred structure. The quality here is calculated based on the best repairs that describe meaningful scenarios. It can be seen that the optional heuristics are useful to improve the output quality of the system. If the user has the background knowledge about what should not be changed, e.g., the arity of a predicate, then optional heuristics can be chosen at the beginning of the repair process. Alternatively, heuristics can be chosen to avoid some unwanted repairs for the next round after all repaired theories are suggested. Then the quality of the newly repaired theories can be improved.

No calculation is needed to compare #P with #NoP. Without the sub-optimal pruning, all the tested cases become NES. Figure 8.2 is drawn based on the found fully repaired theories in NES cases. The number of fault-free theories with short repairs is generally more than one with long repair in Figure 8.2.

Figure 8.2 depicts that NES is caused by the huge number of search branches of fully repaired theories and some of them are of long repair length, even longer than 15! From our tests, the repair length of best repairs is usually quite short. Therefore, sub-optimal pruning is vital which can prune bad branches that are sub-optimal. Meanwhile, a cost limit can be set to prune long ROs.

However, in the case of load car, the unique best repair is pruned before being fully generated, which can be found in the full repaired theories in the log of the no pruning version.

The input of the load car theory is given in Example 8.2.1.

Example 8.2.1. *Faulty Load Car Theory.*

$$\begin{aligned}
&\implies boxShape(load3)\\
&\implies hasCar(car1)\\
&\implies hasCar(car2)\\
&\implies hasCar(car3)\\
&\implies hasCar(car4)\\
&\implies notboxShape(load2)\\
&\implies hasLoad(car1, load1)\\
&\implies hasLoad(car2, load2)\\
&\implies hasLoad(car4, load3)
\end{aligned}$$

$\mathcal{T}(\mathbb{PS}) = \{eastBound(car1), eastBound(car4)\}$

$\mathcal{F}(\mathbb{PS}) = \{eastBound(car2), eastBound(car3), eastBound(load1)\}$

The best repair is to add rule: $boxShape(Y) \wedge hasCar(Z) \wedge hasLoad(Z,Y) \implies eastBound(Z)$, which is generated by combining three repair techniques.

1. Abduction: to repair the insufficiency of $eastBound(car1)$, expand the following rule.
$$hasCar(Z) \wedge hasLoad(Z, load1) \implies eastBound(Z)$$

2. Reformation: to further repair the insufficiency of $eastBound(car4)$, extend the constant $load1$ to an independent variable Y.
$$hasCar(Z) \wedge hasLoad(Z,Y) \implies eastBound(Z)$$

3. Belief revision: to block the incompatibility of $eastBound(car2)$, add a precondition $boxShape(Y)$.
$$boxShape(Y) \wedge hasCar(Z) \wedge hasLoad(Z,Y) \implies eastBound(Z)$$

The above best repair is produced in three steps. As there are fully repaired theories found in two steps, that best repair is pruned as one of the sub-optimal ones. In this case, a solution is to improve the repair algorithm so that the repair can be generated with fewer steps. Another possible solution is to develop more heuristics so that when suitable heuristics are employed, those unwanted repairs of two steps can be avoided. However, that does not guarantee a full avoidance of pruning a best repaired theory.

Consider the advantage of sub-optimal pruning and that its side effect rarely occurs, it is fairly reasonable to keep it and improve it in future work, discussed as point 7 in §9.1.

In summary, from Table 8.2, it can be seen that the best performance is with #P, which shows the benefits of employing the sub-optimal pruning and optional heuristics. This is because the customised heuristics add background knowledge, e.g., domain knowledge to our domain independent repair process and the sub-optimal pruning keeps exploring only the promising search branches of repair plans and abandons bad ones. Both reduce the search space of repair generation so boost the performance of the system.

8.2.1.3 Preferred Structure Test

The preferred structure determines how faulty an input theory is. Thus, it has a big influence on the result of repaired theories. Rather than the size of the preferred structure, the fault sets are the key factor, which is the column of #FN in Table 8.2.

It can be seen that the faults number has neither obvious effects on the quality of fully repaired theories nor the running time. Because it is not the unique factor which has an influence on these performance. For example, the best repairs are of different forms. Some can be found easier and some are not, e.g., one requires the combination of different techniques and one does not. Thus the sub-optimal pruning has different amounts of impact on the quality of the output. Therefore, the effect of the preferred structure is not reflected by Table 8.2.

To focus on the preferred structure, Example 8.2.2 is given to analyse.

Example 8.2.2. *Faulty Missing Parent Theory* $\mathbb{T}\_p$.

$\Longrightarrow female(b)$	$\Longrightarrow male(c)$	$\Longrightarrow parent(a,b)$
$\Longrightarrow female(d)$	$\Longrightarrow male(f)$	$\Longrightarrow parent(a,c)$
$\Longrightarrow male(a)$	$\Longrightarrow male(g)$	$\Longrightarrow parent(d,b)$

Based on different preferred structures for the missing parent theory, some typical fully repaired theories are listed in Table 8.3, where the last column shows the new axioms expanded as a produced repair.

$\mathcal{T}(\mathbb{PS})$	$\mathcal{F}(\mathbb{PS})$	**Produced Repairs of Expansion**
father(a,c), mother(d,b), father(f,a)	NA	$Y \neq Z \implies father(Y,Z)$ $parent(Y,Z) \implies mother(Y,Z)$
father(a,c), mother(d,b), father(f,a)	mother(a,b), mother(a,c)	$Y \neq Z \implies father(Y,Z)$ $parent(Y,Z) \wedge female(Y) \implies mother(Y,Z)$
father(a,c), mother(d,b), father(f,a)	mother(a,b), mother(a,c), father(d,b)	$Y \neq Z \wedge male(Y) \implies father(Y,Z)$ $parent(Y,Z) \wedge female(Y) \implies mother(Y,Z)$
father(a,c), mother(d,b), father(f,a)	mother(a,b), mother(a,c), father(d,b), father(g,a)	$parent(Y,Z) \wedge male(Y) \implies father(Y,Z)$ $parent(Y,Z) \wedge female(Y) \implies mother(Y,Z)$ $\implies parent(f,a)$

Table 8.3: Typical fully repaired theory for Example 8.2.2 w.r.t. different preferred structures: *the last column gives the new axioms added in each repair.*

The first three repairs are not the best one under the commonsense meanings of the predicate but the last one is. It can be seen that as the preferred structure gets more complete, the repairs generated get closer to the best one.

These repairs are hybrid because they are generated by combining abduction and the variant belief revision: abduction finds a rule R which solves an insufficiency and then the variant belief revision adds a precondition to R when R is involved in a proof of incompatibility. For example, if the inputs are $\mathbb{T}_{-}p$ and the preferred structure given in the last row of Table 8.3, then R is $parent(Y,Z) \implies mother(Y,Z)$, which causes an incompatibility of $\mathbb{T}_{-}p \cup \{R\} \implies mother(a,c)$. Then the variant belief revision adds $female(Y)$ to R. Then their combination produces the target rule: $parent(Y,Z) \wedge female(Y) \implies mother(Y,Z)$.

Table 8.3 lists different versions of preferred structure. Each time, the bad repair reveals the missing part of the corresponding concept, which is mother/2 and father/2 here. By upgrading the preferred structure to cover the missing part, the preferred structure becomes more complete than the original. If the preferred structure is the experimental results, then this process can be seen as improving the experiments.

The input theory may also need similar upgrade. For Example 8.2.3, one of the fully repaired theories suggested by ABC is shown in Example 8.2.4, which is incorrect based on commonsense. That incorrectness is because the information about the unique mother is not represented in the original input theory. By upgrading Example 8.2.3 to Example 8.2.5, the theory and the preferred structure stand for that the birth mother *diana* is the mum of unique type to *william*. Then the unique final repaired theory suggested by the system is the best one shown in Example 8.2.6.

Example 8.2.3. *Faulty Motherhood Theory.*

$$\implies mum(diana, william)$$

$$\implies mum(camilla, william)$$

$$mum(X,Z) \wedge mum(Y,Z) \implies X = Y$$

$\mathcal{T}(\mathbb{PS}) = \varnothing$, $\mathcal{F}(\mathbb{PS}) = \{diana = camilla\}$

Example 8.2.4. *Fully Repaired Motherhood Theory (undesired).*

$$\implies mum(diana, william, dummymum2)$$

$$\implies mum(camilla, william, dummymum1)$$

$$mum(X,Z,dummymum1) \wedge mum(Y,Z,dummymum1) \implies X = Y$$

$\mathcal{T}(\mathbb{PS}) = \varnothing$, $\mathcal{F}(\mathbb{PS}) = \{diana = camilla\}$

Example 8.2.5. *Upgraded Faulty Motherhood Theory.*

$$\implies mum(lady\_di, william)$$

$$\implies mum(diana, william)$$

$$\implies mum(camilla, william)$$

$$mum(X,Z) \wedge mum(Y,Z) \implies X = Y$$

$\mathcal{T}(\mathbb{PS}) = \{diana = lady\_di\}$, $\mathcal{F}(\mathbb{PS}) = \{diana = camilla\}$

Example 8.2.6. *Fully Repaired Enriched Motherhood Theory (best).*

$$\implies mum(lady\_di, william, dummymum1)$$
$$\implies mum(diana, william, dummymum1)$$
$$\implies mum(camilla, william, dummymum2)$$
$$mum(X, Z, dummymum1) \wedge mum(Y, Z, dummymum1) \implies X = Y$$

$\mathcal{T}(\mathbb{PS}) = \{diana = lady\_di\}$, $\mathcal{F}(\mathbb{PS}) = \{diana = camilla\}$

In summary, the quality of fully repaired theories is affected by how complete the preferred structure is and how much about a concept is described in the input theory. On the other hand, when a bad repair is suggested, it reflects the missing part about some concepts in the preferred structure and/or the input theory, which can be seen as a trigger to upgrade these inputs.

8.2.1.4 Informational Loss Analysis

In Table 8.2 on page 167, the last column of 'Revert' gives the result of the IL test.

Let the original input theory, its original preferred structure and two fault sets be $\mathbb{T}$, $\mathbb{PS}$, $\mathcal{IC}(\mathbb{T}, \mathbb{PS})$, and $\mathcal{IS}(\mathbb{T}, \mathbb{PS})$ respectively. Recall the definition of incompatibility and insufficiency sets:

$$\mathcal{IS}(\mathbb{T}, \mathbb{PS}) = \{\phi \in \mathcal{T}(\mathbb{PS}) | \mathbb{T} \nvdash \phi\}$$
$$\mathcal{IC}(\mathbb{T}, \mathbb{PS}) = \{\phi \in \mathcal{F}(\mathbb{PS}) | \mathbb{T} \vdash \phi\}$$

To analyse IL, the fault reverting preferred structure $\mathbb{PS}'$ is defined.

Definition 8.2.1 (Fault reverting preferred structure ($\mathbb{PS}'$))**.** *The true set and the false set of $\mathbb{PS}'$ are $\mathcal{T}(\mathbb{PS}')$ and $\mathcal{F}(\mathbb{PS}')$ given below.*

$$\begin{aligned} \mathcal{T}(\mathbb{PS}') &= \mathcal{IC}(\mathbb{T}, \mathbb{PS}) \cup (\mathcal{T}(\mathbb{PS}) - \mathcal{IS}(\mathbb{T}, \mathbb{PS})) \\ \mathcal{F}(\mathbb{PS}') &= \mathcal{IS}(\mathbb{T}, \mathbb{PS}) \cup (\mathcal{F}(\mathbb{PS}) - \mathcal{IC}(\mathbb{T}, \mathbb{PS})) \end{aligned} \tag{8.2}$$

Based on the above definitions, the following theorems are satisfied.

$$\begin{aligned} &\forall \alpha \in \mathcal{T}(\mathbb{PS}'),\ \mathbb{T} \vdash \alpha \\ &\forall \alpha \in \mathcal{F}(\mathbb{PS}'),\ \mathbb{T} \nvdash \alpha \end{aligned} \tag{8.3}$$

Therefore, the original theory $\mathbb{T}$ is expected to be recovered based on $\mathbb{PS}'$ whose true set and false set are $\mathcal{T}(\mathbb{PS}')$ and $\mathcal{F}(\mathbb{PS}')$ respectively.

The process of IL test is to take each of best repaired theories $\mathbb{T}'$ based on GS and $\mathbb{PS}'$, test if the original input theory $\mathbb{T}$ can be recovered. Then IL is considered to be minimal if Equation 8.4 is satisfied, which corresponds to 'Y' on 'Revert' column in Table 8.2. Here $\mathcal{R}$ is the function which takes a Datalog theory and its preferred structure as the input and the fully theories as its output; $\mathcal{GS}$ takes all fully repaired theories as its input and then returns the best ones.

$$\exists \mathbb{T}' \in \mathcal{GS}(\mathcal{R}(\mathbb{T},\, \mathbb{PS})) \implies \mathbb{T} \in \mathcal{R}(\mathbb{T}',\, \mathbb{PS}') \tag{8.4}$$

It can be seen that all sampled best repaired theories in Table 8.2 can be recovered except the families theory. Because nearly all repair plans in our system give the minimal change according to the equations above. However, there is an exception: repair plan of arity increment of a predicate.

The consanguinity theory $\mathbb{T}\_f$ is given in Example 8.2.7; its $\mathbb{PS}'$ and one of its fully repaired theories $\mathbb{T}\_f'$ is given in Example 8.2.8. The last argument of predicate parent is deleted in $\mathbb{T}\_f'$. Taking $\mathbb{T}\_f'$ and its $\mathbb{PS}'$ as the input, the closest recovered theory $\mathbb{T}\_f''$ to the original theory $\mathbb{T}\_f$ is shown in Example 8.2.9.

Example 8.2.7. *Consanguinity Theory* $\mathbb{T}\_f$

$$\begin{aligned} &\implies parent(a,b,birth) \\ &\implies parent(a,c,step) \\ parent(X,Y,birth) &\implies consanguinity(X,Y) \end{aligned}$$

$\mathcal{T}(\mathbb{PS}) = \{consanguinity(a,b), consanguinity(a,c)\},\ \mathcal{F}(\mathbb{PS}) = \varnothing$

Example 8.2.8. *A Best Repaired Consanguinity Theory* $\mathbb{T}\_f'$*: decrease the arity of parent*

$$\begin{aligned} &\implies parent(a,b) \\ &\implies parent(a,c) \\ parent(X,Y) &\implies consanguinity(X,Y) \end{aligned}$$

$\mathcal{T}(\mathbb{PS}') = \{consanguinity(a,b)\},\ \mathcal{F}(\mathbb{PS}') = \{consanguinity(a,c)\}$

Example 8.2.9. *The Recovered Consanguinity Theory $\mathbb{T}\_f''$: increase the arity of parent*

$$\begin{aligned} &\Longrightarrow parent(a,b,dummy1) \\ &\Longrightarrow parent(a,c,dummy2) \\ parent(X,Y,dummy1) &\Longrightarrow consanguinity(X,Y) \end{aligned}$$

$\mathcal{T}(\mathbb{PS}') = \{consanguinity(a,b)\}$, $\mathcal{F}(\mathbb{PS}') = \{consanguinity(a,c)\}$

It can be seen that the original constants $birth$ and $step$ become $dummy1$ and $dummy2$ respectively. The informational loss here is the meaning of the original constants.

Notice that even if an axiom is deleted, it can be recovered based on $\mathbb{PS}'$. Because it can be involved in a wanted proof to unblock, where the predicate and arguments will be determined by the resolution steps that it will unblock. For example, if $\mathbb{T}\_f''$ in Example 8.2.9 is the input theory, and its $\mathcal{T}(\mathbb{PS})$ is $\{parent(a,b,birth), parent(a,c,step)\}$, then $dummy1$ and $dummy2$ will be merged to $birth$ and $step$ respectively by reformation.

In summary, most of repair plans except of arity increment are minimal changes in the sense of being able to reverted.

8.2.2 Running Time Experiment

To make the experiment more complete, ABC is evaluated with faulty theories whose faults number and axioms number are scaled gradually in this section.

Table 8.2 on page 167 gives the E-time of repairing each faulty theory under the condition of pruning the sub-optimal and employing the most suitable heuristics. From the results, it can be seen that their time cost is at the level of seconds, which is acceptable. However, these theories are all of small sizes. As the field of automatic theory repairing is still in its initial stage, it is common to perform the research on small theories. Large theories is left for future work, discussed as point 10 in §9.1.

As the efficiency of algorithms and systems is desirable, we evaluate ABC's performance by applying it to a step in a real geographic KGs alignment task. The sizes of theories in this sub-task are parameterized by n, so we can plot run time against n and classify the resulting graph.

The main factors to analyse for aligning two geographic KGs include their entities' labels, types and locations, where the first two requires natural language analysis, and the last is about the distance between two candidate entities to align. Given a threshold, two entities can be aligned if their distance is smaller than the threshold. This threshold can be set with different values for different types of entities. The propositions of their distance judgement, $range(X, Y, in)$ and $range(X, Y, out)$, are used to declare whether the distance between the entities that instantiated X and Y is smaller or bigger than their threshold respectively.

In this evaluation, the input faulty theory is constituted by three parts:

- assertions of entity IDs from two databases together with their distance judgements;
- a rule to determine whether two entries are correctly aligned w.r.t. their distance judgements;
- a rule without a goal clause to represent that a distance cannot be both smaller and bigger than the threshold.

The example of assertions and the two rules are given by Example 8.2.10, where the entities $idA1$ and $idB1$ are concluded as a correct alignment. It can be seen that there are only two rules in this input theory. 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.

Example 8.2.10. *Assertions and rules in KGs alignment task*

$$
\begin{aligned}
&\implies databaseA(idA1)\\
&\implies databaseB(idB1)\\
&\implies range(idA1, idB1, in)\\
range(X,Y,in) \wedge databaseA(X) \wedge databaseB(Y) &\implies match(x,y)\\
range(X,Y,in) \wedge range(X,Y,out) &\implies
\end{aligned}
$$

The preferred structure is formalised by adding the known misalignments to the false set and the correct alignments to the true set[4]. Then the repair process of a

[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.

[4]All the misalignments are given, so that the GS of the desired repair is determined, which helps to evaluate the E-time of ABC.

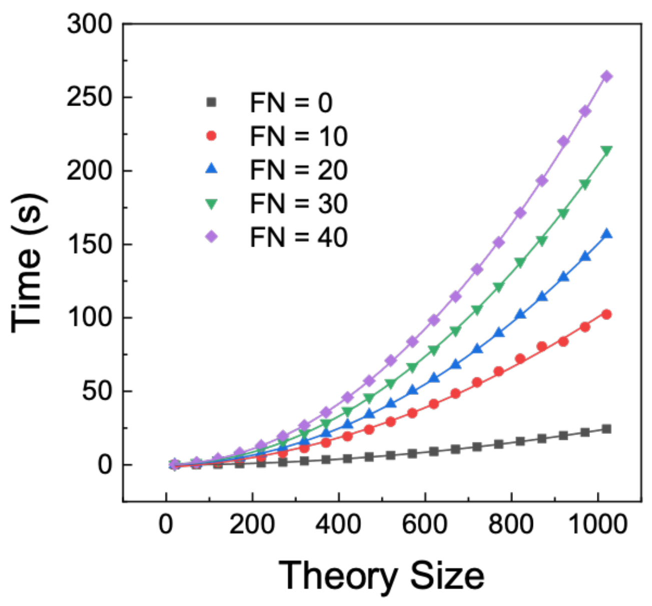


Figure 8.3: The quadratic fit of E-time of repairing the sampled theories in different sizes and different numbers of faults using OriginLab (OriginPro Version 2019b[5]).

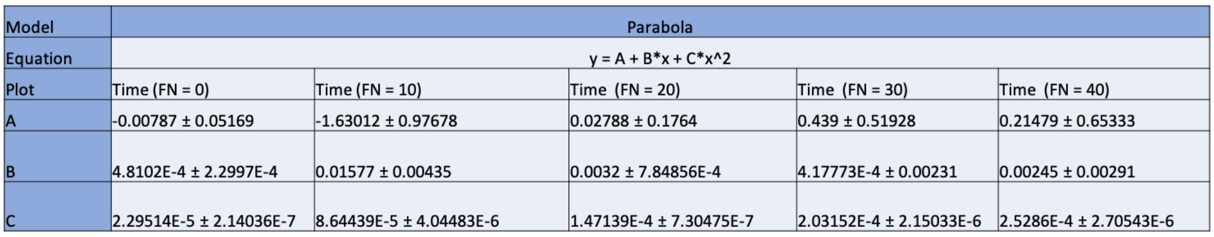
fault is to either delete or add an axiom to the alignment database or rename the third argument of an assertion of range/3. This test illustrates a possible application of ABC to a sub-task in a project with large theories.

In our experiment, the 20 samples of theory size are from 20 to 1020 with a gap of 50 and the faults number are from 0 to 40 with a gap of 10. Figure 8.3 gives the E-time of repairing each sampled faulty theory with dots and draws the quadratic curves which fits these dots best. The fitting functions are parabola generated by OriginLab (OriginPro Version 2019b[5]), whose equations are given in Figure 8.4. This example shows that ABC can scale. Notice that the pre-process of converting an input theory into one in the internal format is not covered in E-time. Because the user can directly give the input in the internal format, so that pre-process can be omitted. Also it is shared by all the samples, so only needs to be incurred once. In the case of $FN = 0$, the time is cost by the fault detection which checks all the propositions in the $\mathbb{PS}$.

The large scale theory is simple in terms of the rules it contains. It is common that for large databases, the proportion of rules against assertions are usually lower than small ones. For example, based on the LOD-a-lot survey (Garcia et al., 2017) of the Linked Open Data cloud, under 2% of total statements are rules[6].

[5]OriginPro Version 2019b: https://www.originlab.com/2019b

[6]For more details, please see https://frankvanharmelen.home.blog/2020/07/13/2-makes-all-the-difference-on-the-lod-cloud/.



Model	Parabola				
Equation	y = A + B*x + C*x^2				
Plot	Time (FN = 0)	Time (FN = 10)	Time (FN = 20)	Time (FN = 30)	Time (FN = 40)
A	-0.00787 ± 0.05169	-1.63012 ± 0.97678	0.02788 ± 0.1764	0.439 ± 0.51928	0.21479 ± 0.65333
B	4.8102E-4 ± 2.2997E-4	0.01577 ± 0.00435	0.0032 ± 7.84856E-4	4.17773E-4 ± 0.00231	0.00245 ± 0.00291
C	2.29514E-5 ± 2.14036E-7	8.64439E-5 ± 4.04483E-6	1.47139E-4 ± 7.30475E-7	2.03152E-4 ± 2.15033E-6	2.5286E-4 ± 2.70543E-6

Figure 8.4: The quadratic fit functions given by OriginLab.

To get a conclusion of how long it takes for generating all possible repair plans w.r.t. one fault, which is SFR-Time, we have collected SFR-Time from the experiments that have been done. From Figure 8.5, it can be seen that most of the SFR-Time is less than 100 ms and the time cost by the cases with heuristics is usually smaller than one without. However, there are two exceptions in the figure where the SFR-time with heuristics is bigger, highlighted with red dots. In the case with heuristics, ROs on the protected items become unavailable and other ROs will be produced. When those unavailable operations are optimal, they will be produced as the solution in the case without heuristics. If they cost less time to generate, then the SFR-Time with heuristics is bigger than without heuristics. Note that heuristics are chosen based on domain knowledge while whether a repair operation is optimal depends on how many faults it solves. The former is based on semantics and the latter syntax. Thus, an optimal repair may violate domain knowledge so it can be prevented by the chosen heuristics.

SFR-Time reflects the average time cost of one node in the search space of repair plans' generation. As SFR-Time is short so the main factor that affects E-time is the size of the search space. Because the size of the input theory and its fault numbers have a big impact on how large the search space, they are the main factors of influence on E-time of ABC.

8.3 Discussion

Based on the tests we have reported, it can be seen that most of the best repairs include the combination of different repair techniques. Furthermore, the final repaired theories contain the best repairs in most of the cases. However, some best repairs may be pruned as sub-optimal ones when they require more steps to produce than the optimal ones, which means that they are best in terms of their meanings but not in terms of requiring the minimal number of ROs.

Although the sub-optimal pruning may lose best repairs in some cases, it is vital

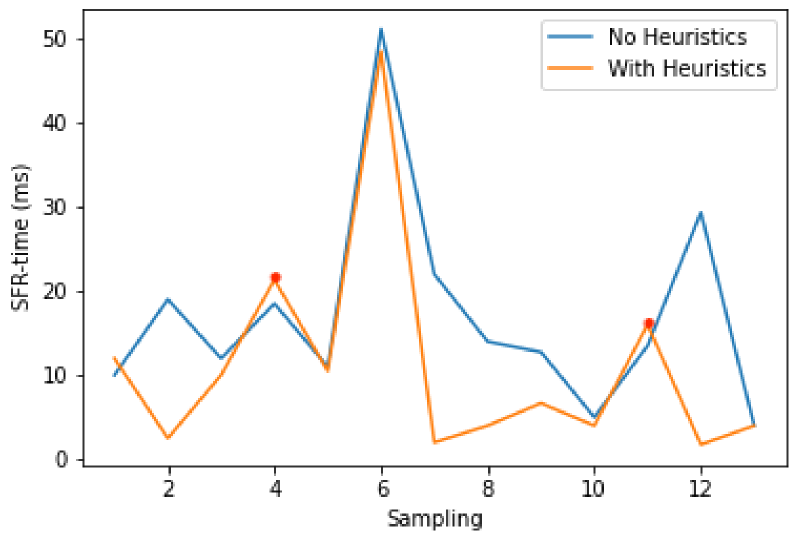


Figure 8.5: The SFR-Time of generating repair plans: *x-axis corresponds to the sampled input of an original theory and its preferred structure; y-axis is the SFR-Time by averaging data from more than 10 times of repairing single fault for each input; It can be seen that the longest generation costs less than 50 ms; the exceptions when the SFR-time with heuristics is bigger are highlighted with red dots.*

to the system as it can reduce the search space of repair generation and collect the final repaired theories rather than let the system fall into a black hole of search. A search branch can be endless when repair plans keep introducing new faults. Thus it is dangerous for the repair process to exhaustively search the whole search space. It has been frequently detected as shown by #NoP column in Table 8.2, where all the tested cases become of NES without the sub-optimal pruning.

There are usually bad repairs among all produced repairs, which are bad in terms of lacking meanings. In some cases, bad repairs reflect the shortage of the preferred structure and/or the diversity of the input theory. For example, a bad repair $father(X,Y) \implies female(Y)$ is produced when all given father samples have a daughter. Then the user can be inspired to improve the input to provide a full picture by enriching the input theory and the preferred structure: a father of a son.

The information and the time cost of the repair process are reasonable. Most of the original input theory can be reverted from its fully repaired theory. SFR-Time is not long while E-time is affected by the size of the theory and the faults set. Reducing E-time cost is the topic of improving search efficiency discussed as point 7 the future work in §9.1. For example, revise the repair process to make the search more efficient; or keep a record of the deleted arguments so that they can be fully recovered when needed and restructure the repair process to make its search more efficient. Currently, there is no domain knowledge or other background knowledge incorporated into ABC except by customising the optional heuristics. A search strategy which employs more background knowledge and hence avoids best repairs being pruned as the sub-optimal ones is also a good direction to explore.

8.4 Summary

This chapter has evaluated the performance of ABC from the perspectives of its ability to generate hybrid repairs, the quality of its output, the effects of sub-optimal pruning and optional heuristics, the influence of preferred structures that are consistent but in different sizes, the informational loss and its running time. Terminology is given in the preliminary section, after which the test plans and their methods are listed. Then the evaluation results are analysed based on different tests, followed by the discussion of the overall conclusion of the evaluation.

From the evaluation, it can be concluded that the developed repair system can produce fully repaired theories that cannot be suggested by any original individual technique it combines. Meanwhile, its repair generation w.r.t. one fault is reasonable while the whole execution time is quadratic to the size of the input theory and its faults number based on our evaluation. Furthermore, most of the repairs are minimal in the sense of being possible to be reverted.

Chapter 9

Conclusion

Throughout the project, we have developed a repair system which combines abduction belief revision and conceptual change via the reformation algorithm. Based on a finite true set and false set given as the preferred structure, the faults in an input theory are defined as insufficiency and incompatibility, which can be detected and repaired automatically. A set of fault-free theories comprise the output of this process, and the best repairs usually require the combination of different techniques which often cannot be produced by any single repair technique alone.

In this chapter, we will outline the limitations and the corresponding future work and then propose possible applications [1].

9.1 Limitations and Future Work

Although the evaluation of our repair system demonstrates the effectiveness of our repair system, there are some limitations of the current work. The limitations, which we now consider, are listed in italics below followed by the anticipated difficulties and suggestions for future work that address them.

1. *The algorithmic properties of ABC have not been analysed.*
   Future work: Analyse and prove the properties that ABC has, e.g., termination, time complexity, correctness wrt specification, logical completeness, and computational complexity.

   - Difficulties: The subroutines of ABC include proof searching and repair generation. The proof search algorithm has exponential worst-case

[1] The main contribution are summarised in §1.3.

behaviour, and repair generation can make it worse when the repair solves one fault but creates more or worse faults with complex or super long proofs. Although we have not encountered such a case, it is theoretically possible. In addition, the language change makes it harder to measure the minimal change and the correctness of repair, e.g., after increase the arity of $mum/2$ into $mum/3$, the model for the original theory does not apply to the repaired one. Also, the creativity of repairs is unbounded, so it is hard to conclude the completeness of repair operations.

- Suggestions: Analysing each property with defined scopes or in a particular perspective, e.g., only prove the completeness within the scope of ABC's repair operations. The complexity of repair generation is relevant to the total number of proofs, the length of each proof, and the signature size, e.g., how many predicates and constant symbols it has. The more predicates it has, the more candidates for a new precondition to add there could be [2].

  A possibility is to analyse the properties under extreme conditions, e.g., analyse the best performance and the worst. However, the worst maybe too bad, which does not occur in practice. Alternatively, one can analyse average case.

2. *The evaluation of the current system is still limited. It has only been applied to examples in the theory repair related literature and a task analogised from a databases alignment project.*

   Future work: Explore more applications of the repair system, e.g., apply ABC to large theories and perform systematically tests.. Some possibilities are discussed in §9.2.

   - Difficulties: It is hard to discover the whole set of the gold standard of the best repairs.
   - Suggestions: Consider theories coming from other fields where best repairs are provided, and then test ABC with randomised choice of faulty theories. Human repairs found manually can be a good source of the gold standard, which can be found in applications where human behaviours are modelled (Bundy et al., 2020). For example, in the field of program debugging, given

[2]The number of predicates is not the only factor that decides the number of repairs of adding a new precondition: among these candidates of new preconditions, only adding the ones that cannot be resolved are the desired repairs.

a faulty program as the theory to repair, the corrected program provided by the literature or the user is one of the gold standards. As a program can be corrected in several ways usually, one should not aim at summarising the whole set of the gold standard, but check how many outputted repaired theories are correct and their ratio against the total repaired theories.

3. *The input theory is currently restricted to Datalog.*
   Future work: Extend the system to other logics, e.g., description logic, temporal logic, epistemic logic and so on.

   - Difficulties: The completeness and soundness of the reasoning in ABC can be challenging dealing with other logics, e.g., full FOL or higher order logics.
   - Suggestions: Start with the logics that are basic and widely used in the possible applications of ABC, e.g., description logic, because it is more likely to have useful research that supports the reasoning tasks.

4. *Dummy terms in a repaired theory are not assigned with any meaning.*
   Future work: Replace dummy terms with semantically meaningful name.

   - Difficulties: It requires domain/background knowledge which is not available from the input of ABC.
   - Suggestions: Consider user interaction or reasoning with background knowledge, e.g., knowledge retrieving. In this task, the main questions to solve include knowledge sources, data-mining techniques and the selection of names from all candidates.

5. *The repair which reorders arguments is not available.*
   Future work: Explore more repair operations which are not available in the current ABC Repair System.

   - Difficulties: It is an open question about what are other repair operations. And for each repair operation, the strategy of applying them is challenging, e.g., which order is correct when reordering arguments?
   - Suggestions: It can be helpful to find inspirations from how human revise and organise their knowledge. For example, when a person has some inconsistencies in his knowledge, he/she may choose the repair solution

which solves the most inconsistencies. Similarly, In terms of reordering arguments, the order that results in the least remaining faults can be seen as the best order.

6. *The preferred structure is fixed, which limits the final repaired theories especially when the preferred structure is not adequate.*
   Future work: Evolving the preferred structure as well as repair the input theory.

   - Difficulties: The preferred structure is the benchmark of the correctness of the input theory, whose more care needs to be taken when changing it. An automatic mechanism of finding the trust worthy evolution of the preferred structure is difficult to achieve.
   - Suggestions: Could design and carry out experiments to determine the truth value of ground propositions. And then evolve the preferred structure accordingly. In the end, the repair process would be an evolution constituted by a series of repair missions. At the beginning, this could require user interactions, and then automatic mechanisms could be established by analysing and modelling human's work.

7. *Although the search space of repair operations of a single fault is not big, their combination for all faults can result in a large search space. Then the consequence can be a long execution time or a non-exhaustive search.*
   Future work: Improve search efficiency.

   - Difficulties: An expected possible side effect of increasing the search efficiency is losing some best repairs, which should be avoided when developing new search strategies.
   - Suggestions: 1) One possibility is to reduce the search space of repair operations of a single fault, e.g., only merge two predicates from one sort or one class, which will be relevant to sorted logic or description logic; reduce the search space of the precondition to add to the relevant domain based on background knowledge available. 2) Another option is to reduce the number of search branches. The existing relevant method is sub-optimal pruning. It will be useful to explore if other techniques can do further pruning or a better algorithm to improve the performance of the sub-optimal pruning, e.g., prune some search branches when they tend to violate background knowledge. 3) Besides, the reduction of the length of

search branches will also help. For example, by analysing the problematic proofs or evidence further, generate a compound repair operations which solves multiple faults in one step of repair generation rather than one repair operation in one step of generation.

8. *The question of which produced fault-free theory is more likely to be involved in the best repairs in terms of its semantic interpretations.*
   Future work: Employ probability based on the domain knowledge.
   - Difficulties: Domain knowledge is required to make the decision.
   - Suggestions: When the domain knowledge can be incorporated into the repair system, it will be useful to develop an algorithm for calculating the probability of a fault-free theory being the best repair. For example, when the fault is with either the date or the day-of-week, then the latter has a bigger chance to be correct. Because the possibility of the latter being correct is $1/7$ while the former usually $1/30$.

9. *A random minimal input theory is selected when there are multiple smallest candidates. It is unknown whether a selection function is necessary to develop for approaching the best repairs.*
   Future work: Investigate whether the repairs are significantly different depending on various minimal input theories. If yes, then it is worthwhile to develop a selection function for picking the minimal theory based on which the best repairs can be generated.
   - Difficulties: There could be huge numbers of minimal sets of one input theory.
   - Suggestions: Build an algorithm to automatically evaluate the relation between different minimal sets and the quality of their repaired theories.

10. *It has not been compared between human experts and ABC.*
    Future work: the gaps provide directions about how domain knowledge effects repair result.
    - Difficulties: This work requires domain experts.
    - Suggestions: Collaborate with knowledge management projects where repair tasks are involved and the gold standard given by domain experts are provided.

9.2 Applications

Theory repair is a common task in many AI fields. Except from applying our system on a single theory repair task, some possible applications in a more complex context in future are proposed.

- It can be a component of a strategy-making agent. For example, as a part of an agent using a game theory to generate a winning strategy under certain context (Bundy et al., 2020), where the setting of the game can be given as the input theory with the unchanged part protected in optional heuristics and the proposition representing winning in the true set of the preferred structure and the losing ones in the false set.

  Another example is to customise a travel plan where the places to go can be listed in the true set and the places to avoid in the false set. Similarly, the preferences of transportation can be also described in the preferred structure. Then the overall information about the targeting area and transportation can be formalised as the input theory together with rules which describe the basic plan strategy, e.g., the time to leave a place should not be earlier than 9am if the user does not want to get up early.

  As the unexpected repairs can reflect the unknown elements of a system that are not covered by the preferred structure, our system can be used to assist the experiment designer (Bundy, 2019). By analysing these unexpected repairs, more experiments can be designed to test whether these repairs are correct or not. This would help to improve the completeness (in logical terms) of an experimental research project.

- Our system can be used to address the mismatches found in Databases alignment or merging. It can be done by formalising all information about the entries as axioms, their alignment assertions and the alignment rules which decide a correct alignment as the input theory. We then assume that all alignment assertions are correct and derive the assumed properties that each aligned entry should have based on alignment rules; and formalise these properties as propositions in the preferred structure. If there is an incorrect alignment, the assumed property of its entries will not be coherent with the actual property axioms, which will be detected as incompatibilities and insufficiencies. Then repair system will adjust alignment rules or remove incorrect alignment assertions.

- It can be used to update a database by communicating with the user. For example, when the user wants something that is inconsistent with the conclusion derived from the database, the user's wish can be formalised as the true set of the preferred structure. Together with the data in the database as the input theory, the repair system can produce a set of fault-free theories as candidates of updating the database. Then it can ask the user about why that is.

  As in Example 9.2.1, when the user asks the agent to play a song at 13 o'clock on 16th July, which is a sunny day, the agent would play a rock song based on its database. To save space, the year is omitted from the time stamp in the example. Assume that the user refuses it and lets it play jazz. In this scenario, the preferred structure can be formalised based on user's choice, based on which, the repair system can detect an insufficiency of $play(131607, c)$ and an incompatibility of $play(131607, a)$. Then the theory can repair the database in different ways.

  One repair plan is to rename the type of songs whose fault-free theory is shown in Example 9.2.2. Or it can analogise the rule of which music to play as in Example 9.2.3. To confirm the user's preference, the agent can ask the user, "Why would you like to play c this time? Is it also a piece of rock music?" If the user replied, "Yes, c is rock, but a isn't", then the repair in Example 9.2.2 is the best repair and the dummy term can be renamed by further retrieving its information online or by asking the user. On the other hand, if the user replied, "Because it is working hours." Then the repair in Example 9.2.4 is the best repair, which can be produced by renaming dummy predicates with the keywords from the user's answer.

Example 9.2.1. *Music Original.*

$$\begin{aligned}
&\implies song(a, rock)\\
&\implies song(c, jazz)\\
&\implies timeStamp(131607)\\
&\implies weather(131607, sunny)\\
timeStamp(X) \wedge weather(X, sunny) \wedge song(Y, rock) &\implies play(X, Y)
\end{aligned}$$

$\mathcal{T}(\mathbb{PS}) = \{play(131607, c)\}$, $\mathcal{F}(\mathbb{PS}) = \{play(131607, a)\}$

Example 9.2.2. *Music 1.*

$$
\begin{aligned}
&\Longrightarrow song(a, dummy\_rock)\\
&\Longrightarrow song(c, rock)\\
&\Longrightarrow timeStamp(131607)\\
&\Longrightarrow weather(131607, sunny)\\
timeStamp(X) \wedge weather(X, sunny) \wedge song(Y, rock) &\Longrightarrow play(X, Y)
\end{aligned}
$$

$\mathcal{T}(\mathbb{PS}) = \{play(131607, c)\}$, $\mathcal{F}(\mathbb{PS}) = \{play(131607, a)\}$

Example 9.2.3. *Music 2.*

$$
\begin{aligned}
&\Longrightarrow song(a, rock)\\
&\Longrightarrow song(c, jazz)\\
&\Longrightarrow timeStamp(131607)\\
&\Longrightarrow weather(131607, sunny)\\
dummyPred1(X) \wedge notdummyPred1(X) &\Longrightarrow\\
dummyPred1(X) \wedge timeStamp(X) \wedge &\\
weather(X, sunny) \wedge song(Y, jazz) &\Longrightarrow play(X, Y)\\
notdummyPred1(X) \wedge timeStamp(X) \wedge &\\
weather(X, sunny) \wedge song(Y, rock) &\Longrightarrow play(X, Y)
\end{aligned}
$$

$\mathcal{T}(\mathbb{PS}) = \{play(131607, c)\}$, $\mathcal{F}(\mathbb{PS}) = \{play(131607, a)\}$

Example 9.2.4. *Music 3.*

$$\begin{aligned}
&\implies song(a, rock)\\
&\implies song(c, jazz)\\
&\implies timeStamp(131607)\\
&\implies weather(131607, sunny)\\
workingHour(X) \wedge notWorkingHour(X) &\implies\\
workingHour(X) \wedge timeStamp(X) \wedge&\\
weather(X, sunny) \wedge song(Y, jazz) &\implies play(X, Y)\\
notWorkingHour(X) \wedge timeStamp(X) \wedge&\\
weather(X, sunny) \wedge song(Y, rock) &\implies play(X, Y)
\end{aligned}$$

$\mathcal{T}(\mathbb{PS}) = \{play(131607, c)\}$, $\mathcal{F}(\mathbb{PS}) = \{play(131607, a)\}$

Although our repair system will need to be extended to fulfil some of the tasks for the proposed applications listed, the core part of the automatic repair generations are adequate to do the job. Especially in the last case, natural language processing will also play an important role to make it work.

9.3 Summary

This section surveys the original work in this thesis. The hypothesis of this project has been successfully evaluated. Section 1.3 outlines our contributions including the combination of abduction, belief revision and conceptual change via reformation; the variant belief revision; the variant abduction; the analogical abduction and the trace-back reformation. Meanwhile, the sub-optimal pruning is developed to reduce the search space of fault-free theories. Also, optional heuristics are provided to the user so that the most entrenched items from the view of the domain knowledge can be protected from being changed. Additionally, UNAE is developed to simplify the representation of the equalities and inequalities. Based on the preferred structure, algorithms for scoring the entrenchment of axioms, preconditions and the signature elements have been created which support the repair generation and repaired theories ranking. In the last chapter, the evaluation of ABC shows that the repair system produces fully repaired theories that cannot be suggested by any of the original individual techniques: belief revision, abduction or reformation. Section 9.1 lists the limitation of our current work and the corresponding future work that address them. Section 9.2 proposes some possible applications of our work where the developed repair system can serve real tasks.

Appendix A

Examples

The theories tested in Chapter 8 are given in this appendix, together with one of their best repaired theories (the gold standard). Changes in repaired theories are highlighted in red. These are the theories tested in the evaluation outlined in Chapter 8. ABC has been tested in diverse domains, such as in (Bundy et al., 2020; Urbonas et al., 2020).

1. The *Families* Theory is repaired by enriching the constant *birth* into a variable in $A1$.

Example A.0.1. *Families Theory.*

$$parent(X,Y,birth) \implies families(X,Y) \quad (A1)$$

$$\implies parent(a,b,birth) \quad (A2)$$

$$\implies parent(a,c,step)] \quad (A3)$$

$\mathcal{T}(\mathbb{PS}) = \{families(a,b),\ families(a,c)\}$
$\mathcal{F}(\mathbb{PS}) = \varnothing$

Example A.0.2. *Repaired Families Theory.*

$$parent(X,Y,Z) \implies families(X,Y) \quad (A1)$$

$$\implies parent(a,b,birth) \quad (A2)$$

$$\implies parent(a,c,step)] \quad (A3)$$

$\mathcal{T}(\mathbb{PS}) = \{families(a,b),\ families(a,c)\}$
$\mathcal{F}(\mathbb{PS}) = \varnothing$

2. The *Tweety* theory is repaired by increasing the arity of *bird*. The repaired theory

says that all birds have feathers and only normal types of birds fly, while penguin is a special type.

Example A.0.3. *Tweety Theory.*

$$penguin(X) \implies bird(X) \tag{A1}$$
$$bird(X) \implies feather(X) \tag{A2}$$
$$bird(X) \implies fly(X) \tag{A3}$$
$$\implies bird(polly) \tag{A4}$$
$$\implies penguin(tweety) \tag{A5}$$

$\mathcal{T}(\mathbb{PS}) = \{feath(tweety),\ feath(polly), fly(polly)\}$
$\mathcal{F}(\mathbb{PS}) = \{fly(tweety)\}$

Example A.0.4. *Repaired Tweety Theory.*

$$penguin(X) \implies bird(X, dummy2) \tag{A1}$$
$$bird(X, Y) \implies feather(X) \tag{A2}$$
$$bird(X, dummy1) \implies fly(X) \tag{A3}$$
$$\implies bird(polly, dummy1) \tag{A4}$$
$$\implies penguin(tweety) \tag{A5}$$

$\mathcal{T}(\mathbb{PS}) = \{feath(tweety),\ feath(polly), fly(polly)\}$
$\mathcal{F}(\mathbb{PS}) = \{fly(tweety)\}$

3. The *Married Women* theory is repaired by renaming the instance of predicate $hadHusband$ in $A2$ with $hasHusband$.

Example A.0.5. *Married Women Theory.*

$$divorced(X) \wedge notDivorced(X) \implies \qquad (A1)$$

$$hadHusband(X) \implies marriedWoman(X) \qquad (A2)$$

$$marriedWoman(X) \implies notDivorced(X) \qquad (A3)$$

$$\implies hadHusband(leticia) \qquad (A4)$$

$$\implies hasHusband(flor) \qquad (A5)$$

$\mathcal{T}(\mathbb{PS}) = \{notDivorced(flor)\}$
$\mathcal{F}(\mathbb{PS}) = \{notDivorced(leticia)\}$

Example A.0.6. *Repaired Married Women Theory.*

$$divorced(X) \wedge notDivorced(X) \implies \qquad (A1)$$

$$hasHusband(X) \implies marriedWoman(X) \qquad (A2)$$

$$marriedWoman(X) \implies notDivorced(X) \qquad (A3)$$

$$\implies hadHusband(leticia) \qquad (A4)$$

$$\implies hasHusband(flor) \qquad (A5)$$

$\mathcal{T}(\mathbb{PS}) = \{notDivorced(flor)\}$
$\mathcal{F}(\mathbb{PS}) = \{notDivorced(leticia)\}$

4. The *Researcher* theory is repaired by adding a precondition to *A*2. By interpreting *dummyPred* as paid, the repair says that only a paid author is employed.

Example A.0.7. *Researcher Theory.*

$$writes(X, papers) \implies author(X) \qquad (A1)$$

$$author(X) \implies employee(X) \qquad (A2)$$

$$activeResearcher(X) \implies writes(X, papers) \qquad (A3)$$

$$\implies activeResearcher(ann) \qquad (A4)$$

$\mathcal{T}(\mathbb{PS}) = \{activeResearcher(ann)\}$
$\mathcal{F}(\mathbb{PS}) = \{employee(ann)\}$

Example A.0.8. *Repaired Researcher Theory.*

$$writes(X, papers) \implies author(X) \quad \text{(A1)}$$

$$dummyPred(X) \wedge author(X) \implies employee(X) \quad \text{(A2)}$$

$$activeResearcher(X) \implies writes(X, papers) \quad \text{(A3)}$$

$$\implies activeResearcher(ann) \quad \text{(A4)}$$

$\mathcal{T}(\mathbb{PS}) = \{activeResearcher(ann)\}$
$\mathcal{F}(\mathbb{PS}) = \{employee(ann)\}$

5. The *Super Penguin* theory is repaired by adding a precondition to $A4$. By interpreting *dummyPred* as non-super penguin, the repair says that a bird with broken wings cannot fly if it is not a super penguin.

Example A.0.9. *Super Penguin Theory.*

$$cannotFly(X) \wedge fly(X) \implies \quad \text{(A1)}$$

$$superPenguin(X) \implies penguin(X) \quad \text{(A2)}$$

$$penguin(X) \implies bird(X) \quad \text{(A3)}$$

$$bird(X) \wedge brokenWing(X) \implies cannotFly(X) \quad \text{(A4)}$$

$$bird(X) \implies fly(X) \quad \text{(A5)}$$

$$\implies superPenguin(opus) \quad \text{(A7)}$$

$$\implies brokenWing(opus) \quad \text{(A8)}$$

$\mathcal{T}(\mathbb{PS}) = \{fly(opus)\}$
$\mathcal{F}(\mathbb{PS}) = \varnothing$

Example A.0.10. *Repaired Super Penguin Theory.*

$$cannotFly(X) \wedge fly(X) \implies \tag{A1}$$
$$superPenguin(X) \implies penguin(X) \tag{A2}$$
$$penguin(X) \implies bird(X) \tag{A3}$$
$$dummyPred(X) \wedge bird(X) \wedge brokenWing(X) \implies cannotFly(X) \tag{A4}$$
$$bird(X) \implies fly(X) \tag{A5}$$
$$\implies superPenguin(opus) \tag{A7}$$
$$\implies brokenWing(opus) \tag{A8}$$

$\mathcal{T}(\mathbb{PS}) = \{fly(opus)\}$
$\mathcal{F}(\mathbb{PS}) = \varnothing$

6. The *Buy Stock* theory is faulty due to the of $buyStock(acme)$ and $dontBuyStock(acme)$, which is repaired by adding a precondition to $A6$, and then the wanted proof of $riskyCompany(acme)$ is blocked. Here $dummyPred(X)$ can be interpreted as non-strong steel.

Example A.0.11. *Buy Stock Theory.*

$$buyStock(X) \wedge dontBuyStock(X) \implies \tag{A1}$$
$$goodPrice(X) \implies buyStock(X) \tag{A2}$$
$$goodPrice(X) \wedge riskyCompany(Y) \implies dontBuyStock(X) \tag{A3}$$
$$stong(steel) \wedge inFusion(X, steel) \implies notRiskyCompany(X) \tag{A4}$$
$$closing(X, Y) \implies riskyCompany(X) \tag{A5}$$
$$inFusion(X, steel) \implies riskyCompany(X) \tag{A6}$$
$$\implies strong(steel) \tag{A7}$$
$$\implies inFusion(acme, steel) \tag{A8}$$
$$\implies goodPrice(acme) \tag{A9}$$

$\mathcal{T}(\mathbb{PS}) = \{buyStock(acme)\}$
$\mathcal{F}(\mathbb{PS}) = \varnothing$

Example A.0.12. *Repaired Buy Stock Theory.*

$$\begin{aligned} buyStock(X)\wedge dontBuyStock(X) &\implies && \text{(A1)}\\ goodPrice(X) &\implies buyStock(X) && \text{(A2)}\\ goodPrice(X)\wedge riskyCompany(Y) &\implies dontBuyStock(X) && \text{(A3)}\\ stong(steel)\wedge inFusion(X,steel) &\implies notRiskyCompany(X) && \text{(A4)}\\ closing(X,Y) &\implies riskyCompany(X) && \text{(A5)}\\ \color{red}{dummyPred(X)}\wedge inFusion(X,steel) &\implies riskyCompany(X) && \text{(A6)}\\ &\implies strong(steel) && \text{(A7)}\\ &\implies inFusion(acme,steel) && \text{(A8)}\\ &\implies goodPrice(acme) && \text{(A9)} \end{aligned}$$

$\mathcal{T}(\mathbb{PS}) = \{buyStock(acme)\}$
$\mathcal{F}(\mathbb{PS}) = \varnothing$

7. The axiom $A3$ in *Working Student* theory says that an undergraduate student is an adult. Then the fault is caused by $working(lily)$ and $notWorking(lily)$. The theory is repaired by adding a new argument to $notWorking$. By interpreting $dummy1$ as full-time and $dummy2$ as part-time, the repaired theory says that an undergraduate student cannot do full-time work.

Example A.0.13. *Working Student Theory.*

$$\begin{aligned} notWorking(X)\wedge working(X) &\implies && \text{(A1)}\\ undStudent(X) &\implies student(X) && \text{(A2)}\\ undStudent(X) &\implies adult(X) && \text{(A3)}\\ student(X) &\implies notWorking(X) && \text{(A4)}\\ adult(X) &\implies working(X) && \text{(A5)}\\ &\implies undStudent(lily) && \text{(A6)} \end{aligned}$$

$\mathcal{T}(\mathbb{PS}) = \{undStudent(lily),\ working(lily)\}$
$\mathcal{F}(\mathbb{PS}) = \{\}$

Example A.0.14. *Repaired Working Student Theory.*

$$notWorking(X,dummy1) \wedge working(X) \implies \tag{A1}$$
$$undStudent(X) \implies student(X) \tag{A2}$$
$$undStudent(X) \implies adult(X) \tag{A3}$$
$$student(X) \implies notWorking(X,dummy2) \tag{A4}$$
$$adult(X) \implies working(X) \tag{A5}$$
$$\implies undStudent(lily) \tag{A6}$$

$\mathcal{T}(\mathbb{PS}) = \{undStudent(lily),\ working(lily)\}$
$\mathcal{F}(\mathbb{PS}) = \{\}$

8. The *Missing Parent* theory is repaired by adding rule $A11$.

Example A.0.15. *Missing Parent Theory.*

$$male(X) \wedge parent(X,Y) \implies father(X,Y) \tag{A1}$$
$$\implies female(b) \tag{A2}$$
$$\implies female(d) \tag{A3}$$
$$\implies male(a) \tag{A4}$$
$$\implies male(c) \tag{A5}$$
$$\implies male(f) \tag{A6}$$
$$\implies male(g) \tag{A7}$$
$$\implies parent(a,b) \tag{A8}$$
$$\implies parent(a,c) \tag{A9}$$
$$\implies parent(d,b) \tag{A10}$$

$$\mathcal{T}(\mathbb{PS}) = \{father(a,b),\ father(a,c),\ mother(d,b),\ father(f,a)\}$$
$$\mathcal{F}(\mathbb{PS}) = \{mother(a,b),\ mother(a,c),\ father(d,b),\ father(g,a),\ father(g,c)\}$$

Example A.0.16. *Repaired Missing Parent Theory.*

$$male(X) \wedge parent(X,Y) \implies father(X,Y) \quad (A1)$$
$$\implies female(b) \quad (A2)$$
$$\implies female(d) \quad (A3)$$
$$\implies male(a) \quad (A4)$$
$$\implies male(c) \quad (A5)$$
$$\implies male(f) \quad (A6)$$
$$\implies male(g) \quad (A7)$$
$$\implies parent(a,b) \quad (A8)$$
$$\implies parent(a,c) \quad (A9)$$
$$\implies parent(d,b) \quad (A10)$$
$$female(Y) \wedge parent(Y,Z) \implies mother(Y,Z) \quad (A11)$$

$$\mathcal{T}(\mathbb{PS}) = \{father(a,b),\ father(a,c),\ mother(d,b),\ father(f,a)\}$$
$$\mathcal{F}(\mathbb{PS}) = \{mother(a,b),\ mother(a,c),\ father(d,b),\ father(g,a),\ father(g,c)\}$$

9. The *Parent* theory is repaired by adding two rules: $A7$ and $A8$.

Example A.0.17. *Parent Theory.*

$$\implies female(c) \quad (A1)$$
$$\implies female(d) \quad (A2)$$
$$\implies male(a) \quad (A3)$$
$$\implies parent(a,b) \quad (A4)$$
$$\implies parent(a,c) \quad (A5)$$
$$\implies parent(d,b) \quad (A6)$$

$$\mathcal{T}(\mathbb{PS}) = \{father(a,b),\ father(a,c),\ mother(d,b)\}$$
$$\mathcal{F}(\mathbb{PS}) = \{mother(a,b),\ mother(a,c),\ father(d,b)\}$$

Example A.0.18. *Repaired Parent Theory.*

$$\begin{align}
&\implies female(c) && \text{(A1)}\\
&\implies female(d) && \text{(A2)}\\
&\implies male(a) && \text{(A3)}\\
&\implies parent(a,b) && \text{(A4)}\\
&\implies parent(a,c) && \text{(A5)}\\
&\implies parent(d,b) && \text{(A6)}\\
male(Y) \wedge parent(Y,Z) &\implies father(Y,Z) && \text{(A7)}\\
female(Y) \wedge parent(Y,Z) &\implies mother(Y,Z) && \text{(A8)}
\end{align}$$

$\mathcal{T}(\mathbb{PS}) = \{father(a,b),\ father(a,c),\ mother(d,b)\}$
$\mathcal{F}(\mathbb{PS}) = \{mother(a,b),\ mother(a,c),\ father(d,b)\}$

10. The *LoadCar* theory is repaired by summarising rule *A*12.

Example A.0.19. *Load Car Theory.*

$$\begin{align}
boxShape(X) \wedge notboxShape(X) &\implies && \text{(A1)}\\
&\implies boxShape(load1) && \text{(A2)}\\
&\implies boxShape(load3) && \text{(A3)}\\
&\implies hasCar(car1) && \text{(A4)}\\
&\implies hasCar(car2) && \text{(A5)}\\
&\implies hasCar(car3) && \text{(A6)}\\
&\implies hasCar(car4) && \text{(A7)}\\
&\implies notboxShape(load2) && \text{(A8)}\\
&\implies hasLoad(car1,load1) && \text{(A9)}\\
&\implies hasLoad(car2,load2) && \text{(A10)}\\
&\implies hasLoad(car4,load3) && \text{(A11)}
\end{align}$$

$\mathcal{T}(\mathbb{PS}) = \{eastBound(car1),\ eastBound(car4)\}$
$\mathcal{F}(\mathbb{PS}) = \{eastBound(load1),\ eastBound(car2),\ eastBound(car3)\}$

Example A.0.20. *Repaired Load Car Theory.*

$$
\begin{aligned}
boxShape(X) \wedge notboxShape(X) &\implies && \text{(A1)}\\
&\implies boxShape(load1) && \text{(A2)}\\
&\implies boxShape(load3) && \text{(A3)}\\
&\implies hasCar(car1) && \text{(A4)}\\
&\implies hasCar(car2) && \text{(A5)}\\
&\implies hasCar(car3) && \text{(A6)}\\
&\implies hasCar(car4) && \text{(A7)}\\
&\implies notboxShape(load2) && \text{(A8)}\\
&\implies hasLoad(car1, load1) && \text{(A9)}\\
&\implies hasLoad(car2, load2) && \text{(A10)}\\
&\implies hasLoad(car4, load3) && \text{(A11)}\\
boxShape(Y) \wedge hasCar(Z) \wedge hasLoad(Z, Y) \implies eastBound(Z) & && \text{(A12)}
\end{aligned}
$$

$\mathcal{T}(\mathbb{PS}) = \{eastBound(car1),\ eastBound(car4)\}$

$\mathcal{F}(\mathbb{PS}) = \{eastBound(load1),\ eastBound(car2),\ eastBound(car3)\}$

Glossary

abduction	A repair technique which adds new axioms to prove a previously unprovable goal, introduced in §3.4.4, §4.4.3.
analogical abduction	Seek for an explanation of a given phenomenon by analogically formalising a rule based on existing rules, introduced in §5.2 .
conceptual change	Repair the signature of a logical theory via reformation, introduced in §3.4.3, §4.4.3.1, §4.4.3.
constraint violation	A constraint violation occurs when all the preconditions of a constraint axiom are provable as the theorems of the Datalog theory, Definition 4.2.1.
constraint axiom	A constraint axiom is a rule without a head in a Datalog theory, Definition 3.1.1 .
epistemic entrenchment	Describe the informational value of axioms: The more entrenched an axiom is, the more valuable it is, introduced in §3.4.2, §6.2.
grammar of a Datalog theory	Provide the format of terms, propositions, assertions, rules, constraint axioms, goal clauses and the empty clause, given by Definition 3.1.1.
Herbrand structure	Interpret the Herbrand base by defining a subset of the Herbrand base as true, Definition 3.2.3.

Herbrand model	A Herbrand structure that is a model of the theory, Definition 3.2.4.
Herbrand base	A set of all possible ground propositions written in the signature of a Datalog theory, Definition 3.2.2.
Herbrand universe	The set of all constants in the signature of a Datalog theory, Definition 3.2.1.
incompatibility	A fault that the theory proves a proposition in the false set of the preferred structure, Definition 4.2.4.
insufficiency	A fault that the theory fails to prove a proposition in the true set of the preferred structure, Definition 4.2.4.
maximal set of commutative repair plans	a collection of most commutative repair plans, Definition 6.3.2.
preferred proposition	a proposition in the true set of the preferred structure.
reformation	a domain-independent repair algorithm for conceptual change, introduced in §3.4.3, §4.4.3.
repair plan	strategy to repair incompatibilities or insufficiencies, shown in Table 5.1 and Table 5.2, respectively.
signature	describe the representation language in which a logical theory is written..
SL-Resolution	The inference rule employed in the ABC Repair System, which tries to derive the empty clause by resolving the left most subgoal with an input axiom .

unique name assumption with exceptions	a mechanism that makes it possible to describe how pairs of syntactically distinct constants are equal or unequal in the context of Datalog, introduced in §6.1, §7.1.

Acronyms

$\mathbb{PS}$	Preferred Structure.
ABC	ABC Repair System.
AR	Automated Reasoning.
CNF	Conjunctive Normal Form.
DL	Description Logic.
E-time	Execution time.
EE	Epistemic Entrenchment.
FOL	First-Order Logic.
GS	Gold Standard.
HOL	Higher-Order Logic.
IL	Informational Loss.
ILP	Inductive Logic Programming.
KG	Knowledge Graph.
MIL	Meta-Interpretive Learning.
ML	Machine Learning.
MSCR	maximal set of commutative repair plans.

NES	Non-Exhaustive Search.
NML	Non-Monotonic Logic.
OWL	W3C Web Ontology Language.
PE	preference entrenchment.
RO	Repair Operation.
RP	Repair Plan.
RS	Resolution Step.
SFR-Time	Single Fault Repair Time.
SL-Resolution	Linear Resolution with Selection Function.
UNAE	Unique Name Assumption with Exceptions.

Bibliography

Alchourrón, C. E. and Makinson, D. (1985). On the logic of theory change: Safe contraction. *Studia logica*, 44(4):405–422.

Antoniou, G. and Van Harmelen, F. (2004). Web ontology language: Owl. In *Handbook on ontologies*, pages 67–92. Springer.

Baader, F., Horrocks, I., Lutz, C., and Sattler, U. (2017). *An Introduction to Description Logic*. Cambridge University Press.

Bancilhon, F. and Ramakrishnan, R. (1988). Performance evaluation of data intensive logic programs. In *Foundations of Deductive Databases and Logic Programming*, pages 439–517. Elsevier.

Bancilhon, F. and Ramakrishnan, R. (1989). An amateur's introduction to recursive query processing strategies. In *Readings in Artificial Intelligence and Databases*, pages 376–430. Elsevier.

Bárány, V., Cate, B. T., Kimelfeld, B., Olteanu, D., and Vagena, Z. (2017). Declarative probabilistic programming with Datalog. *ACM Transactions on Database Systems (TODS)*, 42(4):1–35.

Barr, A. and Feigenbaum, E. A. (2014). *The Handbook of Artificial Intelligenc*. Butterworth-Heinemann.

Battigalli, P. and Siniscalchi, M. (2002). Strong belief and forward induction reasoning. *Journal of Economic Theory*, 106(2):356 – 391.

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.

Benavides, D., Trinidad, P., and Ruiz-Cortés, A. (2005). Automated reasoning on feature models. In *International Conference on Advanced Information Systems Engineering*, pages 491–503. Springer.

Bishop, M. (1999). A breadth-first strategy for mating search. In *International Conference on Automated Deduction*, pages 359–373. Springer.

Bride, H., Dong, J., Dong, J. S., and Hóu, Z. (2018). Towards dependable and explainable machine learning using automated reasoning. In *International Conference on Formal Engineering Methods*, pages 412–416. Springer.

Bridge, J. P., Holden, S. B., and Paulson, L. C. (2014). Machine learning for first-order theorem proving. *Journal of automated reasoning*, 53(2):141–172.

Bundy, A. (1985). *The Computer Modelling of Mathematical Reasoning*. Academic Press Professional, Inc., San Diego, CA, USA.

Bundy, A. (2015). Reformation and the agm postulates. Blue Book Note 1802, Artificial Intelligence Applications Institute, Edinburgh.

Bundy, A. (2019). Proposing experiments that would suggest new observations for the ABC system. Blue Book Note 1851, Artificial Intelligence Applications Institute, Edinburgh.

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

Bundy, A., Philalithis, E., and Li, X. (2020). Modelling virtual bargaining using logical representation change. In *Machine Intelligence 21 workshop*, pages 1135–1149.

Calì, A., Gottlob, G., and Lukasiewicz, T. (2012). A general Datalog-based framework for tractable query answering over ontologies. *Journal of Web Semantics*, 14:57–83.

Cartlidge, E. (2012). Loose cable may unravel faster-than-light result.

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.

Chu, S.-C., Xue, X., Pan, J.-S., and Wu, X. (2020). Optimizing ontology alignment in vector space. *Journal of Internet Technology*, 21(1):15–22.

Consortium, G. O. (2019). The gene ontology resource: 20 years and still going strong. *Nucleic acids research*, 47(D1):D330–D338.

Constable, R. L., Allen, S. F., Bromley, H. M., et al. (1986). *Implementing Mathematics with the Nuprl Proof Development System.* Prentice Hall.

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

De Raedt, L. and Bruynooghe, M. (1992). Belief updating from integrity constraints and queries. *Artificial Intelligence*, 53(2-3):291–307.

Delgrande, J., Schaub, T., Tompits, H., and Wang, K. (2004). A classification and survey of preference handling approaches in nonmonotonic reasoning. *Computational Intelligence*, 20(2):308–334.

Dimou, A., Kontokostas, D., Freudenberg, M., Verborgh, R., Lehmann, J., Mannens, E., Hellmann, S., and Van de Walle, R. (2015). Assessing and refining mappingsto rdf to improve dataset quality. In *International Semantic Web Conference*, pages 133–149. Springer.

Douven, I. (2017). Abduction. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2017 edition.

Doyle, J. (1992). Reason maintenance and belief revision: Foundations vs. coherence theories. *Belief revision*, 29:29–51.

Dubois, D., Lang, J., and Prade, H. (1994). Automated reasoning using possibilistic logic: Semantics, belief revision, and variable certainty weights. *IEEE transactions on knowledge and data engineering*, 6(1):64–71.

Duval, B. (1991). Abduction for explanation-based learning. In *European Working Session on Learning*, pages 348–360. Springer.

Fan, W. and Siméon, J. (2003). Integrity constraints for xml. *Journal of Computer and System Sciences*, 66(1):254–291.

Fernandez, J. A., Grant, J., and Minker, J. (1996). Model theoretic approach to view updates in deductive databases. *Journal of Automated Reasoning*, 17(2):171–197.

Flouris, G., Plexousakis, D., and Antoniou, G. (2006). Evolving ontology evolution. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 14–29. Springer.

Fu, Z. and Malik, S. (2006). On solving the partial MAX-SAT problem. In Biere, A. and Gomes, C. P., editors, *Theory and Applications of Satisfiability Testing - SAT 2006*, pages 252–265, Berlin, Heidelberg. Springer Berlin Heidelberg.

Fuhrmann, A. (1991). Theory contraction through base contraction. *Journal of philosophical logic*, pages 175–203.

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.

Garcia, J. D. F., Beek, W., Martínez-Prieto, M. A., and Arias, M. (2017). Lod-a-lot: A queryable dump of the LOD cloud. In D'Amato, C., Fernández, M., Tamma, V., Lecue, F., Cudré-Mauroux, P., Sequeda, J., Lange, C., and Heflin, J., editors, *The Semantic Web - ISWC 2017*. Springer International Publishing, Cham.

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.

Gärdenfors, P. (2003). *Belief revision*, volume 29. Cambridge University Press.

Gärdenfors, P. and Makinson, D. (1988). Revisions of knowledge systems using epistemic entrenchment. In *Proceedings of the 2nd conference on Theoretical aspects of reasoning about knowledge*, pages 83–95. Morgan Kaufmann Publishers Inc.

Goldfarb, W. D. (1981). The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13(2):225–230.

Gómez, A., Sergio, Iván, Carlos, C., and Ricardo, S. G. (2010). Reasoning with inconsistent ontologies through augmentation. *Applied Artificial Intelligence*, pages 102–148.

Gordon, M. and Melham, T. (1993). Introduction to HOL: a theorem proving environment for higher order logic. *Journal of Functional Programming*.

Gómez, S. A., Chesñevar, C. I., and Simari, G. R. (2010). Reasoning with inconsistent ontologies through argumentation. *Applied Artificial Intelligence*, 24(1-2):102–148.

Haig, B. D. (2013). Analogical modeling: a strategy for developing theories in psychology. *Frontiers in psychology*, 4:348.

Hansson, S. O. (2003). Ten philosophical problems in belief revision. *Journal of logic and computation*, 13(1):37–49.

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

Hintikka, J. (1999). What is abduction? the fundamental problem of contemporary epistemology. In *Inquiry as inquiry: A logic of scientific discovery*, pages 91–113. Springer.

Hobbs, J. R., Stickel, M. E., Appelt, D. E., and Martin, P. (1993a). Interpretation as abduction. *Artificial Intelligence*, 63(1):69 – 142.

Hobbs, J. R., Stickel, M. E., Appelt, D. E., and Martin, P. (1993b). Interpretation as abduction. *Artificial intelligence*, 63(1-2):69–142.

Hodges, W. and Scanlon, T. (2018). First-order model theory. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2018 edition.

Janowicz, K., Haller, A., Cox, S. J., Le Phuoc, D., and Lefrançois, M. (2019). Sosa: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics*, 56:1–10.

Jin, Y. and Thielscher, M. (2008). Reinforcement belief revision. *Journal of Logic and Computation*, 18(5):783–813.

Kaminski, M., Nenov, Y., and Grau, B. C. (2016). Datalog rewritability of disjunctive Datalog programs and non-Horn ontologies. *Artificial Intelligence*, 236:90–118.

Katsuno, H. and Mendelzon, A. O. (1991). Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52(3):263–294.

Keeler, M. and Priss, U. (2013). Toward a Peircean theory of human learning: Revealing the misconception of belief revision. In *International Conference on Conceptual Structures*, pages 193–209. Springer.

Kendall, E. F. and McGuinness, D. L. (2019). Ontology engineering. *Synthesis Lectures on The Semantic Web: Theory and Technology*, 9(1):i–102.

Knublauch, H., Fergerson, R. W., Noy, N. F., and Musen, M. A. (2004). The Protégé OWL plugin: An open development environment for Semantic Web applications. In *International semantic web conference*, pages 229–243. Springer.

Koopmann, P. (2019). Ontology-based query answering for probabilistic temporal data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2903–2910.

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

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Lee, K. and Meyer, T. (2004). A classification of ontology modification. In *Australasian Joint Conference on Artificial Intelligence*, pages 248–258. Springer.

Lehmann, J., Chan, M., and Bundy, A. (2013). A higher-order approach to ontology evolution in physics. *Journal on Data Semantics*, 2(4):163–187.

Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., and Scarcello, F. (2006). The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic (TOCL)*, 7(3):499–562.

Li, X., Bundy, A., and Smaill, A. (2018). ABC repair system for Datalog-like theories. In *KEOD*, pages 333–340.

Liebig, T. and Noppens, O. (2004). Ontotrack: Combining browsing and editing with reasoning and explaining for owl lite ontologies. In *International Semantic Web Conference*, pages 244–258. Springer.

McDermott, D. and Doyle, J. (1980). Non-monotonic logic I. *Artificial Intelligence*, 13(1-2):41–72.

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.

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

Motik, B., Nenov, Y., Piro, R., and Horrocks, I. (2019). Maintenance of Datalog materialisations revisited. *Artificial Intelligence*, 269:76–136.

Muggleton, S. and De Raedt, L. (1994). Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19, 20:629–679.

Muggleton, S. H. (2015). Learning efficient logical robot strategies involving composable objects. In *Proceedings of the 24th International Joint Conference Artificial Intelligence (IJCAI 2015)*, pages 3423–3429.

Muggleton, S. H. (2017). Meta-interpretive learning: achievements and challenges. In *International Joint Conference on Rules and Reasoning*, pages 1–6. Springer.

Nebel, B. (1991). Belief revision and default reasoning: Syntax-based approaches. *KR*, 91:417–428.

Nebel, B. (2001). Logics for knowledge representation. *International Encyclopedia of the Social and Behavioral Sciences, Kluwer, Dordrecht.*

Nipkow, T., Paulson, L. C., and Wenzel, M. (2002). *Isabelle/HOL: a proof assistant for higher-order logic*, volume 2283. Springer Science & Business Media.

Nuamah, K. and Bundy, A. (2018). Calculating error bars on inferences from web data. In *Proceedings of SAI Intelligent Systems Conference*, pages 618–640. Springer.

Owre, S., Rajan, S., Rushby, J. M., Shankar, N., and Srivas, M. (1996). Pvs: Combining specification, proof checking, and model checking. In *International Conference on Computer Aided Verification*, pages 411–414. Springer.

Paulson, L. (1990). Isabelle: the next 700 theorem provers. In Odifreddi, P., editor, *Logic and Computer Science*, pages 77–90. Academic Press.

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

Plotkin, G. D. (1971). *Automatic Methods of Inductive Inference*. PhD thesis, University of Edinburgh.

Poole, D. L. and Mackworth, A. K. (2010). *Artificial Intelligence: foundations of computational agents*. Cambridge University Press.

Pople, H. E. (1973). On the mechanization of abductive logic. In *IJCAI*, volume 73, pages 147–152. Citeseer.

Prendinger, H., Ishizuka, M., and Yamamoto, T. (2000). The hyper system: Knowledge reformation for efficient first-order hypothetical reasoning. In *Pacific Rim International Conference on Artificial Intelligence*, pages 93–103. Springer.

Qian, Z. (1993). Linear unification of higher-order patterns. In *Colloquium on Trees in Algebra and Programming*, pages 391–405. Springer.

Raad, J., Beek, W., Van Harmelen, F., Pernelle, N., and Saïs, F. (2018). Detecting erroneous identity links on the web using network metrics. In *International semantic web conference*, pages 391–407. Springer.

Rector, A., Nowlan, W., and Glowinski, A. (1993). Goals for concept representation in the galen project. In *Proceedings of the Annual Symposium on Computer Application in Medical Care*, page 414. American Medical Informatics Association.

Reggia, J. A., Nau, D. S., and Wang, P. Y. (1983). Diagnostic expert systems based on a set covering model. *International journal of man-machine studies*, 19(5):437–460.

Rodler, P. and Eichholzer, M. (2019). On the usefulness of different expert question types for fault localization in ontologies. In Wotawa, F., Friedrich, G., Pill, I., Koitz-Hristov, R., and Ali, M., editors, *Advances and Trends in Artificial Intelligence. From Theory to Practice*, pages 360–375, Cham. Springer International Publishing.

Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice-Hall International, London.

Sakama, C. and Inoue, K. (2003). An abductive framework for computing knowledge base updates. *Theory and Practice of Logic Programming*, 3(6):671–715.

Samek, W., Montavon, G., Vedaldi, A., Hansen, L. K., and Müller, K.-R. (2019). *Explainable AI: interpreting, explaining and visualizing deep learning*, volume 11700. Springer Nature.

Sanfilippo, E. M., Belkadi, F., and Bernard, A. (2019). Ontology-based knowledge representation for additive manufacturing. *Computers in Industry*, 109:182–194.

Schmidt-Schauß, M. and Smolka, G. (1991). Attributive concept descriptions with complements. *Artificial intelligence*, 48(1):1–26.

Schurz, G. (2008). Patterns of abduction. *Synthese*, 164(2):201–234.

Shapiro, E. (1982). *Algorithmic Program Debugging*. PhD thesis, Yale University. Also available as Computer Science Research Report no. 237.

Shkapsky, A., Yang, M., Interlandi, M., Chiu, H., Condie, T., and Zaniolo, C. (2016). Big data analytics with Datalog queries on spark. In *Proceedings of the 2016 International Conference on Management of Data*.

Smullyan, R. M. (1992). *Godel's incompleteness theorems*, volume 19. Oxford University Press.

Smullyan, R. M. (1995). *First-order logic*. Courier Corporation.

Strasser, C. and Antonelli, G. A. (2019a). Non-monotonic logic. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2019 edition.

Strasser, C. and Antonelli, G. A. (2019b). Non-monotonic logic. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2019 edition.

Svatoš, M., Šourek, G., Železnỳ, F., Schockaert, S., and Kuželka, O. (2017). Pruning hypothesis spaces using learned domain theories. In *International Conference on Inductive Logic Programming*, pages 152–168. Springer.

Thagard, P. R. (1978). The best explanation: Criteria for theory choice. *The journal of philosophy*, 75(2):76–92.

Tsialos, A. (2015). Repairing inconsistent description logic ontologies using reformation,. *UG4 Final Year Project, University of Edinburgh, UK*.

Ullman, J. D. (1985). Implementation of logical query languages for databases. *ACM Transactions on Database Systems (TODS)*, 10(3):289–321.

Urban, J., Vyskočil, J., and Štěpánek, P. (2011). Malecop machine learning connection prover. In *International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 263–277. Springer.

Urbonas, M., Bundy, A., Casanova, J., and Li, X. (2020). The use of max-sat for optimal choice of automated theory repairs. In Bramer, M. and Ellis, R., editors, *Artificial Intelligence XXXVII*, pages 49–63, Cham. Springer International Publishing.

Van Harmelen, F., Lifschitz, V., and Porter, B. (2008). *Handbook of knowledge representation*. Elsevier.

van Harmelen, F., van den Besselaar, P., et al. (2018). Network metrics for assessing the quality of entity resolution between multiple datasets. In *European Knowledge Acquisition Workshop*, pages 147–162. Springer.

Visser, U., Stuckenschmidt, H., Schuster, G., and Vögele, T. (2002). Ontologies for geographic information processing. *Computers & Geosciences*, 28(1):103–117.

Wiharja, K., Pan, J. Z., Kollingbaum, M., and Deng, Y. (2018). More is better: Sequential combinations of knowledge graph embedding approaches. In Ichise, R., Lecue, F., Kawamura, T., Zhao, D., Muggleton, S., and Kozaki, K., editors, *Semantic Technology*, pages 19–35, Cham. Springer International Publishing.

Wikipedia contributors (2017). Herbrand structure — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Herbrand_structure&oldid=800715701`. [Online; accessed 5-February-2019].

Wos, L., Overbeek, R., Lusk, E., and Boyle, J. (1984). *Automated Reasoning: Introduction and Applications*. Prentice-Hall.

Wrobel, S. (1994). Concept formation during interactive theory revision. *Machine Learning*, 14(2):169–191.

Zhang, S., Sridharan, M., Gelfond, M., and Wyatt, J. (2014). Towards an architecture for knowledge representation and reasoning in robotics. In *International conference on social robotics*, pages 400–410. Springer.