

Automating Theory Repair in First Order Logic

Wan Ki Wong¹, Xue Li¹ and Alan Bundy¹

¹*School of Informatics, The University of Edinburgh, United Kingdom*

Abstract

Automatic theory repair systems help identify and repair faults in a knowledge base, which has useful applications in artificial intelligence such as decision systems. The ABC system is a state-of-the-art implementation of such systems which combines three existing techniques: abduction, belief revision and conceptual change, but with a limitation that it only accepts Datalog logic. To enhance its expressive power, this study extends the ABC system to first-order logic (ABC_FOL), by augmenting the fault detection module and adding new repair plans to the system. The resultant extended system is able to correctly identify faults and generate sensible repairs across a diverse set of first-order logic examples that cannot be expressed in Datalog logic.

Keywords

automated theory repair, abduction, belief revision, conceptual change, reformation, first-order logic

1. Introduction

Automated theory repair is a subfield of AI focused on rectifying errors in logical theories or knowledge bases. These theories are crucial for AI tasks like reasoning, planning, and learning [1]. Automated reasoning is gaining importance in applications like autonomous vehicles, where safety and accuracy are vital. Unlike machine learning models, reasoning systems are preferred for their reliability and explainability [2].

Logical theories represent an agent's environment, such as traffic rules for autonomous vehicles. These theories can become faulty, for instance, due to changes in the environment or new objectives [3], like when new traffic laws are introduced that requires the agent to reason and behave differently than before.


The ABC repair system by Li and Bundy [4, 3] addresses these issues by integrating abduction, belief revision, and conceptual change in Prolog to repair theories. ABC takes a *Theory* (\mathbb{T}) and a *Preferred Structure* (\mathbb{PS}) consisting of a true set and false set as input. If the theory fails in proving any proposition in the true set, it is insufficient. If the theory proves any proposition in the false set, it is incompatible. ABC system repairs these two kinds of faults and outputs a set of fault-free theories (which might be empty). \mathbb{PS} contains true and false sets, which represents the propositions that should be provable or unprovable by the theory respectively. The pipeline is shown in Figure 1.


However, ABC was initially designed only for Datalog-like theories, which excludes negations, functions and existential quantification. Rules are also restricted to be Horn clauses. When

Cognitive AI 2023, 13th-15th November, 2023, Bari, Italy.

✉ s2447989@ed.ac.uk (W. K. Wong); xli3310@ed.ac.uk (X. Li); A.Bundy@ed.ac.uk (A. Bundy)

ORCID 0009-0002-1555-7226 (W. K. Wong); 0000-0002-6665-2242 (X. Li); 0000-0002-0578-6474 (A. Bundy)

 © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

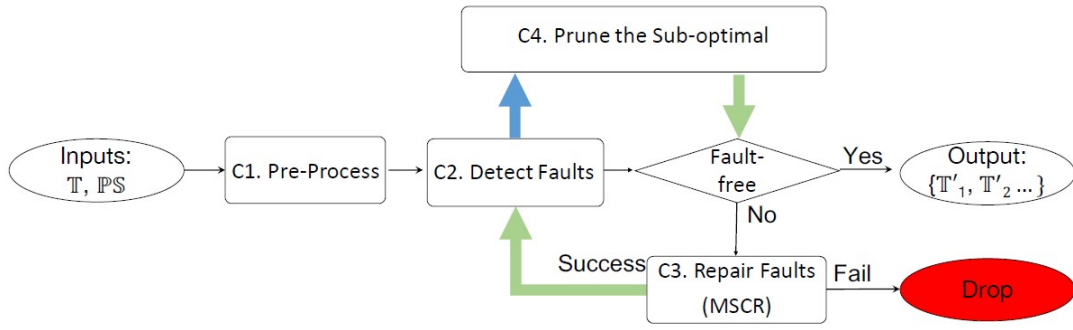


Figure 1: Pipeline of the ABC model [4].

considering real-world scenarios, some complex rules (such as Government laws and regulations) simply cannot be encoded in Datalog logic. It is hence important to consider first-order logic (FOL), a more expressive logic that allows translation of almost any natural language statement to FOL statements [5]. This project aims to extend ABC to support FOL while retaining domain-independence. The hypothesis is that *ABC techniques can be adapted to repair FOL theories, offering some potential repairs, though not all of them.*

Referring to figure 1, the project focuses on modifying C2 and C3 of ABC, as follows: 1. Developing an ABC theorem prover for FOL, generating partial proofs for repairs (C2). 2. Extending existing repair strategies to accommodate FOL (C3). 3. Creating new FOL-specific repair strategies (C3).

2. Extension of Fault Detection (C2)

2.1. Ancestor Resolution

Ancestor resolution involves resolving the current clause with one of its own ancestors [6]. This concept is used in SL-resolution, where ancestors refer to intermediate goal clauses derived through resolution steps (RS). Incorporating ancestor resolution in FOL is essential due to its ability to handle non-Horn clauses, enabling new rules and theorems to emerge during the resolution process. This adaptability proves crucial in correctly deriving desired goals.

Consider the following example:

$$\begin{aligned}
 &like(X, hiking) \implies like(X, running). \\
 &like(X, running) \implies like(X, hiking). \\
 &\implies like(george, running) \vee like(george, hiking)
 \end{aligned}$$

Figure 2 shows the inference result of $like(george, running)$. The final resolution step, as highlighted in green, reuses the derivation result of the initial step (red) to prove the desired result. Without ancestor resolution, we would not be able to prove the result.

$$\begin{array}{c}
\frac{\text{like}(\text{george}, \text{running}) \implies}{\implies \text{like}(\text{george}, \text{hiking})} \implies \text{like}(\text{george}, \text{running}) \vee \text{like}(\text{george}, \text{hiking}) \\
\frac{\implies \text{like}(\text{george}, \text{hiking})}{\implies \text{like}(\text{george}, \text{running})} \text{like}(X, \text{hiking}) \implies \text{like}(X, \text{running}) \\
\implies \text{like}(\text{george}, \text{running}) \implies \text{like}(\text{george}, \text{running})
\end{array}$$

Figure 2: The inference of $\text{like}(\text{george}, \text{running})$

2.2. Occurs Check

The occurs check is crucial for inference soundness, preventing cyclic substitutions where a variable is bound to a term containing itself, e.g., $f(Y)/Y$ [6]. Unlike the prior ABC version, the new FOL context introduces functions, rendering the occurs check necessary. This issue arises only when binding a variable to a function-containing a term, not a constant.

Consider the illustrative example:

$$\text{data}(X, X) \equiv \text{data}(Y, \text{model}(Y))$$

The sign \equiv represents a unification problem. Here, unifying the second argument encounters the challenge $Y \equiv \text{model}(Y)$, causing the occurs check to fail due to cyclic substitution.

The project reinstates the occurs check into the unification algorithm. The condition $x \in \mathcal{V}(s)$ is verified within the *Occurs* case of the standard unification. Readers can refer to [6] for details of the occurs check.

3. Extension of Repair Generation (C3)

A summary of old and new repair plans in ABC_FOL are listed in table 1, appendix A. The description of the changes are as follows.

3.1. Extension of ABC's Framework

ABC's framework needs to be modified for FOL, including a new trace-back, incorporating functions, and removing restrictions of Datalog.

A repair is generated to block an unwanted unification or unblock a wanted unification in a resolution process, which has to modify a source axiom that is originally from the input theory rather than a theorem that is derived during resolution. Otherwise, the fault will still be derivable from the source axioms.

Thus, a trace-back algorithm is pivotal in pinpointing the source axiom for a targeted unification from a proof. This necessity arises from ancestor resolution. The accurate change propagation to the pertinent axiom hinges on the trace-back process.

In order to incorporate functions, we adapted ABC's framework to accept functions and be able to accurately identify constants, variables, predicates and functions.

Finally, a number of restrictions are modified in each specific old repair plan, that targets FOL's distinction from Datalog - non-Horn clauses, allowing orphan variables (variables that are solely present in the head but not the body), predicates with function arguments.

3.2. New Repair Plans

A brief summary of each of the repair plans is provided as follows. The first three fix an incompatibility that a proposition in the false set of preferred structure is derived so that we want to break a unification in its proof. The last is for an insufficiency that a proposition in the true set of the preferred structure cannot be derived, where we want to fix the failed unification to build a proof for it. Readers are directed to the project's github repository [7] for a detailed description of the repair plans.

CR7-10: Break unification of a function: These repairs break unification of two functions by renaming functions, renaming constant arguments, weakening variables or adding different constants, which inherently breaks the predicate-level unification. The repair plans accommodate functions with various nesting depths.

CR11: Break $p_1(\vec{S}^n) \equiv p(\vec{T}^n)$ by adding a variable Y to k : This repair breaks unification by failing the occurs check, which is achieved by adding a variable that would cause cyclic substitution.

CR12: Break $p_1(\vec{S}^n) \equiv p(\vec{T}^n)$ by adding an unprovable alternative $q(\vec{Z}^w)$: Appends an unprovable positive literal q to the original axiom that satisfies certain conditions, effectively breaking the unification.

SR6: Fix the failed unification $s_i \neq t_i$ by removing all occurrences of a variable X : Fixing a failed occurs check by removing all occurrences of the variable that leads to a cyclic substitution.

An example scenario that utilizes the new repair plans is a repair of an erroneous mathematical equation. Consider the equation: $\exists Z, \forall Y. Y \neq Z$. This equation can be formulated in ABC_FOL in the following form: $\neg eq(Y, c)$. This causes a contradiction with the fact $eq(X, X)$, which is also added as an axiom in ABC_FOL. The new repair plan, CR11, is able to remedy this by introducing a new variable to the constant c , changing equation $\neg eq(Y, c)$ to the following:

$$\neg eq(Y, dummyc1(Y)) \quad (1)$$

This reorders the quantifiers into $\forall Y, \exists Z. Y \neq Z$, which now holds true.

Note that *dummyc1* is a function yet to be assigned any meaning, as in any functions which arise from skolemization. In many applications, we need to assign values to the new functions for the logical system to behave correctly. Suppose that, based on other observations, a new rule $dummyc1(Z) = Z$ is added to the theory - this raises yet another contradiction with the rule (1).

Now, CR7 is able to rename the function *dummyc1* in (1) further into *dummydummyc1*:

$$\neg eq(Y, dummydummyc1(Y)) \quad (2)$$

This makes sure the function in equation (1) stands different from other occurrences of *dummyc1* in the theory.

4. Case Study

A precise definition of a general polyhedron has long been argued as there are multiple prevailing ones [8]. A common definition requires a polyhedron to be formed of four or more polygons. Consider the following (faulty) definition of a polygon (vs and ls are the set of vertices and lines respectively):

$$(\forall v \in vs. \exists l_1, l_2 \in ls. l_1 \neq l_2 \wedge meet(v, l_1, l_2)) \implies polygon(n, ls, vs) \quad (3)$$

Equation 1 allows the “*eggtimer*”, as defined in Appendix B, to be a polygon. Suppose we do not want the *eggtimer* to count as a polygon as it cannot be properly extended to form a polyhedron - Lakatos [8] introduced a concept termed “*monster barring*” to exclude counter-examples like this. In ABC_FOL, we can mimic monster barring by the following formulation in clausal form:

$$\neg vs(x) \vee \neg ls(A) \vee \neg ls(B) \vee A = B \vee \neg meet(x, A, B) \vee polygon(eggtimer) \quad (4)$$

$$\neg meet(P, L_1, L_2) \vee vs(P) \quad (5)$$

Equation (4) is the clausal form of equation (3), which was not possible to be formulated with ABC_Datalog. Equation (3) is a wrong formulation which causes the point x , a point that is not in the vertex set, to be wrongly included in the vertex set. The remaining equations that are not shown include the definition of all *meet* points and all lines in the set ls , as referenced from Appendix B. The only item in the false set of this example is $polygon(eggtimer)$, which is the target proposition to block, while the true set is empty.

Passing this formulation to FOL_ABC, a total of 56 repair plans are generated, spanning the use of various repair plans in repairing incompatibility. One of the solution uses the repair CR6 to add an unprovable precondition to (3), as follows:

$$\neg dummyPred(P) \vee meet(P, L_1, L_2) \vee vs(P) \quad (6)$$

Given that the above formulation has no other reasonable candidates for the unprovable precondition, the predicate *dummyPred* is used. This could be linked to some specific mathematical definitions that establish point p as a vertex.

5. Conclusion

This paper extends the theory repair system ABC from Datalog to accommodate first-order logic. The proposed enhancements involve introducing ancestor resolution, an occurs check to the fault detection module and some framework adaptations for incorporating functions. Furthermore, novel repair strategies are devised to handle FOL’s specific characteristics, including functions and non-Horn clauses.

The extension is successfully integrated into the ABC codebase¹. The evaluation of ABC_FOL supports the research hypothesis. The system generates numerous potential repairs to rectify identified faults while ensuring semantic coherence. ABC_FOL is poised for diverse applications

¹Github Link: https://github.com/tpmmthomas/ABC_FOL

in decision systems, law enforcement, and knowledge graphs, offering broader usability than ABC_Datalog.

The project encountered several limitations, including the limited literature on automated theory repair techniques, inadequate data availability for FOL theorems, and evaluation constraints due to the subjective nature of repair output assessment. Future work suggestions include allowing non-ground assertions in \mathcal{PS} to handle more complex statements, designing tailored heuristics for FOL theories to improve efficiency, exploring sorted logic extension for better proof search guidance, and conducting a more thorough evaluation of ABC_FOL.

References

- [1] Chapter xii - automatic deduction, in: P. R. Cohen, E. A. Feigenbaum (Eds.), *The Handbook of Artificial Intelligence*, Butterworth-Heinemann, 1982, pp. 75–123. URL: <https://www.sciencedirect.com/science/article/pii/B9780865760912500071>. doi:<https://doi.org/10.1016/B978-0-86576-091-2.50007-1>.
- [2] S. Kothawade, V. Khandelwal, K. Basu, H. Wang, G. Gupta, AUTO-DISCERN: autonomous driving using common sense reasoning, CoRR abs/2110.13606 (2021). URL: <https://arxiv.org/abs/2110.13606>. arXiv:2110.13606.
- [3] A. Bundy, X. Li, Representational change is integral to reasoning, *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* (2023).
- [4] X. Li, A. Bundy, An overview of the abc repair system for datalog-like theories, in: A. Bundy, D. Mareschal (Eds.), *Proceedings of 3rd International Workshop on Human-Like Computing HLC2022 @ IJCLR*, volume 3227 of *Human-Like Computing Workshop 2022*, CEUR Workshop Proceedings (CEUR-WS.org), 2022, pp. 11–17. URL: <https://ijclr22.doc.ic.ac.uk/hlc2022.html/index.html>.
- [5] J. Barwise, An introduction to first-order logic, in: J. Barwise (Ed.), *HANDBOOK OF MATHEMATICAL LOGIC*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, Elsevier, 1977, pp. 5–46. URL: <https://www.sciencedirect.com/science/article/pii/S0049237X08710978>. doi:[https://doi.org/10.1016/S0049-237X\(08\)71097-8](https://doi.org/10.1016/S0049-237X(08)71097-8).
- [6] A. Bundy, *The Computer Modelling of Mathematical Reasoning*, Academic Press Professional, Inc., USA, 1985.
- [7] T. W. (forked from Xue Li), *Abc_fol*, https://github.com/tpmmthomas/ABC_FOL, 2023.
- [8] I. Lakatos, *Proofs and Refutations: The Logic of Mathematical Discovery*, New York: Cambridge University Press, 1976.
- [9] A. Bundy, An ambiguous polygon, Blue Book Note 1887, 2023.

A. Repair Plans Summary

Table 2 shows a summary of all the repair plans (old and new) in ABC_FOL.

Table 1

A summary of repair plans: Unification is denoted by \equiv , '=' denotes equality between terms or symbols, p_1 and p_2 are predicates, S and T are arguments of a predicate (constants, variables or functions), c is a constant, X is a variable, f_1 and f_2 are functions, x and y are arguments of a function (constants, variables or functions), k is either a constant or a function (but not variable). The left of the unification comes from the sub-goal, while the right comes from an input clause. Text highlighted in red denotes new repair plans, the background colour denotes sets of repair plans which are self-inverse.

RS	Target match / mismatch	Repair Plan	Technique	
$p_1(\vec{S}^n) \equiv p_2(\vec{T}^n)$ (Incompatibility)	$p_1 = p_2$	CR1. Rename predicate on either side: $p_1(\vec{S}^n) \neq p_2(\vec{T}^n)$ or $p_1(\vec{S}^n) \neq p_2(\vec{T}^n)$.	Reformation	
	$s_i \equiv t_i \equiv c$	CR2. Rename s_i or t_i to c' .		
	$(s_i = X \wedge t_i = c) \vee (s_i = c \wedge t_i = X)$	CR3. Weaken variable X to c' .	Belief Revision	
		CR4. Add different constants: $p_1(\vec{S}^n, c_1) \neq p_2(\vec{T}^n, c_2)$.		
		CR5. Delete the axiom A_u .	Belief Revision (Variant)	
			CR6. Add an unprovable precondition $q(\vec{Z}^w)$ to A_u .	Reformation
		$f_1 = f_2$	CR7. Rename predicate on either side: $f_1(\vec{x}) \neq f_2(\vec{y})$ or $f_1(\vec{x}) \neq f_2(\vec{y})$.	
	$s_i = f_1(\vec{x})$	$x_i = y_i = c$	CR8. Rename x_i or y_i to c' .	
	$t_i = f_2(\vec{y})$	$(x_i = X \wedge y_i = c) \vee$ $(x_i = c \wedge y_i = X)$	CR9. Weaken variable X to c' .	
	$s_i \equiv t_i$		CR10. Add different constants: $f_1(\vec{x}, c_1) \neq f_2(\vec{y}, c_2)$.	
	$(s_i = X \wedge t_i = k) \vee (s_i = k \wedge t_i = X)$		CR11. Add variable Y to k .	
	$p_1(\vec{S}^n) \neq p_2(\vec{T}^m)$ (Insufficiency)		CR12. Add an unprovable alternative $q(\vec{Z}^w)$ to A_u .	Belief Revision (Variant)
$p_1 = p_2, s_i \neq t_i$		SR1. Reform either $p_2(\vec{T}^m)$ or $p_1(\vec{S}^n)$ to the other. SR2. Extend t_i to variable Z .	Reformation	
		SR3. Add the assertion $p_1(\vec{S}^n)$. SR4. Add a rule which proves $p_1(\vec{S}^n)$.	Abduction	
		SR5. Delete $p_1(\vec{S}^n)$ from original input axiom. SR6. Remove all occurrences of variable Y in k .	Abduction (Var.) Reformation	

B. Definition of Eggtimer

The *eggtimer*, which is the target polygon to block, can be formalized by the set $ls = \{l_1, l_2, l_3, l_4\}$, $vs = \{v_1, v_2, v_3, v_4\}$, and the meeting points $meet(v_1, l_1, l_2)$, $meet(v_2, l_2, l_3)$, $meet(v_3, l_3, l_4)$, $meet(v_4, l_4, l_1)$, $meet(x, l_1, l_3)$. A graphical illustration is provided as follows.

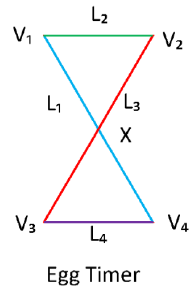


Figure 3: Visualization of eggtimer [9]