ABC in Root Cause Analysis: Discovering Missing Information and Repairing System Failures

ABC in Root Cause Analysis: Discovering Missing Information and Repairing System Failures

Xue Li[1], Alan Bundy[1], Ruiqi Zhu [1], Fangrong Wang [1]
Stefano Mauceri[2], Lei Xu[2], and Jeff Z. Pan[1]

[1]School of Informatics, University of Edinburgh, UK;
[2]Huawei Ireland Research Centre
{Xue.Shirley.Li, A.Bundy, Ruiqi.Zhu, Sylvia.Wang, J.Z.Pan}@ed.ac.uk,
{Stefano.Mauceri1, xulei139}@huawei.com

Abstract. Root-cause analysis (RCA) is a crucial task in software system maintenance, where system logs play an essential role in capturing system behaviours and describing failures. Automatic RCA approaches are desired, which face the challenge that the knowledge model (KM) extracted from system logs can be faulty when logs are not correctly representing some information. When unrepresented information is required for successful RCA, it is called missing information (MI). Although much work has focused on automatically finding root causes of system failures based on the given logs, automated RCA with MI remains under-explored. This paper proposes using the Abduction, Belief Revision and Conceptual Change (ABC) system to automate RCA after repairing the system's KM to contain MI. First, we show how ABC can be used to discover MI and repair the KM. Then we demonstrate how ABC automatically finds and repairs root causes. Based on automated reasoning, ABC considers the effect of changing a cause when repairing a system failure: the root cause is the one whose change leaves the fewest failures. Although ABC outputs multiple possible solutions for experts to choose from, it hugely reduces manual work in discovering MI and analysing root causes, especially in large-scale system management, where any reduction in manual work is very beneficial. This is the first application of an automatic theory repair system to RCA tasks: KM is not only used, it will be improved because our approach can guide engineers to produce KM/higher-quality logs that contain the spotted MI, thus improving the maintenance of complex software systems.

Keywords: Root Cause Analysis · Missing Information · System management · Automatic Theory Repair.

1 Introduction

Software system failures are unavoidable, so fast diagnosis and repair are crucial in system maintenance, where root cause analysis (RCA) is required. Traditionally, experts manually analyse raw system logs for RCA, where the challenges

include that 1) the first arisen failure is usually not the root cause; 2) single causes can trigger multiple failures; 3) the system is enormous. Thus, manual RCA is difficult; 4) the model of the system may be inaccurate so that automatic methods become unreliable.

Much work has focused on automatically mining logs and assisting experts in discovering the root cause of system failures, including log filtering that collects the most relevant logs [20], log extraction as knowledge graphs (KG) [19], clustering logs [15, 11], mining and representing information from logs [6, 8]; automating network management in software based on KG [21] and analysing causality patterns among components in a software system [12, 14, 3], where the last takes extra tests to learn and validate dependencies by experts manually. It can be seen that log-based RCA is popular because logs are arguably the most straightforward source of information about the system. However, these log-based RCA methods' performance is restricted by the quality of logs, e.g., whether they cover all essential information for diagnosing root causes and whether they are written in ways that data-driven RCA pipelines can effectively consume.

The higher quality the log is, the more accurate the corresponding model of the system built from that log is. If the symptoms of the root cause can be inferred from the current model of the system, then this model is seen accurate and we can analyse that inference to discover the root cause. Otherwise, the current model is inaccurate, which is missing some crucial information, i.e. MI, about the current state of the system.

We argue that in time, continuous improvement of system logs, in the sense of incorporating MI[1] as necessary, could significantly enhance the performance of automated RCA pipelines or any other task related to log-mining. In manual RCA, experts sometimes need extra system tests to identify MI. Thus, an automated RCA not only needs to be aware of the MI in the system logs, but also should account for the domain experts' knowledge and experience [17].

Accounting for these considerations, the Abduction, Belief Revision and Conceptual Change system (ABC) [10, 16], which repairs faulty logical theories based on a given benchmark, is here employed to automate RCA[2]. The main input is the system's KM that comprises of: 1) an automatically extracted KG from both system logs and the manual, and 2) rules manually formalised to introduce domain knowledge. Based on the KM and the given observed system failures, ABC discovers MI first and repairs the system model so that it predicts the failures, and then ABC analyses the repaired model to find the root cause. When there are multiple possible MI, ABC provides all possibilities to experts so that the correct MI can be found interactively.

This paper demonstrates the first application of a theory repair system to RCA tasks. The main contribution includes the follows:

1. An approach to discover log MI that is instrumental to RCA and the fault recovery process.

[1] The damage caused by MI in RCA is described in Figure 2 and further discussed in the next section.

[2] ABC's code is available on GitHub `https://github.com/XuerLi/ABC_Datalog`.

2. An approach to automatically identify the root cause and suggest data-driven repairs.
3. An approach to guide experts to enrich the KM or to extend the system logs as to continuously improve the performance of the RCA pipeline.

Essential definitions of the ABC repair mechanism are given in §2. Root-cause analysis is introduced in §3, where MI is repaired by ABC in §3.1 first, and then the discovering and repairing of root causes are discussed in §3.2. An initial evaluation is given in §4, followed by the conclusion in §5.

2 ABC Repair Mechanism

ABC represents environments using logical theories based on the DataLog logic programming language [1], where axioms are Horn clauses. In *Kowalski Form*, these clauses take one of the following forms, i.e.,

$$Q_1 \wedge \ldots \wedge Q_m \implies P \quad (1)$$
$$Q_1 \wedge \ldots \wedge Q_m \implies \quad (2)$$
$$\implies P \quad (3)$$
$$\implies \quad (4)$$

where m is a natural number; $Q_j,\ 1 \leq j \leq m$ and P are propositions. Then the above clauses represent a rule, a goal of m sub-goals, an assertion, and the empty clause, respectively[3].

In DataLog, the arguments of propositions are either constants or variables, i.e., there are no non-nullary functions. This makes Selected Literal Resolution (SL) [9] with a fair search strategy a decision procedure for DataLog theories. Decidability is important for establishing certainty in the detection of faults. Example 1 in §4 illustrates a DataLog theory. Note that the $\implies$ arrow is retained even when $m = 0$.

Figure 1 shows ABC's workflow. The inputs to ABC are a Datalog theory $\mathbb{T}$ and the preferred structure $\mathbb{S}$ which consists of a pair of sets of ground propositions: those propositions that are observed to be true $\mathcal{T}(\mathbb{S})$ and those observed to be false $\mathcal{F}(\mathbb{S})$. The pre-process in C1 reads and rewrites inputs into the internal format for later use. Then in C2, ABC applies SL to $\mathbb{T}$ to detect incompatibility and insufficiency faults based on $\mathcal{F}(\mathbb{S})$ and $\mathcal{T}(\mathbb{S})$, defined below. Incompatibilities are conflicts between the theory and the observations and insufficiencies are the failure of the theory to predict observations.

Definition 1 (Types of Fault)**.** *Let $\mathbb{T}$ be a DataLog theory.*

Incompatibility: $\exists\phi.\ \mathbb{T} \vdash \phi \wedge \phi \in \mathcal{F}(\mathbb{S})$*;*
Insufficiency: $\exists\phi.\ \mathbb{T} \nvdash \phi \wedge \phi \in \mathcal{T}(\mathbb{S})$

[3] Keeping $\implies$ is required by the inference of refutation.

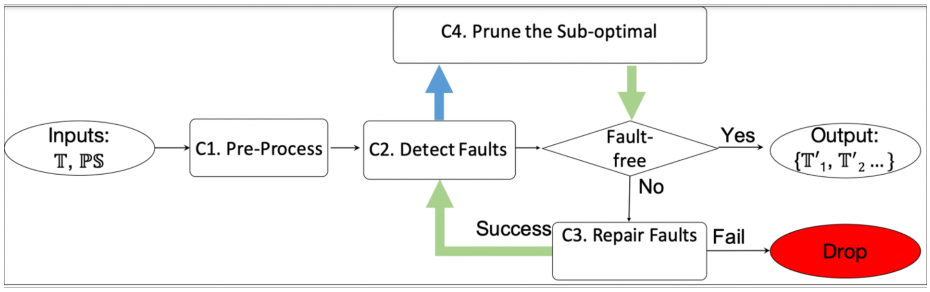

Fig. 1. Flowchart of the ABC: green arrows deliver a set of theories one by one to the next process; the blue arrow collects and delivers theories as a set; When a faulty-theory is not repairable, it will be dropped from the repair process.

As ABC uses SL-resolution[9] which is not only sound and complete [5], but also decidable [13] for Datalog theories so that proofs can always be detected if there are any.

In C3, repairs are generated to fix detected faults. An insufficiency is repaired by unblocking a proof with additional necessary SL steps, while an incompatibility is repaired by blocking all its proofs, which can be done by breaking one SL step in each of them [16]. ABC repairs faulty theories using eleven *repair operations*. There are five for repairing incompatibilities and six for repairing insufficiencies, defined below.

Definition 2 (Repair Operations for Incompatibility)**.** *In the case of incompatibility, the unwanted proof can be blocked by causing any of the SL steps to fail. Suppose the targeted SL step is between a goal,* $P(s_1, \ldots, s_n)$*, and an axiom,* $Body \implies P(t_1, \ldots, t_n)$*, where each* s_i *and* t_i *pair can be unified. Possible repair operations are as follows:*

Belief Revision 1: *Delete the targeted axiom:* $Body \implies P(t_1, \ldots, t_n)$.

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

Reformation 3: *Rename* P *in the targeted axiom to either a new predicate or a different existing predicate* P'.

Reformation 4: *Increase the arity of all occurrences* P *in the axioms by adding a new argument. Ensure that the new arguments in the targeted occurrence of* P*, are not unifiable. In Datalog, this can only be ensured if they are unequal constants at the point of unification.*

Reformation 5: *For some* i*, suppose* s_i *is* C*. Since* s_i *and* t_i *unify,* t_i *is either* C *or a variable. Change* t_i *to either a new constant or a different existing constant* C'.

Definition 3 (Repair Operations for Insufficiency)**.** *In the case of insufficiency, the wanted but failed proof can be unblocked by causing a currently failing SL step to succeed. Suppose the chosen SL step is between a goal* $P(s_1, \ldots, s_m)$ *and*

an axiom Body $\implies P'(t_1, \ldots, t_n)$*, where either* $P \neq P'$ *or for some* i*,* s_i *and* t_i *cannot be unified. Possible repair operations are:*

Abduction 1: *Add the goal* $P(s_1, \ldots, s_m)$ *as a new assertion and replace variables with constants.*

Abduction 2: *Add a new rule whose head unifies with the goal* $P(s_1, \ldots, s_m)$ *by analogising an existing rule or formalising a precondition based on a theorem whose arguments overlap with the ones of that goal.*

Abduction 3: *Locate the rule axiom whose precondition created this goal and delete this precondition from the rule.*

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

Reformation 5: *Suppose* s_i *and* t_i *are not unifiable. Decrease the arity of all occurrences* P' *by 1 by deleting its* i^{th} *argument.*

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

ABC has a protection heuristic that allows the user to specify any term that should be protected from being changed. Usually a faulty theory requires multiple repairs to be fully repaired. Due to the diverse repairs, ABC tends to be over-productive [16]. Thus, only those with the fewest faults are selected as the optimal among alternatives [10, 18] in C4. ABC repeats its repair process until there is no fault left.

As aforementioned, the KM in this paper contains a KG, which needs to be translated to Datalog first. The translation is straightforward as a triple is an assertion of a binary predicate and the TBox contains logical rules. To be succinct, we omit this format translation and directly represent the KM as a Datalog theory in this paper.

3 ABC in Root Cause Analysis

Given the assertion representing a failure as the goal[4], a cause of that failure is an axiom involved in the subset of KM which entails the goal. Thus, the root causes of a set of failures are defined as follows.

Definition 4 (Root Causes $\mathbb{R}$)**.** *Given a set of system failures* $\mathbb{E}$*, and KM of the system* $\mathbb{T}$*, the root causes of* $\mathbb{E}$ *are a minimal set of axioms* $\mathbb{R}$ *involved in the proofs of* $\mathbb{E}$*.*

$$\forall \beta \in \mathbb{E}, \mathbb{T} \setminus \mathbb{R} \nvdash \beta \bigwedge \forall \alpha \in \mathbb{R}, \exists \beta \in \mathbb{E}, \mathbb{T} \setminus \{\alpha\} \nvdash \beta \tag{5}$$

The first half of equation (5) says that a system failure β won't exist if the system does not contain root causes $\mathbb{E}$. The second half represents that any

[4] For example, a triple represents an alarm about a system failure.

axiom α which representing root causes are necessary in terms of resulting at least one system failure.

ABC starts its RCA by finding MI and then repairing the KM to include it. Figure 2 shows the damage caused by MI in RCA. All nodes are explicitly in KM except the dashed node[5]. Due to MI, the green node will not be diagnosed as the root cause of all four failures, as it should be.

Figure 3 depicts ABC's workflow in RCA, where $\mathbb{T}$ is the faulty model of the network which lacks of MI, $\mathbb{T}_1$ is correct model of the faulty network and $\mathbb{T}_2$ is how the repaired network will look.

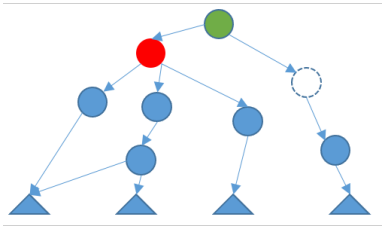

Fig. 2. Failed RCA due to MI: triangles are propositions describing system failures; circle nodes are axioms or theorems representing system behaviours; an arrow starts from a behaviour's representation to its logical consequence's; the dashed node corresponds to the axiom that should be added to represent MI, which is not in the original KM.

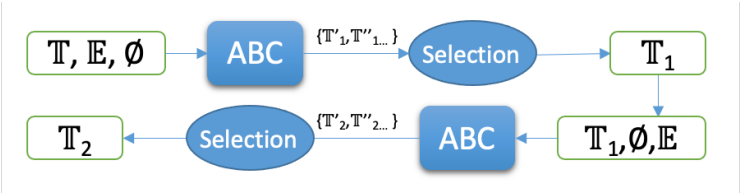

Fig. 3. RCA's flowchart: RCA's input are 1) KM $\mathbb{T}$; 2) the observed system failures as a set of assertions $\mathbb{E}$. RCA's output is the repaired KM $\mathbb{T}_2$ where the root cause is addressed. Here ABC's inputs are a KM, $\mathcal{T}(\mathbb{S})$ and $\mathcal{F}(\mathbb{S})$ in turn: in the first step $\mathcal{T}(\mathbb{S}) = \mathbb{E}$, $\mathcal{F}(\mathbb{S}) = \emptyset$; ABC outputs potential repairs $\{\mathbb{T}'_1, \mathbb{T}''_1 ...\}$, from which the selected $\mathbb{T}_1$ is the input KM of the second step, where $\mathcal{T}(\mathbb{S}) = \emptyset$, $\mathcal{F}(\mathbb{S}) = \mathbb{E}$.

Before the detailed discussion in §3.1 and §3.2, a general introduction of the workflow is given here. The main input KM, $\mathbb{T}$ written in Datalog, contains two parts: 1) knowledge extracted from system logs and user manual; and 2) rules

[5] A cause may be missing while its logical consequence exists in a KM, e.g., only the latter is recorded in the log.

representing experts' domain knowledge. Firstly, ABC repairs $\mathbb{T}$ w.r.t. the MI. Based on the enriched KM $\mathbb{T}_1$, which infers all failures, ABC repairs $\mathbb{T}_1$ and then system failures can be fixed in $\mathbb{T}_2$. Consequently, the knowledge changed in this repair constitutes the root cause. As ABC outputs multiple repaired KMs, which are correct representations for the system in different scenarios. So the 'Selection' in Figure 3 allows domain experts to choose the one that represents the target system correctly. In future, this selection can be automated by employing probability, where the selected KM should have the most significant sum of axiom probability that represents how much an axiom is trusted in describing the system accurately.

The input KG and the assertions of observed system failures are extracted from system logs and the user manual by [19] and Datalog rules representing domain knowledge are formalised by domain experts or the results of rule mining tools after being validated by experts.

3.1 Repairing Knowledge Model to Include MI

A KM containing all essential information about a system failure is the base of RCA. Thus, it is important to find MI first before analysing root causes. This section introduces why a KM has MI; how ABC detects MI and repairs KM to cover MI.

A system failure is a logical consequence of that system's setup and behaviour. Therefore, a correct representation for modelling the system ($\mathbb{M}$) should entail all of the theorems ($\mathbb{E}$) which represent the system's failures, shown by equation (6). Otherwise, that model is incorrect. The information that is well represented in a correct model but not in an incorrect model is called MI.

$$\forall \beta \in \mathbb{E},\ \mathbb{M} \implies \beta \tag{6}$$

Typical causes of MI are summarised as the following.

- KM is incomplete where new axioms need to be added.
- KM is inconsistent where old axioms need to be deleted
- KM is poorly written so that its representation needs to be adjusted, e.g., rewriting a misspelled constant.

ABC system is chosen to address MI because it has rich operations of axiom deletion/addition and representation changes. By giving assertions of observed system failures as ABC's $\mathcal{T}(\mathbb{S})$, ABC checks whether each of them is a logical consequence of the system's KM. If a failure $\alpha \in \mathcal{T}(\mathbb{S})$ is not a theorem, then ABC repairs KM to build a proof of α. These repairs represent MI about system failures so that all essential information about the observed system failures are well represented in the repaired model.

Definition 5 (Repair KM for MI)**.** *Given the input of a KM $\mathbb{T}$ to repair, all assertions of the observed system failures $\mathbb{E}$ as the true set of preferred structure ($\mathcal{T}(\mathbb{S})$) and empty set as the false set of preferred structure ($\mathcal{F}(\mathbb{S})$), ABC*

generates repaired KM ($\mathbb{T}_1$) that logically entails system failures.

$$\mathbb{T}_1 = \begin{cases} \mathbb{T} : \ \forall \alpha \in \mathbb{E}, \mathbb{T} \vdash \alpha \\ \mathbb{T}' : \mathbb{T}' \in \nu(\mathbb{T}, \mathbb{E}, \emptyset), \ \exists \alpha \in \mathbb{E}, \mathbb{T} \nvdash \alpha \end{cases} \tag{7}$$

where ν is ABC's repair function whose inputs are the knowledge model, preferred structure: $\mathcal{T}(\mathbb{S})$ and $\mathcal{F}(\mathbb{S})$ in turn.

If the current $\mathbb{T}$ has all essential information for observed system failures, then $\mathbb{T}$ entails all these failures so it does not need to be repaired, which is the first case in equation (7). Otherwise, ABC will repair $\mathbb{T}$ and generate multiple possible KMs, among which, experts can select the accurate one that describes the target system correctly. Traditionally, experts need to brainstorm all system settings or behaviours which maybe relevant to the system failures [4]. Thus, ABC's multiple solutions are not trivial: they provide directions for experts to further discover the relevant information about system failures.

However, this step examines each system failure individually, but not combining all of them to find the root cause: it is an important preparation of the RCA discussed in §3.2.

3.2 Root-Cause Discovering and Repairing

After the last step, the KM contains all essential information for system failures. This step aims to find the root cause of all failures by theory repair. The effect of repairing a cause reveals root causes: *the root cause is one whose repair leaves the fewest failures.* Based on the sub-optimal pruning mechanism [10, 18], the optimal repairs that use the minimal number of operations but solve the maximal number of faults are the ones that fix the root causes.

Definition 6 (RCA by ABC Theory Repair)**.** *Let $\mathbb{T}_1$ be a KM to repair, which entails all observed system failures: $\forall \beta \in \mathbb{E}, \mathbb{T}_1 \implies \beta$, and all assertions of observed system failures $\mathbb{E}$ as the false set of preferred structure ($\mathcal{F}(\mathbb{S})$) and empty set as the true set of preferred structure ($\mathcal{T}(\mathbb{S})$). ABC generates repaired KM $\mathbb{T}'$ that fixes system failures, where $\mathbb{T}' \in \nu(\mathbb{T}_1, \emptyset, \mathbb{E})$, from which experts can choose the best KM for the failure-free system $\mathbb{T}_2$. Then the knowledge changed in $\mathbb{T}_2$ is the root cause. Here ν is the same function defined in Equation (7)*

$$\mathbb{T}_2 = \mathbb{T}', \ \textit{where } \mathbb{T}' \in \nu(\mathbb{T}_1, \emptyset, \mathbb{E}) \bigwedge \forall \beta \in \mathbb{E}, \mathbb{T}_1 \implies \beta \tag{8}$$

We claim that given the KM $\mathbb{T}_1$ as the input theory to repair, and assertions of system failures $\mathbb{E}$ as $\mathcal{F}(\mathbb{S})$, ABC blocks all proofs of $\mathbb{E}$ with the minimal repair operations, so root causes are identified by examining the repairs that need to be made to remove them in its repaired KM $\mathbb{T}_2$. Thus, the changed parts of these repairs are the root causes.

As the KM contains some fundamental information about a system that is always correct, e.g, IP address format, ABC's protection heuristic is useful to

Example 1. *Knowledge Model $\mathbb{T}$ with MI suffering from all failures.*

$$\Longrightarrow microservice(id1, s1) \quad (1)$$
$$\Longrightarrow microservice(id2, s1) \quad (2)$$
$$\Longrightarrow microservice(id3, s2) \quad (3)$$
$$\Longrightarrow full(d1) \quad (4)$$
$$\Longrightarrow createOn(id2, d1) \quad (5)$$
$$\Longrightarrow sameRoute(id2, id3) \quad (6)$$
$$full(X) \wedge createOn(Y, X) \Longrightarrow fail(Y) \quad (7)$$
$$ms(X, s1) \wedge ms(Y, s2) \wedge sameRoute(X, Y) \Longrightarrow depend(Y, X) \quad (8)$$
$$depend(X, Y) \wedge fail(Y) \Longrightarrow fail(X) \quad (9)$$

$$\mathcal{T}(\mathbb{S}) = \{fail(id1), fail(id2), fail(id3)\}$$
$$\mathcal{F}(\mathbb{S}) = \emptyset$$

protect them from being repaired. This protection avoids incorrect repairs so prunes fake root causes.

As aforementioned, ABC is guaranteed to find fault proofs when there are any. Once a theory is detected as faulty, then ABC will recursively try to repair its faults. If this repair process terminates with success, we can guarantee that a root cause has been found. But there is no guarantee that this repair process will always terminate successfully. It could introduce new faults at least as fast as it removes them theoretically: no such a case occurs so far.

4 Evaluation

In this section, our solution is validated in the context of Huawei's 5G network where the failure of one microservice could have a waterfall effect. As our method of repairing system model when tackling RCA tasks is unique, there will be no comparison but a detailed example to illustrate our method in this evaluation, especially in terms of the model repairing. Finally, we will discuss how ABC finds and repairs the root cause. The comparison with other RCA systems that do not addressing MI is a future work, which will be given as a part of the evaluation of the TREAT project [22].

Example 1 describes the KM, $\mathbb{T}$, written in Datalog by following the convention given by Equation (1): three microservice instances' IDs are $id1$, $id2$ and $id3$, respectively, and the first two are of type $s1$ and the last $s2$, represented by axiom (1-3). Axioms (4-5) say that $id2$ is created on $d1$ and that the device

$d1$ is already full. Rule (7) tells that a microservice fails when it is created on a full device, $fail(id2)$ is a theorem of this $\mathbb{T}$. Note that there is an inconsistent representation between microservice in (1-3) and its acronym 'ms' in (8), which makes the reasoner fail in unifying $microservice(X, Y)$ and $ms(X, Y)$. Otherwise, based on (2, 3, 6, 8), we would know that $id2$ and $id3$ were for the same route and $depend(id3, id2)$. By combining (9), $fail(id3)$ would be concluded. Thus, among all system failures of $fail(id1)$, $fail(id2)$ and $fail(id3)$ given by $\mathcal{T}(\mathbb{S})$, only $fail(id2)$ can be predicted by $\mathbb{T}$.

Assume that the full device $d1$ causes two newly created microservice instances $id1$ and $id2$ to fail. In addition, another microservice $id3$ also fails because it depends on the failed $id2$. However, the incomplete log only contains the information about creating instance $id2$ and misses creating $id1$, e.g., it is deleted due to the log's limit being reached. Thus, RCA needs to discover the MI about $id1$'s creation and then diagnose that the full device is the root cause of these three failures. In addition, the inconsistent use of $microservice$ and ms need to be corrected as well.

Example 2. *Enriched KM $\mathbb{T}_1$ suffering from all failures.*

$$\begin{aligned}
&\Longrightarrow microservice(id1, s1) && (1)\\
&\Longrightarrow microservice(id2, s1) && (2)\\
&\Longrightarrow microservice(id3, s2) && (3)\\
&\Longrightarrow full(d1) && (4)\\
&\Longrightarrow createOn(id2, d1) && (5)\\
&\Longrightarrow createOn(id1, d1) && (5^*)\\
&\Longrightarrow sameRoute(id2, id3) && (6)\\
full(X) \wedge createOn(Y, X) &\Longrightarrow fail(Y) && (7)\\
microservice(X, s1) \wedge microservice(Y, s2) \wedge & \\
sameRoute(X, Y) &\Longrightarrow depend(Y, X) && (8')\\
depend(X, Y) \wedge fail(Y) &\Longrightarrow fail(X) && (9)
\end{aligned}$$

$$\mathcal{T}(\mathbb{S}) = \emptyset$$
$$\mathcal{F}(\mathbb{S}) = \{fail(id1), fail(id2), fail(id3)\}$$

In this first step of RCA,$\mathbb{T}$ needs to be repaired so that it can predict not only $fail(id2)$, but also $fail(id1)$ and $fail(id3)$. Among all ABC's repairs that build a proof for $fail(id1)$, adding $createOn(id1, d1)$ is the correct cause of $fail(id1)$ in this scenario, which is generated by Abduction 1 in Definition 3. Meanwhile,

by renaming *ms* into *microservice* in axiom (8), the inconsistency is repaired, which is generated by Reformation 6 in Definition 3. Then $\mathbb{T}_1$ in Example 2 is selected as the enriched model for the next step of RCA, where changes are highlighted in red.

In the second step of RCA, $\mathbb{T}_1$ is the input and failures are given as $\mathcal{F}(\mathbb{S})$. Then ABC generates repairs that block all proofs of three failures, among which deleting axiom (4) and rewriting (4) as a new axiom $\Longrightarrow dummy\_full(d1)$ fix the root cause of $full(d1)$. These two solutions are shown in Example 3 and 4 whose repair operations are from Belief Revision 1 and in Reformation 3 in Definition 2, respectively. The repaired axiom with the new predicate *dummy_full* represents that the root cause is board $d1$ being full, and the operations that can change $d1$'s status from full to not full can address these failures.

Example 3. *Repaired KM $\mathbb{T}_2$ derives no failures.*

$$\Longrightarrow microservice(id1, s1) \quad (1)$$
$$\Longrightarrow microservice(id2, s1) \quad (2)$$
$$\Longrightarrow microservice(id3, s2) \quad (3)$$
$$\Longrightarrow dummy\_full(d1) \quad (4)$$
$$\Longrightarrow createOn(id2, d1) \quad (5)$$
$$\Longrightarrow createOn(id1, d1) \quad (5^*)$$
$$\Longrightarrow sameRoute(id2, id3) \quad (6)$$
$$full(X) \wedge createOn(Y, X) \Longrightarrow fail(Y) \quad (7)$$
$$microservice(X, s1) \wedge microservice(Y, s2) \wedge sameRoute(X, Y) \Longrightarrow depend(Y, X) \quad (8')$$
$$depend(X, Y) \wedge fail(Y) \Longrightarrow fail(X) \quad (9)$$

$$\mathcal{T}(\mathbb{S}) = \emptyset$$
$$\mathcal{F}(\mathbb{S}) = \{fail(id1), fail(id2), fail(id3)\}$$

This example shows how the ABC repair system detects root-causes when there is missing information: it extend the KM with MI about a failure and then diagnoses and repairs the observed failures' root cause. To our best knowledge, other RCA methods do not deal with MI so cannot find correct root causes when there is MI.

In addition, discovering the missing information about *createOn* is guidance for experts to optimise system logs: records about *createOn* are important in RCA so they should be protected to avoid being deleted in future. On the other hand, it spots and solves the representation inconsistency of using both *microservice* and *ms* by replacing all occurrences of the latter with the former.

Example 4. *Repaired KM $\mathbb{T}_3$ by Belief Revision derives no failures.*

$$\Longrightarrow microservice(id1, s1) \quad (1)$$
$$\Longrightarrow microservice(id2, s1) \quad (2)$$
$$\Longrightarrow microservice(id3, s2) \quad (3)$$
$$\Longrightarrow createOn(id2, d1) \quad (5)$$
$$\Longrightarrow createOn(id1, d1) \quad (5^*)$$
$$\Longrightarrow sameRoute(id2, id3) \quad (6)$$
$$full(X) \wedge createOn(Y, X) \Longrightarrow fail(Y) \quad (7)$$
$$microservice(X, s1) \wedge microservice(Y, s2) \wedge sameRoute(X, Y) \Longrightarrow depend(Y, X) \quad (8)$$
$$depend(X, Y) \wedge fail(Y) \Longrightarrow fail(X) \quad (9)$$

$$\mathcal{T}(\mathbb{S}) = \emptyset$$
$$\mathcal{F}(\mathbb{S}) = \{fail(id1), fail(id2), fail(id3)\}$$

5 Conclusion

This paper introduces a novel RCA mechanism for system failures by applying ABC to the system's KM and system failures: firstly, ABC automatically discovers possible MI when failures cannot be deduced from the given KM. After adding the interactively selected MI to KM, ABC automatically repairs the enriched KM to fix root causes so that failures are not logical consequences.

Automatically discovering possible MI makes ABC less restricted by the completeness of the KM than other approaches. It also guides experts to enrich the KM by extending the system logs w.r.t. MI as to continuously improve the performance of the RCA pipeline.

Limitations reveal the future work: 1) ABC's scalability limit for massive KM. Possible solutions include optimising its computation flow and minimising its input by pruning axioms that are irrelevant to the failures, e.g. clustering logs [11]. 2) Experts are needed to select the best answer from ABC's output currently. To minimise manual work, ABC's output can be ranked by incorporating probability in future; 3) a more sophisticated evaluation is crucial, ideally, with open-source data[2, 7].

ACKNOWLEDGMENTS

The authors would like to thank Huawei for supporting the research and providing data on which this paper was based under grant CIENG4721/LSC. Also we

gratefully acknowledge UKRI grant EP/V026607/1 and the support of ELIAI (The Edinburgh Laboratory for Integrated Artificial Intelligence) EPSRC (grant no EP/W002876/1). Thanks are also due to Zhenhao Zhou for the valuable discussions around network software systems. In addition, anonymous reviewers also gave us very useful feedback that improved the quality of this paper.

References

1. Ceri, S., Gottlob, G., Tanca, L.: Logic Programming and Databases. Surveys in Computer Science, Springer-Verlag, Berlin (1990)
2. Chapman, A., Simperl, E., Koesten, L., Konstantinidis, G., Ibáñez, L.D., Kacprzak, E., Groth, P.: Dataset search: a survey. The VLDB Journal **29**(1), 251–272 (2020)
3. Cherrared, S., Imadali, S., Fabre, E., Gössler, G.: Sfc self-modeling and active diagnosis. IEEE Transactions on Network and Service Management (2021)
4. Dalal, S., Chhillar, R.S.: Empirical study of root cause analysis of software failure. ACM SIGSOFT Software Engineering Notes **38**(4), 1–7 (2013)
5. Gallier, J.: SLD-Resolution and Logic Programming. Chapter 9 of Logic for Computer Science: Foundations of Automatic Theorem Proving (2003), originally published by Wiley, 1986.
6. He, P., Zhu, J., He, S., Li, J., Lyu, M.R.: An evaluation study on log parsing and its use in log mining. In: 2016 46th annual IEEE/IFIP international conference on dependable systems and networks (DSN). pp. 654–661. IEEE (2016)
7. He, S., Zhu, J., He, P., Lyu, M.R.: Loghub: A large collection of system log datasets towards automated log analytics. arXiv preprint arXiv:2008.06448 (2020)
8. Jia, T., Chen, P., Yang, L., Li, Y., Meng, F., Xu, J.: An approach for anomaly diagnosis based on hybrid graph model with logs for distributed services. In: 2017 IEEE International Conference on Web Services (ICWS). pp. 25–32. IEEE (2017)
9. Kowalski, R.A., Kuehner, D.: Linear resolution with selection function. Artificial Intelligence **2**, 227–60 (1971)
10. Li, X.: Automating the Repair of Faulty Logical Theories. Ph.D. thesis, School of Informatics, University of Edinburgh (2021)
11. Lin, Q., Zhang, H., Lou, J.G., Zhang, Y., Chen, X.: Log clustering based problem identification for online service systems. In: 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C). pp. 102–111. IEEE (2016)
12. Lu, J., Dousson, C., Krief, F.: A self-diagnosis algorithm based on causal graphs. In: The Seventh International Conference on Autonomic and Autonomous Systems, ICAS. vol. 2011 (2011)
13. Pfenning, F.: Datalog. Lecture 26, 15-819K: Logic Programming (2006), `https://www.cs.cmu.edu/~fp/courses/lp/lectures/26-datalog.pdf`
14. Qiu, J., Du, Q., Yin, K., Zhang, S.L., Qian, C.: A causality mining and knowledge graph based method of root cause diagnosis for performance anomaly in cloud applications. Applied Sciences **10**(6), 2166 (2020)
15. Shima, K.: Length matters: Clustering system log messages using length of words. arXiv preprint arXiv:1611.03213 (2016)
16. Smaill, A., Li, X., Bundy, A.: ABC repair system for Datalog-like theories. In: KEOD. pp. 333–340 (2018)
17. Solé, M., Muntés-Mulero, V., Rana, A.I., Estrada, G.: Survey on models and techniques for root-cause analysis. arXiv preprint arXiv:1701.08546 (2017)

18. Urbonas, M., Bundy, A., Casanova, J., Li, X.: The use of max-sat for optimal choice of automated theory repairs. In: Bramer, M., Ellis, R. (eds.) Artificial Intelligence XXXVII. pp. 49–63. Springer International Publishing, Cham (2020)
19. Wang, F., Alan Bundy, X.L., Zhu, R., Nuamah, K., Xu, L., Mauceri, S., Pan, J.Z.: Lekg: A system for constructing knowledge graphs from log extraction. The 10th International Joint Conference on Knowledge Graphs (2021)
20. Zawawy, H., Kontogiannis, K., Mylopoulos, J.: Log filtering and interpretation for root cause analysis. In: 2010 IEEE International Conference on Software Maintenance. pp. 1–5. IEEE (2010)
21. Zhou, Q., Gray, A.J., McLaughlin, S.: Seanet–towards a knowledge graph based autonomic management of software defined networks. arXiv preprint arXiv:2106.13367 (2021)
22. Zhu, R., Li, X., Wang, F., Bundy, A., Pan, J.Z., Nuamah, K., Xu, L., Mauceri, S.: TREAT: Automated construction and maintenance of probabilistic knowledge bases from logs (extended abstract). The 8th Annual Conference on machine Learning, Optimization and Data science (LOD) (2022)