Diagnosis of Runtime Property Violations

Marco Bozzano[1], Alessandro Cimatti[1], Alberto Sambrotta[1,2], and Stefano Tonetta[1]

[1] Fondazione Bruno Kessler, Trento, Italy
{bozzano,cimatti,asambrotta,tonettas}@fbk.eu
[2] Università degli Studi di Udine, Udine, Italy
sambrotta.alberto@spes.uniud.it

Abstract. Runtime verification checks, during execution, whether a system satisfies a given property. When a violation is detected, identifying the underlying causes is essential for fault recovery. However, existing approaches to model-based diagnosis do not fully address this problem. Model-Based Diagnosis (MBD) explains deviations between observed and nominal behaviors, but does not target property violations specifically. Model-Based Safety Analysis (MBSA), on the other hand, explains possible causes of a top-level event, but does not exploit runtime observations. In this work, we address the Property Violation Diagnosis (PVD) problem, which lies at the intersection of runtime verification and model-based diagnosis: given a system description, a monitored property, and a set of observations that indicate its violation, we aim to compute the set of faults that are relevant to explain the violation. We provide a formal characterization of PVD in propositional logic and propose symbolic algorithms for generating the relevant diagnoses. Our experimental evaluation demonstrates the applicability of the proposed approach.

Keywords: Property Violation · Model-Based Diagnosis · Model-Based Safety Analysis · Fault Diagnosis

1 Introduction

Safety-critical systems require the continuous monitoring of specific properties to ensure correct operation [7]. When a violation is detected, a recovery mechanism must be triggered [6, 12, 16]. However, an effective recovery requires identifying precisely which faults caused the property violation. A classic example of such requirements can be found in the aerospace domain [1, 22, 24], where Fault Detection, Identification, and Recovery (FDIR) software is heavily utilized to continuously monitor the system state and trigger appropriate recovery mechanisms upon fault detection. Furthermore, depending on the specific application and the criticality of the fault, the time available to initiate recovery may be subject to strict real-time constraints.

Standard Model-Based Diagnosis (MBD) [9] identifies fault sets that render the system model consistent with observations. However, MBD explains general deviations from nominal behavior rather than specific property violations. This is

insufficient for our context, as we must isolate the faults specifically responsible for the violation to trigger the right recovery. The core issue with applying standard MBD here is the inclusion of "irrelevant" faults. From observations - i.e., information about the state of the system - we may infer the occurrence of a fault that is completely unrelated to the violation of the monitored property.

We term this specific challenge Property Violation Diagnosis (PVD). PVD lies at the intersection of runtime verification [2] and model-based diagnosis: given a system description, a monitored property, and a set of observations indicating a violation, we aim to compute the set of faults that are strictly relevant to explaining that violation.

In this paper, we first provide a formal characterization of PVD in propositional logic. To ensure this definition captures the intuitive notion of "blame" in partial observability, we validate our formalization against a set of motivating examples. These scenarios are designed to illustrate cases where standard diagnosis fails to distinguish between faults that caused the violation and those that were merely coincidentally present. In the related work, we also discuss why other similar approaches such as model-based safety analysis, causal analysis and abductive reasoning are not suitable to solve the PVD problem.

We then translate this formalization into a quantified Boolean formula and propose a solution based on propositional quantifier elimination. Our algorithm leverages symbolic methods to compute the set of relevant diagnoses.

Note that, although our presentation focuses on propositional logic and uses BDD-based quantifier elimination [18], the definitions and algorithms extend naturally to richer theories in first-order logic, and can be instantiated using SMT solvers with only modest changes to the underlying reasoning procedures.

The main contributions of this paper are the following:

- We formally define the PVD problem and we validate this definition through varied scenarios involving partial observability and irrelevant faults.
- We provide a symbolic algorithm based on propositional quantifier elimination to compute relevant diagnoses.
- We provide an experimental evaluation demonstrating the applicability of the proposed approach, suitable for an online application.

The rest of the paper is organized as follows. Section 2 reviews the background on MBD. Section 3 presents the motivating examples used to validate our definition. Section 4 discusses related work. Section 5 details the formalization of the PVD problem. Section 6 presents the symbolic approach to solve PVD and Section 7 presents the experimental evaluation. Finally, Section 8 concludes the paper.

2 Background

We assume standard background in propositional logic (formulae, models, satisfiability, entailment), which forms the basis of our propositional formalization. Only the notions required for diagnosis are summarized below [17].

Definition 1 (set of cutsets). *Let β be a formula in propositional logic, W a set of models and X a set of variables. The set of cutsets for β in the set of models W, over the variables in X is defined as*

$$CS(X, W, \beta) = \{cs \subseteq X \mid \exists w \in W.\ w \models \beta,\ \forall x \in cs.\ w \models x,\ \forall x \notin cs.\ w \models \neg x\} \tag{1}$$

Definition 2 (minimal/maximal cutsets). *Let S be a set of cutsets. The set of minimal/maximal cutsets in S is defined as*

$$MinCS(S) = \{cs \in S \mid \nexists cs' \in S.\ cs' \subset cs\} \tag{2}$$

$$MaxCS(S) = \{cs \in S \mid \nexists cs' \in S.\ cs \subset cs'\} \tag{3}$$

In the following sections it will be necessary to represent cutsets as propositional formulae. Given a cutset cs, its formula representation is defined as $\varphi(cs) = \bigwedge_{f \in cs} f \wedge \bigwedge_{f \notin cs} \neg f$.

Model-Based Diagnosis (MBD) MBD is defined for first-order logic, but it can be applied to a variety of logics; in this paper we focus on the propositional logic. Let $COMPS = \{c_1, ..., c_n\}$ be the *set of components*, represented by constants. For a component c, its *nominal behaviour* is defined by the propositional formula $B(c)$, and variable f_c determines whether c is faulted or nominal. The *system description* is defined by the formula $SD := \bigwedge_{c \in COMPS} \neg f_c \rightarrow B(c)$. An observation OBS is an assignment to the variables of SD representing the input/outputs of the components. The system is not behaving nominally iff $SD \wedge OBS \wedge \bigwedge_{c \in COMPS} \neg f_c$ is unsatisfiable. A *diagnosis* for $\langle SD, COMPS, OBS \rangle$ is a minimal set $\Delta \subseteq COMPS$ such that $SD \wedge OBS \wedge \bigwedge_{c \in \Delta} f_c \wedge \bigwedge_{c \in COMPS \setminus \Delta} \neg f_c$ is satisfiable.

We adapt MBD to our notation as follows: $COMPS$ is the set of state Boolean variables V, SD is a propositional formula defined over V and fault variables F, and OBS is a truth assignment over a set of observable variables O. The system is not behaving nominally iff $SD \wedge OBS \wedge \bigwedge_{f \in F} \neg f \models \bot$, and the set of diagnoses for $\langle SD, COMPS, OBS \rangle$ is defined as $MinCS(CS(F, SD \wedge OBS, \top))$.

3 Motivating Examples

Consider a system with two rooms, A and B. In room A, there is a permanent emergency light and three lights controlled by a single switch. Room B has one light that turns on if the switch is on and motion is detected. The breakage of the switch turns off all connected lights permanently, while a broken light only affects itself. Additionally, Room B's light has a sensor to detect if it's malfunctioning. And last, we say that room A is lit if at least one of the lights in it is on. The

described system can be formalized as the following propositional formula:

$$
\begin{aligned}
SD := \ & \text{power\_on} \leftrightarrow (\text{switch\_on} \wedge \neg\text{switch\_faulted}) \wedge \\
& \text{emergency\_light\_on} \leftrightarrow \neg\text{emergency\_light\_faulted} \wedge \\
& (\wedge_{i=1}^{3} \text{light}_{Ai}\text{\_on} \leftrightarrow (\text{power\_on} \wedge \neg\text{light}_{Ai}\text{\_faulted})) \wedge \\
& \text{light}_{B1}\text{\_on} \leftrightarrow (\text{power\_on} \wedge \neg\text{light}_{B1}\text{\_faulted} \wedge \text{motion\_detected}) \wedge \\
& \text{roomA\_dark} \leftrightarrow ((\wedge_{i=1}^{3} \neg\text{light}_{Ai}\text{\_on}) \wedge \neg\text{emergency\_light\_on}) \wedge \\
& \text{observe\_light}_{B1}\text{\_broken} \leftrightarrow \text{light}_{B1}\text{\_faulted}
\end{aligned}
$$

Suppose that we want to monitor three different properties and that for each one we are given different information about the system:

- "$PROP_1 :=$ switch_on $\rightarrow$ (light$_{A1}$_on $\wedge$ light$_{A2}$_on)", we want to monitor the nominal behaviour of the first and second light in room A, meaning that if the switch is on then both lights must be on:
  - "$OBS_1 :=$ $\neg$light$_{A1}$_on$\wedge$light$_{A2}$_on$\wedge$light$_{A3}$_on". Since we can observe that only the first light is off, we can say with certainty that it is broken;
  - "$OBS_2 :=$ $\neg$light$_{A1}$_on $\wedge$ $\neg$light$_{A2}$_on $\wedge$ light$_{A3}$_on". The observation shows that only the third one is on, while the first and second light are off, meaning that they must be broken, but since the property is violated with either one of them broken they are both relevant faults;
  - "$OBS_3 :=$ $\neg$light$_{A1}$_on$\wedge$light$_{A2}$_on$\wedge\neg$light$_{A3}$_on". We know that only the second light is on, while the first and third lights are off, meaning that they are broken, but only the breakage of the first light is relevant, since even if the third light was on (thus not broken), the violation would have still occurred;
  - "$OBS_4 :=$ switch_on $\wedge$ $\neg$light$_{A1}$_on $\wedge$ $\neg$light$_{A2}$_on $\wedge$ $\neg$light$_{A3}$_on". We can observe that the switch is on but all of the lights in room A are off. The breakage of the switch can be a cause, since we can't observe any light on, and the breakage of the third light is not relevant, as in previous case, while the breakage of either the first or the second light can be considered as relevant faults.
- "$PROP_2 :=$ emergency_light_on", we want to monitor the correct behaviour of the emergency light, stating that it must stay on no matter what:
  - "$OBS_1 :=$ switch_on $\wedge$ roomA_dark." Here the switch is on but there is no light in the room. If the system was nominal then there would be light in the room, but since it is not illuminated we can conclude all of the lights are off. But since the property states that just the emergency light must be on then its breakage must be relevant to the violation.[1]
- "$PROP_3 :=$ switch_on $\rightarrow$ (light$_{A1}$_on$\wedge$(motion_detected $\rightarrow$ light$_{B1}$_on))", we want to monitor the nominal behaviour of the first light in room A and the light in room B, stating that if the switch is on then the first light of room A must be on, and if there is motion in room B, then also the light in room B must be on:

[1] Note that if we observed $\neg$roomA_dark then we wouldn't have been able to deduce the violation, since we don't know if the emergency light is on.

- "$OBS :=$ switch_on$\wedge\neg$light$_{A1}$_on$\wedge\neg$light$_{B1}$_on$\wedge\neg$motion_detected$\wedge$ observe_light$_{B1}$_broken". The switch is on, both of the lights are off, the light in room B is broken, and finally there is no motion in room B. Even if there are scenarios in which the breakage of the light in room B can violate the property, in this one it didn't contribute, since the sensor didn't detect any movement. Thus, the faults relevant to the violation are either the breakage of the first light or the switch.

4 Related Work

In the following, we compare our work with Halpern and Pearl definition of Actual Causality, MBD, and MBSA (which are the most relevant work w.r.t. our approach), and then we discuss other works of the state-of-the-art and compare them with our work.

Our notion of relevance is broadly related with causality. Halpern and Pearl, see [14], have defined (several variants of) the notion of (actual) causality based on structural equations. Causes are taken from a generic set of endogenous variables, whereas we defined explanations as sets of faults. We inherit from them the concept of counterfactual reasoning. However, in all cases our notion of causality is different: in the original definition, causes may be restricted to single conjuncts, whereas we consider sets; in the modified definition [14] conjunctive and disjunctive scenarios yield different results w.r.t. our definition. Our work is rooted in the setting of Model-Based Diagnosis [9] and extends it to compute relevant explanations for the violation of a given property. Starting from the observations, we compute minimal sets of causes that do not contain faults that are irrelevant (i.e, they do not contribute to the violation). To the best of our knowledge, this is the first work that addresses the notion of relevance for diagnosing property violations, for partially observable systems. Model-Based Safety Analysis (MBSA) [4, 5, 15] looks for explanations in terms of faults, however the analysis is done at design time rather than at runtime, without considering observability.

In [13] the authors define causality for Discrete Event Systems, where causes are subsequences of events consistent with the system and the observations. Their notion is similar to ours, however minimization is carried out on a larger set of events, other than faults, which, as the authors admit, may result in explanations including irrelevant events, e.g., in case of system initialization. In [11] the authors look for temporal explanations of property violation on a given execution trace (no system description is given). The framework is very expressive, it covers both MTL and LTL, and causality is rooted in the notion of *prime implicant*, which is a generalization of *cut set*, however they do not consider the notion of irrelevance. [3] aims to pinpoint the possible explanations for an LTL property on a given execution trace (no system description is given). Causality is defined using counterfactual reasoning and a notion of contingency inspired from [14]. However, causes may only contain signals occuring in the property, whereas we consider faults, that are generally unobservable and do not

appear in properties. Moreover, they consider individual causes only, whereas we consider sets. [25] aims to compute instant causes of the violation of a property expressed in Signal Temporal Logic (STL) in the context of hybrid systems (no system description is given). The authors quantify the relevance of each instant, however their notion of causality is local to each time instant, and is tailored to signal evaluation in hybrid systems.

Finally, there is a close relationships between Model-Based Diagnosis and abduction [10, 20]. Our theory of diagnosis could be equivalently formulated in the realm of abduction. As in the case of diagnosis, the theory of abduction does not address the notion of irrelevance that is central in our work.

5 Problem Formalization

In this section, we formalize the PVD problem.

Definition 3 (Monitor Setting). *The monitor setting* $MS = \langle V, O, F, SD, PROP, OBS \rangle$ *is a tuple where:* V *is the set of state variables,* $O \subseteq V$ *is the set of observable variables,* F *is the set of fault variables,* $SD(V, F)$ *a formula representing the system description,* $PROP(V)$ *a formula representing the property to be monitored, and* $OBS(O)$ *a conjunction of assignments over each variable* $o \in O$ *representing an observation on the system.*

We will often refer to assignment to fault variables as *fault configuration*, and $\neg PROP$ as Top-Level Event (TLE). Throughout the rest of the paper, the term *occurrence of faults* is used to denote a set of fault variables assigned to $\top$. We now present three desired properties of a monitor setting that are assumed to be true for the rest of the paper and then, at the end of the section, show how the results are affected if these assumptions don't hold.

Assumption 1 (Nominal behaviour) *Let* SD*,* $PROP$*, and* F *be as in Definition 3. We assume that the following holds:*

$$SD \models \left(\bigwedge_{f \in F} \neg f \right) \rightarrow PROP \tag{4}$$

Informally, if the system is behaving nominally, then the property is satisfied, implying that if $(SD \wedge OBS) \models TLE$ holds, then the system is not behaving nominally, meaning that it can be diagnosed. [2]

Assumption 2 (TLE event monotonicity w.r.t. faults) *Let* SD, TLE*, and* F *be as in Definition 3. We assume that the following holds:*

$$SD \models \forall \Delta \in CS(F, SD, TLE).\ \forall \Delta^{+}.\ (\Delta \subseteq \Delta^{+} \rightarrow \Delta^{+} \in CS(F, SD, TLE)) \tag{5}$$

[2] If $(SD \wedge OBS) \models PROP$ holds, then the system is nominal. If $(SD \wedge OBS) \not\models PROP$ and $(SD \wedge OBS) \not\models \neg PROP$ both hold, then we don't know if the property is satisfied or not.

Informally, if the TLE is satisfied with the fault configuration Δ, then there are no possible additions to Δ such that they restore the property.

Definition 4 (Irrelevant subsets of faults). *Let MS be a monitor setting such that $(SD \wedge OBS) \models TLE$ is satisfied, and let $\Delta \in CS(F, SD \wedge OBS, \top)$. A subset $\Delta' \subseteq \Delta$ is irrelevant iff by taking $\Delta^- = \Delta \setminus \Delta'$ there exists a conjunction OBS' of a subset of literals in OBS such that:*

$$(SD \wedge OBS' \wedge \varphi(\Delta^-)) \not\models \bot \text{ and } (SD \wedge OBS' \wedge \varphi(\Delta^-)) \models TLE \qquad (6)$$

The intuition is to use the counterfactual approach to filter out faults in a cutset consistent with the observation that did not contribute to the TLE. By assuming Assumption 2, only the occurrence of faults can be considered as a diagnosis. Given $\Delta \in CS(F, SD \wedge OBS, \top)$ we can "flip" the faults of a subset Δ' in Δ and check whether the TLE still occurs or not in the context given by the observation. It could be possible that modified cutset $\Delta^- = \Delta \setminus \Delta'$, makes the formula $SD \wedge OBS \wedge \Delta^-$ unsatisfiable. In order to avoid this problem, we search for a subset OBS' of OBS such that $SD \wedge OBS' \wedge \Delta^-$ is satisfiable. The idea is to find a "similar" situation to the one given by the observation in order to consider a different fault configuration.

Definition 5 (Relevant Diagnosis). *Let MS be a monitor setting such that $(SD \wedge OBS) \models TLE$ is satisfied. Δ_r is a relevant diagnosis for the TLE if there exists $\Delta \in CS(F, SD \wedge OBS, \top)$ such that $\Delta_r \subseteq \Delta$ and it does not contain subsets of irrelevant faults.*

Note that *"it does not contain subsets of irrelevant faults"* is a minimality condition, meaning that the aim is to remove the largest subsets of irrelevant faults.

Finally, we can define the PVD problem as follows

Definition 6 (PVD Problem). *Let MS be a monitor setting such that $(SD \wedge OBS) \models TLE$ is satisfied. The PVD problem aims to find all relevant diagnosis for the TLE.*

Clearly, MBD doesn't solve the PVD problem, since it is independent from the property. However, it is useful in our approach since a diagnosis consistent with the observation can be used as a starting point.

Remark on the properties assumed on the system If Assumption 1 doesn't hold, then $PROP$ can be violated when everything is nominal, i.e., the empty cutset is a potential cause of the TLE, and by minimality, it would be the only diagnosis returned. If Assumption 2 doesn't hold, then situations in which the non-occurrence of a fault is a potential cause of the TLE are allowed, but Definition 5 focuses only on occurrence of faults, i.e., it ignores the non-occurrence of faults.

6 Algorithms

In the following we propose two BDD-based algorithms: one that enumerates the cut sets of MBD and prunes irrelevant faults and one that is fully symbolic based just on BDD operations. Both algorithms are rooted in knowledge compilation concepts [8], meaning that we pre-compute a parameterized BDD to query at runtime substituting the parameters with the actual values.

Parameters Let SD, TLE, V, O, F be defined as in Definition 3, Obs be an assignment to the variables in O representing OBS (i.e., $OBS = \bigwedge_{o \in O} o \leftrightarrow Obs(o)$), $\Delta \in CS(F, SD \wedge OBS, \top)$, A_O and PH_O be two sets of Boolean variables, where $a_o \in A_O, ph_o \in PH_O$ are renaming of the variable o for each $o \in O$, and a set of Boolean variables PH_F, where $ph_f \in PH_F$ is a renaming of the variable f for each $f \in F$.

We refer to variables A_O as *activation variables* and we use them to represent the subset OBS' of Definition 4. The variables PH_O and PH_F act as symbolic parameters that allow us to compile the BDDs offline without knowing the actual runtime observations (Obs) or the specific fault configuration (Δ) ahead of time; at runtime, these placeholders are simply replaced with concrete assignments. Specifically, PH_O is required for both proposed approaches, whereas PH_F is used only by the enumerative algorithm.

6.1 Enumerative Algorithm

Enumerative relevant diagnoses encoding The following formula encodes Definition 5:

$$\begin{aligned} \text{relevant}(F, PH_O, PH_F) = &\exists V, A_O.\ (\text{lhsE}(V, F, A_O, PH_O, PH_F) \wedge \\ &(\forall V'.\ \text{lhsE}(V', F, A_O, PH_O, PH_F) \rightarrow TLE)) \end{aligned} \tag{7}$$

where:

- $\text{lhsE}(V, F, A_O, PH_O, PH_F) = SD(V, F) \ \wedge\ \text{act\_obs}(A_O, O, PH_O) \ \wedge\ \text{sub\_of}(F, PH_F)$, represents the formula $(SD \wedge OBS' \wedge \varphi(\Delta^-))$;
- $\text{act\_obs}(A_O, O, PH_O) = \bigwedge_{o \in O} a_o \rightarrow (o \leftrightarrow ph_o)$, stating that for each $o \in O$, activation variable a_o controls the variables O that take the same value as defined by ph_o (i.e., they belong to OBS');
- $\text{sub\_of}(F, PH_F) = \bigwedge_{f \in F} f \rightarrow ph_f$, stating that the assignment to the variables in F must be a subset of the assignment to the variables in PH_F.

The minimality condition is encoded into Formula (7) by adding the constraint $\nexists V', F', A'_O.(\text{lhsE}(V', F', A'_O, PH_O, PH_F) \wedge (\forall V''.\ \text{lhsE}(V'', F', A'_O, PH_O, PH_F) \rightarrow TLE) \wedge \text{sub\_of}(F', F) \wedge \bigvee_{f \in F} \neg(f \leftrightarrow f'))$.

Let ϕ_E denote the BDD of Formula (7). Notice that ϕ_E is independent from Obs and Δ, meaning that it can be computed statically for any given set O and propositional formula $PROP$. In the following algorithm we assume that ϕ_E has been pre-computed.

Algorithm 1 Relevant diagnoses

1: **function** COMPUTE_RELEVANT_DIAGNOSES(V, O, F, SD, Obs, ϕ_E)
2: $\mathcal{D} = \emptyset$
3: $CS = \exists V.\ SD \wedge \bigwedge_{o \in O} o \leftrightarrow Obs(o)$
4: **for** $\Delta \in CS$ **do**
5: $\phi'_E = \phi_E[ph_{o_1}/Obs(o_1), ..., ph_{o_m}/Obs(o_m), ..., ph_{f_1}/\Delta_{f_1}, ..., ph_{f_n}/\Delta_{f_n}]$
6: $\mathcal{D} = \mathcal{D} \cup \Delta_r$ **for** Δ_r **in** enumerate_models(ϕ'_E)
7: **end for**
8: return $\mathcal{D}$
9: **end function**

Algorithm 1 computes every cutset $\Delta \in CS(F, SD \wedge OBS, \top)$ (line 3). Then, for each Δ, it computes ϕ' by substituting ph_o with $Obs(o)$ for each $o \in O$, and ph_f with Δ_f for each $f \in F$ (line 5). Then it calls the enumerate_models function which takes a BDD as a parameter and it returns all of its satisfying assignments. Finally, it returns the set of relevant diagnoses (line 8). We can say that $\Delta_r \subseteq F$ is a relevant diagnosis iff $\varphi(\Delta_r) \models \phi'_E$, meaning that Δ_r is a model of ϕ'_E.

Algorithm 1 computes the set $CS(F, SD \wedge OBS, \top)$, which can be exponential in size. To optimize this enumeration, we note the following property.[3]

Lemma 1. *If Δ_r is a relevant diagnosis in Δ, then Δ_r is a relevant diagnosis for each Δ^+ such that $\Delta \subseteq \Delta^+$.*

Lemma 2. *The set obtained by the union of the relevant diagnoses of each $cs \in CS(F, SD \wedge OBS, \top)$ is equal to the set obtained by the union of the relevant diagnoses of each $mcs \in MaxCS(CS(F, SD \wedge OBS, \top))$.*

Lemma 2 allows to check only cutsets in $MaxCS(CS(F, SD \wedge OBS, \top))$.

6.2 Fully Symbolic Approach

Symbolic relevant diagnoses encoding The following formula encodes Definition 5:

$$\exists V, F'.\ (SD(V, F') \wedge \bigwedge_{o \in O} o \leftrightarrow ph_o \ \wedge\ \text{sub\_of}(F, F') \wedge$$
$$(\exists V', A'_O.\ \text{lhsS}(V', F, A'_O, PH_O) \ \wedge\ (\forall V''.\ \text{lhsS}(V'', F, A'_O, PH_O) \rightarrow TLE))) \tag{8}$$

where:

- $\text{lhsS}(V, F, A_O, PH_O) = SD(V, F) \ \wedge\ \text{act\_obs}(A_O, O, PH_O)$, represents the formula $(SD \wedge OBS' \wedge \varphi(\Delta^-))$;

[3] All proofs are available in the extended version online at https://es-static.fbk.eu/people/sambrotta/safecomp26/extended_version.pdf

Similarly to Formula (7), the minimality condition is encoded into Formula (8).

Let ϕ_S denote the BDD of Formula (8). As before, ϕ_S can be computed statically for any given set O and propositional formula $PROP$, since it is independent from Obs and Δ. Let $\phi'_S = \phi_S[ph_{o_1}/Obs(o_1), ..., ph_{o_m}/Obs(o_m)]$ be the BDD obtained from ϕ_S by substituting PH_O with Obs. This approach does not require to compute beforehand the set $CS(F, SD \wedge OBS, \top)$, because the first line of Formula (8) is dedicated to the computation of the model-based diagnoses. Then, $\Delta_r \subseteq F$ is a relevant diagnosis iff $\varphi(\Delta_r) \models \phi'_S$, meaning that Δ_r is a model of ϕ'_S. In this case, the diagnoses for a given observation can be computed by performing the above substitution on ϕ_S and by calling the enumerate_models function on ϕ'_S.

6.3 Correctness[3]

Theorem 1. *The set of minimal diagnoses of Formula (8) is equal to the set of relevant diagnoses defined by Definition 5.*

Theorem 2. *The set of minimal diagnoses of Algorithm 1 is equal to the set of relevant diagnoses defined by Definition 5.*

7 Experimental Evaluation

In this section we give a brief description of the systems used for benchmarking, data about the performance of our implementation and some insight about the obtained results.

7.1 Benchmarks

We tested our implementation on two benchmarks: the house electrical circuit example from Chapter 5.5.3 of [19] and the Triple Modular Generator (TMG) model [4].

The house electrical circuit consists of a series of wires, switches, and circuit breakers that connect the power source to lights and power outlets. The faults can affect only circuit breakers and switches. The properties to be monitored define the correct behaviour of the system - e.g., wire 2 is powered if wire 1 is powered and the switch that connects wire 1 and 2 is up.

The TMG consists of three generators that power three buses through six circuit breakers. For a bus to function, it must receive electricity from exactly one input; receiving power from two or more sources simultaneously will cause it to break. Generators and circuit breakers are controlled via commands (i.e., on/off and open/close respectively) but these components can fail: generators can get stuck at off, and breakers can get stuck in either the open or closed position. We stripped away the fault recovery modules to focus purely on detection and identification. The properties to be monitored follow the rule *"if a subset of*

generators is on, then a corresponding subset of buses must be working". Finally, an oracle provides the commands required to satisfy these conditions, assuming the system is operating without any faults.[4]

The algorithms were implemented in Python 3.12.3, using the dd library, which is a wrapper for CUDD [23]. The tests were conducted on an AMD EPYC 7413 CPU with 16 GB of RAM and a 30 minute timeout for both the construction of the BDD of the relevant diagnoses formula and the diagnosis process, and we built the BDDs of SD beforehand without a time limit. Since both benchmarks can be scaled up, we tested our algorithms on the TMG with 3, 4, and 5 generators and buses, and 3 to 12 houses electrical circuits, all connected to the same power source with sizes of O ranging from 10% to 100% the size of V, with a 10% step. The instances of the problems (assignments to V and F) were randomly generated, ensuring that $(SD \wedge OBS) \models TLE$ holds, and both algorithms were tested on the same instances.

7.2 Performance

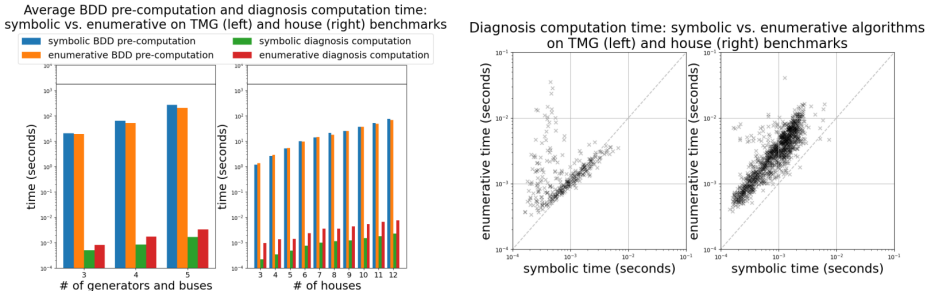

Fig. 1. Comparison between average BDD pre-computation and diagnosis computation time (left) and comparison of diagnosis computation time between both algorithms (right) on both benchmarks.

Figure 1 shows the time execution comparison between symbolic and enumerative algorithms, on the TMG and house benchmarks, for both of BDD pre-computation and diagnosis computation tasks. All problems were solved within the 30 minute time limit. Specifically, on the TMG benchmark, execution times ranged from 1.11s to 1254.19s for BDD pre-computation and from 0.0001s to 0.03s for the diagnosis task. On the house benchmark, these tasks required 0.26s to 278.15s and 0.0001s to 0.04s, respectively. The execution time for the BDD

[4] The properties take the form *premise* $\rightarrow$ *implication*, meaning that if the premise is false then the property is satisfied and the diagnosis is not needed. For this reason the oracle is implemented in such a way that it computes only sets of commands that, in absence of faults, imply that the premise is true.

pre-computation task is roughly equivalent for both algorithms across all benchmarks. We hypothesize that although the enumerative encoding is simpler than the symbolic one, its lower complexity is offset by the larger number of variables - i.e. PH_F variables unique to the enumerative approach. In the diagnosis the symbolic algorithm takes the lead, because it doesn't enumerate explicitly all of the maximal model-based cutsets.

Table 1. Percentage of model-based diagnoses affected and number of faults removed from them.

SD	# vars	# mbds	% unaffected	% affected	avg	median	min	max
house3	76	122	0.59	0.41	1.6	2	1	3
house4	101	123	0.76	0.24	2.03	2	1	3
house5	126	132	0.52	0.48	2.58	2	1	6
house6	151	120	0.54	0.46	1.74	2	1	4
house7	176	120	0.39	0.61	1.95	2	1	5
house8	201	126	0.35	0.65	3.21	3	1	7
house9	226	122	0.39	0.61	2.86	3	1	7
house10	251	120	0.38	0.62	2.33	2	1	6
house11	276	120	0.35	0.65	3.69	3	1	9
house12	301	120	0.36	0.64	2.40	2	1	5
TMG3	195	107	0.24	0.76	1.62	1	1	4
TMG4	260	141	0.2	0.8	2.77	2	1	7
TMG5	325	175	0.22	0.78	3.67	3	1	8

Table 1 shows three main sets of information on both benchmarks: the global number of model-based diagnoses on all instances for a given system, the percentage of them that were affected/unaffected, and the average, median, minimum and maximum number of faults removed from the affected model-based diagnoses. As shown, on the house benchmark on average the number of faults removed doesn't follow a specific trend, but the median stays around 2 and 3 faults removed, while on the other hand on the TMG benchmark both average and median follow an ascending trend.

These results show that our approach is able to remove irrelevant faults (when present) from the model-based diagnoses, making it easier to trigger the correct recovery mechanism in order to restore the property. Also, as shown in Figure 1 it is able to return diagnoses in a matter of milliseconds, which is a key requirement in the runtime verification setting, by querying a pre-computed BDD.

7.3 Insights

In this section we show that the situations identified by the motivating examples can occur in a real system as the TMG. In the following paragraphs we refer to

components by their initial - i.e., G for generator, B for bus, GB for generator breaker, and BB for bus breaker.

Suppose that $PROP := (G1\_\text{is}\_\text{on} \vee G2\_\text{is}\_\text{on}) \rightarrow B1\_\text{is}\_\text{working}$. The commands given to the system are $G1$ on, $G2$ and $G3$ off, $GB1$ and $GB2$ closed, $GB3$ open, $BB1$ and $BB3$ closed, and $BB2$ open. Let $O = \{G1, B1, GB2\}$, and $OBS = \{G1\_\text{is}\_\text{on}, \neg B1\_\text{is}\_\text{working}, \neg GB2\_\text{is}\_\text{closed}\}$. $PROP$ is clearly violated, since $G1$ is on and $B1$ is not working, and the MBD process returns $\{GB1\_\text{stuck@open}, GB2\_\text{stuck@open}\}$. Fault $GB2\_\text{stuck@open}$ would be sufficient in other scenarios to cause the property violation (e.g., one where $G2$ is on instead of $G1$) but not in this one, since $G2$ is off, meaning that the breakage of $GB2$ is not desired in the relevant diagnosis since it didn't contribute in this scenario. In fact, the only relevant diagnosis found in this scenario is $\{GB1\_\text{stuck@open}\}$ as expected.

Now suppose that $O = \{G1, B1, GB2, BB3\}$ and $OBS = \{G1\_\text{is}\_\text{on}, \neg B1\_\text{is}\_\text{working}, \neg GB2\_\text{is}\_\text{closed}, \neg BB3\_\text{is}\_\text{closed}\}$. The observation from the previous case has been augmented with information about $BB3$, that is open even though its command is close. This failure simply does not interfere with the property, so it should be discarded from the relevant diagnosis, and in fact this is what happens, since the only relevant diagnosis remains $\{GB1\_\text{stuck@open}\}$.

Suppose that the system must always be powered, meaning that at least one of the generators must be on. Instead of having one sensor per generator, let's say that there is only one sensor that detects if there is power in the system. For this purpose, the constraint $\text{system}\_\text{powered} \leftrightarrow (G1\_\text{is}\_\text{on} \vee G2\_\text{is}\_\text{on} \vee G3\_\text{is}\_\text{on})$ is added to the system description. Let $PROP := G1\_\text{is}\_\text{on}$, the commands to the generators are $G1$ and $G2$ on, and $G3$ off, $O = \{\text{system}\_\text{powered}\}$ and $OBS = \{\neg\text{system}\_\text{powered}\}$. $G1$ and $G2$ are both off, but since the property states that only $G1$ must be on, the breakage of $G2$ is not relevant, meaning that the only relevant diagnosis should be $\{G1\_\text{stuck@off}\}$, and this is what our definition returns.

8 Conclusion

In this paper, we addressed the problem of diagnosing runtime property violations. We highlighted that standard MBD and other related notions are not suitable for this task because they focus on the consistency between the model and observations, rather than explaining the specific violation of a monitored property. This often leads to diagnoses that include "irrelevant" faults, i.e., faults that are present but did not contribute to the violation.

To overcome this limitation, we introduced the PVD problem. We provided a formal characterization of the problem in propositional logic, distinguishing it from MBD and other causal frameworks. Central to our approach is the notion of "relevance": we apply counterfactual reasoning to filter out fault configurations that are not strictly necessary to explain the violation. We then proposed a symbolic algorithm based on propositional quantifier elimination to compute these relevant diagnoses efficiently. Our experimental evaluation confirmed the

applicability of the approach, demonstrating its ability to isolate relevant causes in systems with partial observability.

There are several promising directions for future research. While this work focused on the combinational setting, extending the framework to dynamic systems where the properties are expressed in Linear Temporal Logic (LTL) like in runtime verification [2] is a natural next step. We plan to investigate more complex fault models, including transient faults and common cause failures, to better capture real-world scenarios. Another direction is to extend diagnosability checking [21] to determine if we have enough observation to identify the relevant diagnosis of a property violation. Incorporating probabilistic information could allow for ranking diagnoses based on likelihood, helping operators prioritize recovery actions. Another promising future direction consists in refining the diagnoses by returning also information about the context in which the diagnosis occurs. Finally, we aim to explore advanced optimization techniques to further improve the scalability of the symbolic algorithms for large-scale systems.

References

1. Aiello, M., Berryman, J., Grohs, J., Schierman, J.: Run-time assurance for advanced flight-critical control systems. In: AIAA Guidance, Navigation, and Control Conference. p. 8041 (2010)
2. Bauer, A., Leucker, M., Schallhart, C.: Runtime verification for LTL and TLTL. ACM Trans. Softw. Eng. Methodol. **20**(4), 14:1–14:64 (2011)
3. Beer, I. and Ben-David, S. and Chockler, H. and Orni, A. and Trefler, R.: Explaining Counterexamples Using Causality. In: Proc. CAV. LNCS, vol. 5643, pp. 94–108 (2009)
4. Bozzano, M., Cimatti, A., Gario, M., Jones, D., Mattarei, C.: Model-based Safety Assessment of a Triple Modular Generator with xSAP. Formal Aspects of Computing **33**(2), 251–295 (2021)
5. Bozzano, M., Villafiorita, A., Åkerlund et al., O.: ESACS: An Integrated Methodology for Design and Safety Analysis of Complex Systems. In: Proc. European Safety and Reliability Conference (ESREL 2003). pp. 237–245. Balkema Publisher (2003)
6. Bozzano, M., Cimatti, A., Gario, M., Tonetta, S.: Formal design of asynchronous fault detection and identification components using temporal epistemic logic. Logical Methods in Computer Science **11** (2015)
7. Cimatti, A., Tian, C., Tonetta, S.: Assumption-based runtime verification with partial observability and resets. In: International Conference on Runtime Verification. pp. 165–184. Springer (2019)
8. Darwiche, A., Marquis, P.: A knowledge compilation map. Journal of Artificial Intelligence Research **17**, 229–264 (2002)
9. De Kleer, J. and Kurien, J.: Fundamentals of model-based diagnosis. In: IFAC Proceedings Volumes. vol. 36, pp. 25–36 (2003)
10. Dupré, D.T.: Abductive and Consistency-Based Diagnosis Revisited: a Modeling Perspective. CoRR **cs.AI/0003016** (2000), https://arxiv.org/abs/cs/0003016
11. Ferrère, T. and Maler, O. and Nickovic, D.: Trace Diagnostics Using Temporal Implicants. In: Proc. ATVA. LNCS, vol. 9364, pp. 241–258 (2015)

12. Gao, Z., Cecati, C., Ding, S.X.: A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches. IEEE transactions on industrial electronics **62**(6), 3757–3767 (2015)
13. Gössler, G. and Mari, T. and Pencolé, Y. and Travé-Massuyès, L.: Towards Causal Explanations of Property Violations in Discrete Event Systems. In: Proc. DX. pp. 1–8 (2019)
14. Halpern, J.: A modification of the Halpern-Pearl definition of causality. In: Proc. IJCAI. pp. 3022 – 3033 (2015)
15. Joshi, A., Miller, S., Whalen, M., Heimdahl, M.: A Proposal for Model-Based Safety Analysis. In: Proc. AIAA / IEEE Digital Avionics Systems Conference (DASC) (2005)
16. Misra, A., Sztipanovits, J.: A model-based failure detection, isolation and recovery system (1996)
17. Mozetič, I.: Model-based diagnosis: An overview. In: Mřík, V., Štěpánková, O., Trappl, R. (eds.) Advanced Topics in Artificial Intelligence. pp. 419–430. Springer (1992)
18. Pan, G., Vardi, M.Y.: Symbolic techniques in satisfiability solving. Journal of Automated Reasoning **35**(1), 25–50 (2005)
19. Poole, D.L., Mackworth, A.K.: Artificial Intelligence: foundations of computational agents. Cambridge University Press (2010)
20. Preist, C., Eshghi, K., Bertolino, B.: Consistency-based and abductive diagnoses as generalised stable models. Annals of Mathematics and Artificial Intelligence **11**, 51–74 (1994)
21. Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., Teneketzis, D.: Diagnosability of discrete-event systems. IEEE Trans. Autom. Control. **40**(9), 1555–1575 (1995)
22. Schierman, J., Ward, D., Dutoi, B., Aiello, A., Berryman, J., DeVore, M., Storm, W., Wadley, J.: Run-time verification and validation for safety-critical flight control systems. In: AIAA Guidance, Navigation and Control Conference and Exhibit. p. 6338 (2008)
23. Somenzi, F.: Cudd: Cu decision diagram package release 2.4. 2. University of Colorado at Boulder (2009)
24. Space Avionics Open interface Architecture: SAVOIR FDIR Handbook, vol. SAVOIR-HB-003. European Space Agency (2019), https://savoir.estec.esa.int/SAVOIRDocuments.htm
25. Zhang, Z. and et al.: Online Causation Monitoring of Signal Temporal Logic. In: Proc. CAV. LNCS, vol. 13964, pp. 62–84 (2023)