HAL
open science

Fault Ascription in Concurrent Systems

Gregor Gössler, Jean-Bernard Stefani

To cite this version:

Gregor Gössler, Jean-Bernard Stefani. Fault Ascription in Concurrent Systems. Trustworthy Global Computing, 2015, Madrid, Spain. pp.16, 10.1007/978-3-319-28766-96 . hal-01246485

HAL Id: hal-01246485

https://inria.hal.science/hal-01246485v1

Submitted on 18 Dec 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Fault Ascription in Concurrent Systems

Gregor Gössler and Jean-Bernard Stefani

INRIA Grenoble – Rhône-Alpes and Univ. Grenoble – Alpes, France

Abstract. Fault diagnosis is becoming increasingly important and difficult with the growing pervasiveness and complexity of computer systems. We propose in this paper a general semantic framework for fault ascription, a precise form of fault diagnosis that relies on counterfactual analysis for identifying necessary and sufficient causes of faults in component-based systems. Our framework relies on configuration structures to handle concurrent systems, partial and distributed observations in a uniform way. It defines basic conditions for a counterfactual analysis of necessary and sufficient causes, and it presents a refined analysis that conforms to our basic conditions while avoiding various infelicities.

1 Introduction

The increasing reliance of our modern societies on computer systems makes the diagnosis of faults in such systems a crucial necessity. In complex computer systems, for instance in large distributed systems, fault diagnosis is a difficult proposition. Several approaches to fault diagnosis have been put forward in the literature, e.g. using techniques from artificial intelligence [17, 16], from automatic control [12], or from concurrency theory [3, 9].

In this paper, we contribute to the latter line of work by developing a general framework for *fault ascription* in concurrent systems. Fault ascription, also called *blaming* [6], is a form of fault diagnosis that goes beyond the identification of *explanations*, typically understood as executions that are congruent with observed behavior, to identify *necessary* and *sufficient* causes for some observed behavior, and that can pinpoint the origin of a fault in the failure of given components to meet their specification.

Intuitively, a *necessary cause* is a set of events that must take place in order for a fault to occur in the context of given observations; a *sufficient cause* is a set of events that is enough to trigger an observed fault. These notions are reminiscent of similar notions in philosophy and legal reasoning [4, 14]. They are required in order to determine, in a complex system, which components are responsible for the occurrence of a fault, and to ascribe legal responsibility for a fault occurring in multi-vendor systems [15]. In contrast to classical fault diagnosis and fault isolation, fault ascription requires some form of *counterfactual reasoning* of the form "would f also have occurred if c had not occurred?" in order to assess the modality of causes.

Example 1. As a very simple example, consider the system depicted by the small Petri net in Figure 1, where observable transitions are colored dark blue, while

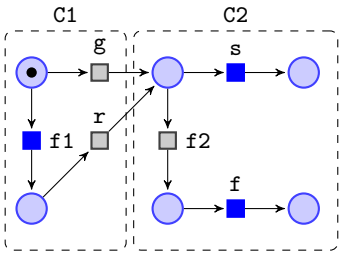

Fig. 1: Running example

unobservable ones are colored light grey. The system comprises two components, C1 and C2. C1 can either perform action g (its normal behavior), or perform action f1 (a fault), followed by action r. C2 can either perform action s (its normal behavior), or perform action f2 (a fault), followed by action f. The composition of the two components enforces the serialization of executions of C1 and C2. The overall behavior of the composition is given by the unfolding of the Petri net in Figure 1, which consists of the following event configurations: $\{\emptyset, \{\mathtt{g}\}, \{\mathtt{g},\mathtt{s}\}, \{\mathtt{g},\mathtt{f2}\}, \{\mathtt{g},\mathtt{f2},\mathtt{f}\}, \{\mathtt{f1}\}, \{\mathtt{f1},\mathtt{r}\}, \{\mathtt{f1},\mathtt{r},\mathtt{s}\}, \{\mathtt{f1},\mathtt{r},\mathtt{f2}\}, \{\mathtt{f1},\mathtt{r},\mathtt{f2},\mathtt{f}\}\}$.

Consider now an observation on the execution of this system that consists of the recording of the following observable event configurations $\{\emptyset, \{\mathtt{f1}\}, \{\mathtt{f1},\mathtt{f}\}\}$, and assume we are interested in knowing which faulty component behavior is to blame for the occurrence of the fault f. Intuitively, it seems clear that C1 is *not* to blame: indeed, even if C1 performs the faulty transition f1, the system can recover from this fault via transition r, and let C2 behave normally. It would thus appear that C2 is to blame, and that the fault that is necessary for f to occur is just f2, for *had* C2 *not misbehaved via action* f2, then the whole system *would not have experienced fault* f. In contrast, the fact that C1 had a fault f1 during the observed execution has no bearing on the final fault since *even without the fault* f1*, the fault* f2 *alone would have been sufficient to entail* f.

In this paper, we develop a general concurrency theoretic framework which formalizes the counterfactual analysis required to analyse fault ascription scenarios as in the above example. It is based on configuration structures [19], and encompasses truly concurrent executions, as well as partial and distributed observations.

The paper is organized as follows. Section 2 details the notations and operations on configuration structures we use in the paper, defines our formalization of component-based systems, of faults and of observation logs. Section 3 motivates constructions needed for fault ascription by means of a simple example, and presents our abstract framework for fault ascription. Section 4 presents an instance of our framework, with definite constructions for ascertaining necessary and sufficient causes of faults, which generalizes previous works based on traces [7, 6, 20]. Section 5 discusses several examples that illustrate various features of our framework. Section 6 discusses related work. Section 7 concludes the paper.

2 Preliminaries

Notations. We use $[n]$ to denote the finite set of naturals $\{1, \ldots, n\}$. We use boldface to denote tuples of elements taken from a given set, as in $\mathbf{s} = \langle s_1, \ldots, s_n \rangle$. We use $\bigcup S$ to denote $\bigcup_{s \in S} s$. A predicate $\mathcal{P}$ that applies to elements of a set $\mathbb{S}$ is identified with a subset of $\mathbb{S}$. In the paper, we use both set operations, e.g. $s \in \mathcal{P}$, or predicate notation, e.g. $\mathcal{P}(s)$, where appropriate.

2.1 Operations on configuration structures

Definition 1 (Configuration structure). *A* configuration structure *is a tuple* $(\mathbb{E}, \mathcal{C})$*, where* $\mathbb{E}$ *is a set (of events), and* $\mathcal{C} \subseteq 2^{\mathbb{E}}$ *is a set of subsets of* $\mathbb{E}$*, called* configurations*.*

A rooted *configuration structure* $(\mathbb{E}, \mathcal{C})$ *is such that* $\emptyset \in \mathcal{C}$*.*

We now define some operations on configuration structures.

- $(\mathbb{E}_1, \mathcal{C}_1) \parallel (\mathbb{E}_2, \mathcal{C}_2) = (\mathbb{E}, \mathcal{C})$ where $\mathbb{E} = \mathbb{E}_1 \cup \mathbb{E}_2$ and $\mathcal{C} = \{c \in 2^{\mathbb{E}} \mid c \cap \mathbb{E}_i \in \mathcal{C}_i, i = 1, 2\}$
- $(\mathbb{E}_1, \mathcal{C}_1) \cap (\mathbb{E}_2, \mathcal{C}_2) = (\mathbb{E}_1 \cap \mathbb{E}_2, \mathcal{C}_1 \cap \mathcal{C}_2)$
- $(\mathbb{E}_1, \mathcal{C}_1) \subseteq (\mathbb{E}_2, \mathcal{C}_2)$ iff $\mathbb{E}_1 \subseteq \mathbb{E}_2 \wedge \mathcal{C}_1 \subseteq \mathcal{C}_2$
- $\max \mathcal{C} = \{c \in \mathcal{C} \mid \forall c' \in \mathcal{C} : c \subseteq c' \Rightarrow c = c'\}$
- $(\mathbb{E}, \mathcal{C})_{\downarrow \mathbb{F}} = (\mathbb{E} \cap \mathbb{F}, \mathcal{C}_{\downarrow \mathbb{F}})$ where $\mathcal{C}_{\downarrow \mathbb{F}} = \{c_{\downarrow \mathbb{F}} \mid c \in \mathcal{C}\}$, and $c_{\downarrow \mathbb{F}} = c \cap \mathbb{F}$.
- Let $(\mathbb{E}, \mathcal{C})$ be a configuration structure, and let $\mathbb{F}$ be a set such that $\mathbb{E} \subseteq \mathbb{F}$. We define $c^{\uparrow \mathbb{F}} = \{c' \subseteq \mathbb{F} \mid c' \cap \mathbb{E} = c\}$, $\mathcal{C}^{\uparrow \mathbb{F}} = \bigcup_{c \in \mathcal{C}} c^{\uparrow \mathbb{F}}$, and $(\mathbb{E}, \mathcal{C})^{\uparrow \mathbb{F}} = (\mathbb{F}, \mathcal{C}^{\uparrow \mathbb{F}})$.

Remark 1. When $\mathbb{E} \subseteq \mathbb{F}$, we have by definition: $\forall d \in c^{\uparrow \mathbb{F}}, d_{\downarrow \mathbb{E}} = c$.

Definition 2 (Hasse diagram). *For a set of configurations* $\mathcal{C}$ *we define the graph* $H_{\mathcal{C}} = (V, E)$ *with vertices* $V = \mathcal{C}$ *and edges*

$$E = \{(c, c') \mid c, c' \in \mathcal{C} \wedge c \subseteq c' \wedge \forall c'' \in \mathcal{C} : c \subseteq c'' \subseteq c' \implies c = c'' \vee c'' = c'\}$$

Definition 3 (Conflict). *Let* $C = (\mathbb{E}, \mathcal{C})$ *be a configuration structure. We say that a pair of events* (e, e') *is conflictual in* C *if there exists no* $c \in \mathcal{C}$ *such that* $\{e, e'\} \subseteq c$*.* C *is conflict-free if no pair of events of* $\mathbb{E}$ *is conflictual.*

2.2 Systems and components

A *component specification* is a rooted configuration structure. A component specification is the expected behavior of an actual component. Similarly, a *system specification* is the abstraction of a system composed of a set of interacting components:

Definition 4 (System specification). *A system specification is a pair* $(\mathbf{S}, B)$*, where:*

- $\mathbf{S} = (S_i)_{i \in I}$ *is a finite tuple of component specifications* $S_i = (\mathbb{E}_i, \mathcal{C}_i)$*, where the sets* $\mathbb{E}_i$ *are assumed to be mutually disjoint, i.e.* $\forall i, j \in I,\ i \neq j \implies \mathbb{E}_i \cap \mathbb{E}_j = \emptyset$.
- $B = (\mathbb{B}, \mathcal{B})$ *is a rooted configuration structure, where* $\mathbb{B} \subseteq \mathbb{E} = \bigcup_{i \in I} \mathbb{E}_i$.

We use the word "component" in a broad sense to denote part of a system behavior. The configuration structure B plays the role of a behavioral model: it is used to express assumptions and constraints on the possible (correct and incorrect) behaviors. In particular, B may be used to model synchronization and coordination between components. The component specifications define the *correct* behavior of components, in the sense of *normality* of [10]; the actual component behavior may violate those specifications. Note that B may contain behaviors not in $S = \|_{i \in I} S_i$, for instance events in $(\bigcup_{i \in I} \mathbb{E}_i) \setminus (\bigcup_{i \in I} \bigcup \mathcal{C}_i)$. Conversely, part of the behaviors of S may not be feasible according to B.

Remark 2. An alternate definition for a system specification that explicitly accounts for events $\mathbb{E}^*$ not appearing in component specifications could be defined as follows:
System specification – alternate definition. A system specification is a pair $(\mathbf{S}, B)$, where:

- $\mathbf{S} = (S_i)_{i \in I}$ is a finite tuple of component specifications $S_i = (\mathbb{E}_i, \mathcal{C}_i)$
- $B = (\mathbb{B}, \mathcal{B})$ is a rooted configuration structure where $\mathbb{B} \subseteq \mathbb{E} \cup \mathbb{E}^*$, where $\mathbb{E} = \bigcup_{i \in I} \mathbb{E}_i$ and $\mathbb{E}^* \cap \mathbb{E} = \emptyset$.

However, one can always transform a system specification $(\mathbf{S}, B)$ according to the above definition into a system specification $\mathtt{A}(\mathbf{S}, B)$ complying with Definition 4: it suffices to define $\mathtt{A}(\mathbf{S}, B) = (\mathbf{S}', B)$, where $\mathbf{S}' = \mathbf{S}, \top_{E^*}$ and $\top_{E^*} = (\mathbb{E}^*, 2^{\mathbb{E}^*})$.

2.3 Faults and logs

Given a system specification $(\mathbf{S}, B)$ with events in $\mathbb{E}$, a *fault* is an incorrect behavior. To define a fault, we require a predicate $\mathcal{P} \subseteq 2^{\mathbb{E}}$, which characterizes the correct configurations. In this paper, we focus on safety properties, using the standard transition system associated with a configuration structure under the asynchronous interpretation [19]. A fault occurs whenever $\mathcal{P}$ is violated. We require that system specifications be *consistent with respect to the given property*, which amounts to say that when all the components behave according to their specification, the system is not at fault. Formally:

Definition 5 (Consistent system specification). *A consistently specified system is a tuple* $(\sigma, \mathcal{P})$ *where* $\sigma = (\mathbf{S}, B)$ *is a system specification with* $\mathbf{S} = ((\mathbb{E}_i, \mathcal{C}_i))_{i \in [n]}$*, and* $\mathcal{P}$ *is a predicate such that* $\|_{i \in [n]} \mathcal{C}_i \cap B \subseteq \mathcal{P}$.

Under a consistent specification, property $\mathcal{P}$ may be violated only if at least one of the components violates its specification. In contrast, the violation of a component specification does not necessarily entail a violation of $\mathcal{P}$. This is useful e.g. to model systems that tolerate certain component failures. Throughout this paper we consider only consistent system specifications.

Remark 3. In addition to being consistent, a meaningful specification of a system should satisfy $\|_{i\in[n]} S_i \cap B \neq \emptyset$ — i.e. B should allow for some correct behavior of its components —, although this is not required for the analysis described below.

Observations of the execution of a system specified by $(\mathbf{S}, B)$, with events $\mathbb{E}$, take the form of *logs*.

Definition 6 (Logs, observables and detected faults). *A log of a system with specification* $(\mathbf{S}, B)$ *is a rooted conflict-free configuration structure* $(\mathbb{O}, \mathcal{L})$ *such that* $(\mathbb{O}, \mathcal{L}) \subseteq B_{\downarrow\mathbb{O}}$*, with* $\mathbb{O} \subseteq \mathbb{E}$*. We call* $\mathbb{O}$ *the set of* observable events *or* observables*. Given a consistently specified system* $((\mathbf{S}, B), \mathcal{P})$ *and a set of observables* $\mathbb{O}$*, we say a fault is* detected *by a log* $(\mathbb{O}, \mathcal{L})$ *whenever* $\mathcal{L} \not\subseteq \mathcal{P}_{\downarrow\mathbb{O}}$*.*

Definition 7 (Filtering $\odot$). *Let* $L = (\mathbb{O}, \mathcal{L})$ *and* $B = (\mathbb{B}, \mathcal{B})$ *be two configuration structures such that* $\mathbb{O} \subseteq \mathbb{B}$*. We define the filter of* B *by* L*, noted* $L \odot B$*, as follows:*

$$L \odot B = \left\{ c \in \mathcal{L}^{\uparrow\mathbb{B}} \cap \mathcal{B} \mid \exists c' \in \mathcal{B} : c \cup \bigcup \mathcal{L} \subseteq c' \right\}$$

The filtering operation extracts configurations from B that are compatible with observations provided by L, avoiding introducing configurations that would be inconsistent with observations (because of conflicts between unobservable events and observed events).

Example 2. For $B = (\mathbb{B}, \mathcal{B})$ with $\mathbb{B} = \{\tau, a, b\}$, $\mathcal{B} = \left\{\emptyset, \{\tau\}, \{a\}, \{a, b\}\right\}$, $\mathbb{O} = \{a\}$, and a log $L = (\mathbb{O}, \mathcal{L})$ with $\mathcal{L} = \left\{\emptyset, \{a\}\right\}$ we have $L \odot B = \left\{\emptyset, \{a\}, \{a, b\}\right\}$. The configuration $\{\tau\}$ is consistent with the observed configuration $\emptyset \in \mathcal{L}$ but inconsistent with the observation $\{a\}$, hence we do not have $\{\tau\}$ in $L \odot B$.

We use filtering $L \odot B$, as in the example above, to retrieve explanations for the observed behavior recorded in a log. One might want to refine the definition of $L \odot B$ so as to be more precise concerning inferred non-observable behavior, that is, eliminate configurations that are not consistent with observed configurations in the log. This is standard practice in fault diagnosis [5]. However, for the sake of simplicity, we will stick in this paper to the definition of filtering given in Section 2, and the simple consistency check it provides.

Example 3. Figure 1 illustrates an example system specification. The system B is specified by the unfolding of the Petri net in the figure (following e.g. [19]). The specification of component `C1` is given by the configurations $\left\{\emptyset, \{\mathtt{g}\}\right\}$ built on events $\mathbb{E}_1 = \{\mathtt{g}, \mathtt{f1}, \mathtt{r}\}$. The specification of component `C2` is given by the configurations $\left\{\emptyset, \{\mathtt{s}\}\right\}$ over $\mathbb{E}_2 = \{\mathtt{s}, \mathtt{f2}, \mathtt{f}\}$. The behavior B adds the faulty transitions $\mathtt{f1}, \mathtt{f2}, \mathtt{f}$ to the behavior of components, as well as the synchronization constraint forcing the occurrence of transitions `s` or `f2` after the occurrence of transitions `g` or `r`. The set of events of B is $\mathbb{B} = \{\mathtt{g}, \mathtt{f1}, \mathtt{r}, \mathtt{s}, \mathtt{f2}, \mathtt{f}\}$. The configurations of B are $\mathcal{B} = \left\{\emptyset, \{\mathtt{g}\}, \{\mathtt{g}, \mathtt{s}\}, \{\mathtt{g}, \mathtt{f2}\}, \{\mathtt{g}, \mathtt{f2}, \mathtt{f}\}, \{\mathtt{f1}\}, \{\mathtt{f1}, \mathtt{r}\}, \{\mathtt{f1}, \mathtt{r}, \mathtt{s}\}, \{\mathtt{f1}, \mathtt{r}, \mathtt{f2}\}, \{\mathtt{f1}, \mathtt{r}, \mathtt{f2}, \mathtt{f}\}\right\}$. Observables $\mathbb{O}$ are events $\{\mathtt{f1}, \mathtt{s}, \mathtt{f}\}$, marked in dark blue in Figure 1. The configurations of the log L in this example are $\{\emptyset, \{\mathtt{f1}\}, \{\mathtt{f1}, \mathtt{f}\}\}$.

Remark 4. To simplify notations in the following sections, given a system specification and its behavioral model $(\mathbb{B}, \mathcal{B})$, we often write $\mathcal{P}$ and in general sets of configurations $\mathcal{X} \subseteq \mathcal{B}$ using logical formulas with events as propositional variables indicating the occurrence of these events. For instance, $\mathcal{X} = f_1$ stands for $\mathcal{X} = \{c \in \mathcal{B} \mid f_1 \in c\}$.

3 A General Framework for Fault Ascription

In this section we define causality of component behaviors for the violation of a system-level property. We assume the following inputs to be given:

- a system specification $\sigma = (\mathbf{S}, B)$ with component specifications $S = ((\mathbb{E}_i, \mathcal{C}_i))_{i \in I}$ and $B = (\mathbb{B}, \mathcal{B})$, with $\mathbb{B} \subseteq \bigcup_{i \in I} \mathbb{E}_i$;
- a set $\mathbb{O} \subseteq \mathbb{B}$ of observable events;
- a property $\mathcal{P}$ such that $(\sigma, \mathcal{P})$ is consistently specified;
- a log $L = (\mathbb{O}, \mathcal{L})$;
- a set $\mathcal{X} \subseteq \mathcal{B} \setminus \|_i \mathcal{C}_i$ of faulty configurations to be checked for causality.

Notice that the set of faulty configurations $(L \odot B) \setminus \|_i \mathcal{C}_i$ is, in general, incomparable with $(L \odot B) \setminus \mathcal{P}$: a violation of $\mathcal{P}$ does not need to occur simultaneously with the violation of component specifications.

In order to verify whether the violations $\mathcal{X}$ are a cause for the violation of $\mathcal{P}$ in L, we construct the (hypothetical) system behavior where the failures in $\mathcal{X}$ and their effects on the observed execution do not occur, under the contingency that the parts of the log that are not impacted by $\mathcal{X}$ remain consistent with the actual observations. We then verify whether all obtained behaviors satisfy $\mathcal{P}$. Let $\mathsf{CF}_{\mathcal{X}}$ ("counterfactuals with respect to $\mathcal{X}$") be an operation on configuration structures with the following property, for $L = (\mathbb{O}, \mathcal{L})$:

$$\begin{cases} \mathsf{CF}_{\mathcal{X}}(\mathcal{L}) \subseteq \mathcal{B} \setminus \mathcal{X} \wedge & \\ \qquad \forall i \in [n] : \big((L \odot B)_{\downarrow \mathbb{E}_i} \subseteq \mathcal{C}_i \Rightarrow \mathsf{CF}_{\mathcal{X}}(\mathcal{L})_{\downarrow \mathbb{E}_i} \subseteq \mathcal{C}_i\big) & \text{if } (L \odot B) \cap \mathcal{X} \neq \emptyset \\ \mathsf{CF}_{\mathcal{X}}(\mathcal{L}) = L \odot B & \text{if } (L \odot B) \cap \mathcal{X} = \emptyset \end{cases} \quad (1)$$

Intuitively, the set of configurations $\mathsf{CF}_{\mathcal{X}}(\mathcal{L})$ models the system behavior "if $\mathcal{X}$ had not happened", while avoiding the introduction of new component failures.

For a given $\mathsf{CF}_{\mathcal{X}}$ we can now define the notions of necessary and sufficient causality.

Definition 8 (Necessary causality). *Consider a consistently specified system $\big((\mathbf{S}, B), \mathcal{P}\big)$ with component specifications $\mathbf{S} = \langle S_1, ..., S_n \rangle$ and $S_i = (\mathbb{E}_i, \mathcal{C}_i)$, a log $L = (\mathbb{O}, \mathcal{L})$ with $L \odot B \not\subseteq \mathcal{P}$, and a predicate $\mathcal{X}$ of faulty configurations. $\mathcal{X}$ is a* necessary cause *of the violation of $\mathcal{P}$ in L (with respect to counterfactual operator CF) if $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}) \subseteq \mathcal{P}$. The faults of a subset $\mathcal{I}$ of components are a necessary cause if $\mathcal{X} \stackrel{\Delta}{=} \{c \in \mathcal{B} \mid \exists i \in \mathcal{I} : c_{\downarrow \mathbb{E}_i} \notin \mathcal{C}_i\}$ is a necessary cause.*

That is, the incorrect configurations in $\mathcal{X}$ are a necessary cause for the violation of $\mathcal{P}$ in L if, in the counterfactual scenarios where configurations in $\mathcal{X}$ do not occur, $\mathcal{P}$ would have been satisfied.

The definition of necessary causality above is parameterized by a counterfactual operator CF. We can check that, regardless of the counterfactual operator used, this definition agrees with a naive notion of necessary causality as necessary condition, defined as follows:

Definition 9 (Naive necessary causality). *Let $B = (\mathbb{B}, \mathcal{B})$ be a system specification, and let $\mathcal{P} \subseteq 2^{\mathbb{B}}$ be a property. Let $x \subseteq \mathbb{B}$ and $\mathcal{Y} \subseteq 2^{\mathbb{B}}$. We say that x is a naive necessary cause for the violation of $\mathcal{P}$ in $\mathcal{Y}$, if x appears as a subset of all faulty configurations in $\mathcal{Y}$, formally: $\forall c \in \mathcal{Y} \setminus \mathcal{P}, x \subseteq c$.*

Example 4. Naive necessary causality is not satisfactory for analyzing causality *relative* to the behavior recorded in the log. To see this, consider two components with specifications $(\mathbb{E}_i, \mathcal{C}_i)$ with $\mathbb{E}_i = \{a_i, f_i\}$, $\mathcal{C}_i = \big\{\emptyset, \{a_i\}\big\}$, $i = 1, 2$, the behavioral model $\mathcal{B} = 2^{\mathbb{E}}$ with $\mathbb{E} = \mathbb{E}_1 \cup \mathbb{E}_2$, and the property $\mathcal{P} = \neg(f_1 \vee f_2)$: a failure f_i of any of the components violates $\mathcal{P}$. For the log $L = (\mathbb{E}, \mathcal{L})$ with $\mathcal{L} = \big\{\emptyset, \{a_1\}, \{a_1, f_2\}\big\}$, f_2 is not a naive necessary cause with respect to $\mathcal{B}$, but it is a necessary cause in L (that is, under the contingency that f_1 has not occurred) since $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}) \subseteq \mathcal{P}$.

Proposition 1 (Completeness with respect to naive necessary causality). *Let $\big((\mathbf{S}, (\mathbb{B}, \mathcal{B})), \mathcal{P}\big)$ be a consistently specified system and $L = (\mathbb{O}, \mathcal{L})$ be a log as specified in Definition 8. Assume that there exists $e \in \mathbb{B}$ such that $\{e\}$ is a naive necessary cause for the violation of $\mathcal{P}$ in $\mathcal{B}$. Then $\mathcal{X} = e$ is a necessary cause of the violation of $\mathcal{P}$ in L.*

Proof. By definition of naive necessary causality, e belongs to all faulty configurations. Hence, we have $\mathcal{B} \setminus \mathcal{P} = \mathcal{X}$. Since $L \odot B \not\subseteq \mathcal{P}$ (by Definition 8), we have $(L \odot B) \cap \mathcal{X} \neq \emptyset$ and thus $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}) \subseteq \mathcal{P}$, as required.

Example 5. Returning to our simple example in Figure 1, it is easy to check that $\mathcal{X} = \mathtt{f2}$ is identified as a naive necessary cause for the violation of $\mathcal{P} = \neg f$ in $\mathcal{B}$, and as a necessary cause for the violation of $\mathcal{P}$ in L using any counterfactual operation meeting Condition 1. Note that in a system consisting of two copies of the Petri Net in Figure 1 running in parallel, with the second copy having primed events $\mathtt{x}'$ where the first has event $\mathtt{x}$, $\mathcal{X} = \mathtt{f2}$ is still identified as a necessary cause for the violation of $\mathcal{P}' = \mathtt{f} \vee \mathtt{f}'$ in L using any counterfactual operation meeting Condition 1, but is *not* a naive necessary cause for the violation of $\mathcal{P}'$ in $\mathcal{B} \parallel \mathcal{B}'$.

Definition 10 (Inevitable). *Given sets $\mathcal{C}$, $\mathcal{C}'$ of configurations with $\emptyset \in \mathcal{C}$, we call $\mathcal{C}'$* inevitable *in $\mathcal{C}$ if for any $c \in \max \mathcal{C}$, any path from $\emptyset$ to c in the Hasse diagram $H_{\mathcal{C}}$ transits by some configuration in $\mathcal{C}'$, and only a finite subset of $\mathcal{C}$ is reachable from $\emptyset$ in $H_{\mathcal{C}}$ without transiting by some configuration in $\mathcal{C}'$.*

$\mathcal{C}$ is inevitably faulty *with respect to a predicate $\mathcal{P}$ if $\mathcal{C} \setminus \mathcal{P}$ — that is, a violation of $\mathcal{P}$ — is inevitable in $\mathcal{C}$.*

Intuitively, a set of configurations is inevitably faulty with respect to $\mathcal{P}$ if all its maximal elements can only be reached through some intermediate configuration violating $\mathcal{P}$.

The definition of *sufficient causality* is dual to necessary causality, where in the alternative worlds we remove the failures *not in* $\mathcal{X}$ and verify whether $\mathcal{P}$ is *still violated*. In order for the definition to correctly cope with configurations simultaneously encompassing component failures within and outside of $\mathcal{X}$, we only define sufficient causality on the level of components, rather than faulty configurations.

Definition 11 (Sufficient cause). *Given*

- *a consistently specified system* $\big((\mathbf{S}, B), \mathcal{P}\big)$ *with* $\mathbf{S} = \langle S_1, ..., S_n \rangle$ *and* $S_i = (\mathbb{E}_i, \mathcal{C}_i)$*,*
- *a log* $L = (\mathbb{O}, \mathcal{L})$ *such that* $L \odot B$ *is inevitably faulty with respect to* $\mathcal{P}$*; and*
- *a subset* $\mathcal{I} \subseteq [n]$ *of components,*

the failures of components in $\mathcal{I}$ *are a sufficient cause for the violation of* $\mathcal{P}$ *in* L *if with* $\mathcal{X} \stackrel{\Delta}{=} \{c \in \mathcal{B} \mid \exists i \in [n] \setminus \mathcal{I} : c_{\downarrow \mathbb{E}_i} \notin \mathcal{C}_i\}$*,* $\mathsf{CF}_{\mathcal{X}}(\mathcal{L})$ *is inevitably faulty with respect to* $\mathcal{P}$*.*

That is, the failures of components in $\mathcal{I}$ are a sufficient cause for the violation of $\mathcal{P}$ if for the counterfactual scenarios where failures of components other than $\mathcal{I}$ do not occur, a violation of $\mathcal{P}$ is still inevitable.

Remark 5. One may wonder whether we have for sufficient causes an equivalent of Proposition 1. Unfortunately, we don't. We can certainly mirror what we did with necessary causality, and define a notion of sufficient cause as *sufficient condition* for a failure, i.e. say that some configuration c is a sufficient cause for a failure f if the occurrence of the events in c inevitably leads to the occurrence of failure f. The two definitions of sufficient and of naive sufficient causality in general lead to different identification of sufficient causes, though. This is because the configuration structures on which inevitable faultiness is verified, are incomparable.

Proposition 2 (Soundness). *If* $\mathcal{X}$ *is a necessary cause for the violation of* $\mathcal{P}$ *in the log* $L = (\mathbb{O}, \mathcal{L})$ *then* $(L \odot B) \cap \mathcal{X} \neq \emptyset$*.*

If the failures of components $\mathcal{I}$ *are a sufficient cause for the violation of* $\mathcal{P}$ *in the log* $L = (\mathbb{O}, \mathcal{L})$ *then* $(L \odot B)_{\downarrow \mathbb{E}_i} \not\subseteq \mathcal{C}_i$ *for some* $i \in \mathcal{I}$*.*

Intuitively, any cause contains some configuration of the log where at least one component has violated its specification.

Proof. Necessary causality: Let $\mathcal{X}$ be such that $(L \odot B) \cap \mathcal{X} = \emptyset$. We show that $\mathcal{X}$ is not a cause. Let $\mathcal{C} = \mathsf{CF}_{\mathcal{X}}(\mathcal{L})$. By hypothesis on $\mathsf{CF}_{\mathcal{X}}$ we have $\mathcal{C} = L \odot B$, thus $\neg(\mathcal{C} \subseteq \mathcal{P})$, and $\mathcal{X}$ is not a cause for the violation of $\mathcal{P}$ in L.

Sufficient causality: Let $\mathcal{I}$ be such that $\forall i \in \mathcal{I}$, $(L \odot B)_{\downarrow \mathbb{E}_i} \subseteq \mathcal{C}_i$. We have $\mathcal{X} = \{c \in \mathcal{B} \mid \exists i \in [n] \setminus \mathcal{I} : c_{\downarrow \mathbb{E}_i} \notin \mathcal{C}_i\}$. Let $\mathcal{C} = \mathsf{CF}_{\mathcal{X}}(\mathcal{L})$. By hypothesis on $\mathsf{CF}_{\mathcal{X}}$ we have $\mathcal{C} \subseteq \mathcal{B} \setminus \mathcal{X} \ \wedge \ \forall i \in [n] : \big((L \odot B)_{\downarrow \mathbb{E}_i} \subseteq \mathcal{C}_i \Rightarrow \mathsf{CF}_{\mathcal{X}}(\mathcal{L})_{\downarrow \mathbb{E}_i} \subseteq \mathcal{C}_i\big)$, hence $\mathcal{C} \subseteq \mathcal{B} \cap \|_i \mathcal{C}_i$. By consistency of the specification it follows that $\mathsf{CF}_{\mathcal{X}}(\mathcal{L})$ is not inevitably faulty with respect to $\mathcal{P}$, hence the failures of components in $\mathcal{I}$ are not a sufficient cause.

Proposition 3 (Completeness). *Each violation (resp. inevitable violation) of $\mathcal{P}$ in $L \odot B$ has a necessary (resp. sufficient) cause.*

Proof. Necessary causality: Let $\mathcal{X} = \mathcal{B} \setminus \|_i \mathcal{C}_i$ and $\mathcal{C} = \mathsf{CF}_{\mathcal{X}}(\mathcal{L})$. By hypothesis on $\mathsf{CF}_{\mathcal{X}}$ we have $\mathcal{C} \subseteq \mathcal{B} \setminus \mathcal{X}$. Thus, $\mathcal{C}$ contains only observations consistent with executions where all components behave correctly. By consistency of the specification we have $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}) \subseteq \mathcal{P}$, thus $\mathcal{X}$ is a necessary cause for the violation of $\mathcal{P}$ in L.

Sufficient causality: Suppose that $L \odot B$ is inevitably faulty with respect to $\mathcal{P}$, and let $\mathcal{I} = \{i \in [n] \mid (L \odot B)_{\downarrow \mathbb{E}_i} \not\subseteq \mathcal{C}_i\}$. We have $\mathcal{X} = \emptyset$. Let $\mathcal{C} = \mathsf{CF}_{\mathcal{X}}(\mathcal{L})$. By hypothesis on $\mathsf{CF}_{\mathcal{X}}$ we have $\mathcal{C} = L \odot B$. Since $L \odot B$ is inevitably faulty with respect to $\mathcal{P}$, so is $\mathcal{C}$, hence $\mathcal{X}$ is a sufficient cause for the violation of $\mathcal{P}$ in L.

Proposition 4. *If the failures of a subset $\mathcal{I}$ of components are a necessary (resp. sufficient) cause then the failures of components $[n] \setminus \mathcal{I}$ are not a sufficient (resp. necessary) cause.*

Proof. If $\mathcal{X} = \{c \in \mathcal{B} \mid \exists i \in \mathcal{I} : c_{\downarrow \mathbb{E}_i} \notin \mathcal{C}_i\}$ is a necessary cause then $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}) \subseteq \mathcal{P}$, thus $\mathsf{CF}_{\mathcal{X}}(\mathcal{L})$ is not inevitably faulty, thus $[n] \setminus \mathcal{I}$ is not a sufficient cause.

Conversely, if $\mathcal{I}$ is a sufficient cause then $\mathsf{CF}_{\mathcal{X}}(\mathcal{L})$, with $\mathcal{X} = \{c \in \mathcal{B} \mid \exists i \in [n] \setminus \mathcal{I} : c_{\downarrow \mathbb{E}_i} \notin \mathcal{C}_i\}$, is inevitably faulty with respect to $\mathcal{P}$, hence $\neg(\mathsf{CF}_{\mathcal{X}}(\mathcal{L}) \subseteq \mathcal{P})$, and $[n] \setminus \mathcal{I}$ is not a necessary cause.

4 An Instantiation

Following Stalnaker's and Lewis' *closest world assumption* [18, 14], $\mathcal{X}$ is a cause for the violation of $\mathcal{P}$ if among the worlds (that is, alternative behaviors) where $\mathcal{X}$ is true, some world where $\mathcal{P}$ is violated is *closer* to the actual world L than any world where $\mathcal{P}$ holds. In this section we first illustrate with Example 6 why "closeness" of the counterfactuals is important also in our framework, and then propose a concrete definition for $\mathsf{CF}_{\mathcal{X}}$. The goal of this instantiation is to construct from L — in the spirit of the closest world assumption — a counterfactual configuration structure where exactly the faults $\mathcal{X}$ to be checked for causality and their effects are eliminated and replaced with correct behaviors.

The following example illustrates that, with the extreme choices of $\mathsf{CF}_{\mathcal{X}}$ satisfying Condition (1), Definitions 8 and 11 do not pinpoint the expected cause.

Example 6 (Extreme choices of $\mathsf{CF}_{\mathcal{X}}$). Let us illustrate why the extreme choices of $\mathsf{CF}_{\mathcal{X}}$ satisfying Condition (1) are not useful in practice.

First, take $\mathsf{CF}^1_{\mathcal{X}}(\mathcal{L}) = \{\emptyset\}$ and consider the component alphabets $\mathbb{E}_i = \{f_i\}$ and component specifications $\mathcal{C}_i = \{\emptyset\}$, $i = 1, 2$, the behavioral model $\mathcal{B} = \big\{\emptyset, \{f_1\}, \{f_2\}, \{f_2, f_3\}\big\}$, the property $\mathcal{P} = \neg(f_1 \vee f_2)$, the log $L = (\mathbb{E}, \mathcal{L})$ with $\mathbb{E} = \mathbb{E}_1 \cup \mathbb{E}_2$ and $\mathcal{L} = \big\{\emptyset, \{f_1\}, \{f_1, f_2\}\big\}$, and $\mathcal{X} = f_1$. Intuitively, both components produce a fault event f_i, each of which is sufficient to violate $\mathcal{P}$. The counterfactual configuration structure where $\mathcal{X}$ does not happen is $\mathsf{CF}^1_{\mathcal{X}}(\mathcal{L}) = \{\emptyset\} \subseteq \mathcal{P}$, thus $\mathcal{X}$ is (wrongly) considered as a necessary cause. This is because

$\mathsf{CF}^1_{\mathcal{X}}$ discards all configurations of $\mathcal{L}$, resulting in complete loss of information about the observed behavior of the second component. In other words, $\mathsf{CF}^1_{\mathcal{X}}$ is not a closest world to L where $\mathcal{X}$ does not happen. Similarly, f_2 is not recognized as a sufficient cause since $\mathsf{CF}^1_{\mathcal{X}}$ is not inevitably faulty with respect to $\mathcal{P}$.

Now take $\mathsf{CF}^2_{\mathcal{X}}(\mathcal{L}) = \{c \in \mathcal{B} \setminus \mathcal{X} \mid \forall i \in [n] : (L \odot B)_{\downarrow \mathbb{E}_i} \subseteq \mathcal{C}_i \Rightarrow c_{\downarrow \mathbb{E}_i} \in \mathcal{C}_i\}$ and consider the component specifications $(\mathbb{E}_i, \mathcal{C}_i)$, $i = 1, 2$, with $\mathbb{E}_1 = \{f_1\}$, $\mathbb{E}_2 = \{f_2, f_3\}$, and $\mathcal{C}_i = \{\emptyset\}$, $i = 1, 2$, the behavioral model $\mathcal{B} = 2^{\mathbb{E}}$ with $\mathbb{E} = \mathbb{E}_1 \cup \mathbb{E}_2$, the property $\mathcal{P} = \neg(f_1 \vee f_3)$, the log $(\mathbb{E}, \mathcal{L}')$ with $\mathcal{L}' = \big\{\emptyset, \{f_1\}, \{f_2\}\big\}$, and $\mathcal{X} = f_1$. Intuitively, the first component is faulty and violates $\mathcal{P}$, whereas the second component is faulty but does not contribute to the violation of $\mathcal{P}$. The counterfactual configuration structure where $\mathcal{X}$ does not happen is $\mathsf{CF}^2_{\mathcal{X}}(\mathcal{L}') = \big\{\emptyset, \{f_2\}, \{f_3\}, \{f_2, f_3\}\big\}$. The occurrence of f_3 violates $\mathcal{P}$, thus $\mathcal{X}$ is (wrongly) not considered as a necessary cause. This is because $\mathsf{CF}^2_{\mathcal{X}}$ encompasses all configurations not satisfying $\mathcal{X}$, including those where the second component fails with f_3, in contrast to its observed behavior.

We now develop a concrete definition of $\mathsf{CF}_{\mathcal{X}}$ where the set of counterfactuals is represented by a configuration structure computed by the composition of a *pruning* and a *grafting* operations. Pruning restricts the faulty configurations in $(L \odot B) \cap \mathcal{X}$ to the maximal non-faulty sub-configurations, while remembering the original configuration.

Definition 12 (Pruning). *The* pruning *of a log $L = (\mathbb{O}, \mathcal{L})$ with respect to a predicate $\mathcal{X}$ is*

$$L/\mathcal{X} = \big\{(c', c \setminus c') \mid c \in L \odot B \wedge c' \textit{ is a maximal subset of } c \textit{ s.t.} \\ \neg\mathcal{X}(c') \ \wedge\ \forall i \in [n] : c'_{\downarrow \mathbb{E}_i} \in \mathcal{B}_{\downarrow \mathbb{E}_i}\big\}$$

Example 7. For $L = (\mathbb{O}, \mathcal{L})$ with $\mathcal{L} = \big\{\emptyset, \{f_1, f_2\}\big\}$, $\mathcal{B} = \big\{\emptyset, \{f_1\}, \{f_2\}, \{f_1, f_2\}\big\}$, and $\mathcal{X} = \big\{\{f_1, f_2\}\big\}$ we have $L/\mathcal{X} = \big\{(\emptyset, \emptyset), (\{f_1\}, \{f_2\}), (\{f_2\}, \{f_1\})\big\}$.

Before instantiating the counterfactual operator CF, we introduce an auxiliary function that will be used to remove the effects of a set $\mathcal{X}$ of faulty configurations in the counterfactual model $\mathsf{CF}_{\mathcal{X}}(\mathcal{L})$.

Definition 13 (Predecessor closure, wf). *Given sets of configurations $\mathcal{C}$ and $\mathcal{C}'$, a configuration $c \in \mathcal{C}$ is* predecessor-closed *in $\mathcal{C}$ with respect to $\mathcal{C}'$ if $c = \emptyset$ or* $\max\{c' \in \mathcal{C}' \mid c' \subsetneq c\} \cap \mathcal{C} \neq \emptyset$. *We say that $\mathcal{C}$ is predecessor-closed with respect to $\mathcal{C}'$ if all its elements are predecessor-closed with respect to $\mathcal{C}'$. Let $wf_{\mathcal{C}'}(\mathcal{C})$ be the greatest transitively predecessor-closed subset of $\mathcal{C}$ with respect to $\mathcal{C}'$.*

Intuitively, a configuration $c \in \mathcal{C}$ is *predecessor-closed* in $\mathcal{C}$ with respect to $\mathcal{C}'$ if some immediate predecessor of c in $\mathcal{C}'$ is in $\mathcal{C}$.

Example 8. For $\mathcal{C} = \big\{\emptyset, \{a, b\}, \{d\}, \{c, d\}\big\}$ and $\mathcal{C}' = \big\{\emptyset, \{a\}, \{a, b\}, \{c\}, \{c, d\}\big\}$ we have $wf_{\mathcal{C}'}(\mathcal{C}) = \big\{\emptyset, \{d\}, \{c, d\}\big\}$.

The goal of grafting is to construct from $L/\mathcal{X}$ a configuration structure modeling the alternative behaviors where the configurations in $\mathcal{X}$ do not occur.

Definition 14 (Grafting). *Let* $\mathbf{S}$ *be a vector of component specifications* $S_i = (\mathbb{E}_i, \mathcal{C}_i)$*. The* grafting *of a set of tuples* S *— obtained by pruning* $\mathcal{L}$ *with respect to a set* $\mathcal{X}$ *— with a set of configurations* $\mathcal{C}$ *is* $S \rhd_{\mathcal{L},\mathcal{X},\mathbf{S}} \mathcal{C} = wf_{\mathcal{B}}(Y)$ *where*

$$Y = \{c \mid (c, \emptyset) \in S\} \cup \big\{c \in \mathcal{C} \setminus \mathcal{X} \mid \exists (c', c'') \in S : c'' \neq \emptyset \wedge c' \subseteq c \wedge \tag{2}$$

$$\forall i : \big(c''_{\downarrow \mathbb{E}_i} = \emptyset \Rightarrow c_{\downarrow \mathbb{E}_i} = c'_{\downarrow \mathbb{E}_i}\big) \wedge \tag{3}$$

$$\big((c' \cup c'')_{\downarrow \mathbb{E}_i} \in \mathcal{C}_i \Rightarrow c_{\downarrow \mathbb{E}_i} \in \mathcal{C}_i\big) \wedge \tag{4}$$

$$\big(c_{\downarrow \mathbb{E}_i} \notin \mathcal{C}_i \Rightarrow c_{\downarrow \mathbb{E}_i} = (c' \cup c'')_{\downarrow \mathbb{E}_i}\big)\big\} \tag{5}$$

That is, the set Y is the union of the unpruned original configurations where $\mathcal{X}$ does not hold, and the configurations of $\mathcal{C}$ that are supersets of some pruned configuration (line (2). For the latter set, Condition (3) ensures that, for each component, only pruned configurations are grafted. Component configurations of the log that have not been pruned could be observed the same way in the counterfactual model, and are not grafted to stay as close as possible to the observed log. Condition (4) ensures the extensions to preserve invariance of the component specifications, that is, no new component failures are introduced. Condition (5) makes sure that configurations of a component that violate its specification are not grafted since in the absence of a fault model — representing all possible incorrect behaviors — we have no knowledge about how to extend faulty behaviors.

A path in $H_{\mathcal{B}}$ from $\emptyset$ to a configuration $c \in L \odot B$ can be seen as an *explanation* of how c may have been reached in L. Intuitively, configurations that cannot be explained in Y represent effects of $\mathcal{X}$ that would not have occurred without $\mathcal{X}$. The role of $wf_{\mathcal{B}}$ in grafting is to remove those configurations.

Proposition 5. *If* $L = (\mathbb{O}, \mathcal{L})$ *is such that* $L \odot B$ *is predecessor-closed with respect to* $\mathcal{B}$ *then with* $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}) = (L/\mathcal{X}) \rhd_{\mathcal{L},\mathcal{X},\mathbf{S}} \mathcal{B}$*, Condition* (1) *is satisfied.*

Proof. If $(L \odot B) \cap \mathcal{X} \neq \emptyset$, the fact that $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}) \subseteq \mathcal{B} \setminus \mathcal{X} \;\wedge\; \forall i \in [n] : \big((L \odot B)_{\downarrow \mathbb{E}_i} \subseteq \mathcal{C}_i \Rightarrow \mathsf{CF}_{\mathcal{X}}(\mathcal{L})_{\downarrow \mathbb{E}_i} \subseteq \mathcal{C}_i\big)$ follows immediately from the observation that both sets whose union defines Y in Definition 14, exclude any configuration satisfying $\mathcal{X}$, or introducing failures of components that behave correctly in L. On the other hand, if $(L \odot B) \cap \mathcal{X} = \emptyset$ we have $L/\mathcal{X} = \{(c, \emptyset) \mid c \in L \odot B\}$ and $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}) = (L/\mathcal{X}) \rhd_{\mathcal{L},\mathcal{X},\mathbf{S}} \mathcal{B} = wf_{\mathcal{B}}(L \odot B) = L \odot B$.

$L \odot B$ is predecessor-closed with respect to $\mathcal{B}$ for any log that is obtained as the projection $\mathcal{M}_{\downarrow \mathbb{O}}$ of some rooted path $\mathcal{M}$ in $H_{\mathcal{B}}$.

5 Examples

Example 9 (Use of grafting). Consider two components with specifications $(\mathbb{E}_i, \mathcal{C}_i)$ where $\mathbb{E}_i = \{e_i, f_i\}$ and $\mathcal{C}_i = \big\{\emptyset, \{e_i\}\big\}$, $i = 1, 2$, the behavioral model $\mathcal{B} = 2^{\mathbb{E}}$

with $\mathbb{E} = \mathbb{E}_1 \cup \mathbb{E}_2$, the property $\mathcal{P} = \neg\big((f_1 \wedge f_2) \vee (f_1 \wedge e_2) \vee (e_1 \wedge f_2)\big)$ — that is, a failure event f_i becomes fatal once the other component produces some event —, and the log $L = (\mathbb{E}, \mathcal{L})$ with $\mathcal{L} = \big\{\emptyset, \{f_1\}, \{f_1, f_2\}\big\}$. In order to check whether $\mathcal{X}_1 = f_1$ is a necessary cause for the violation of $\mathcal{P}$ we compute $L \odot B = \mathcal{L}$, $L/\mathcal{X}_1 = \big\{(\emptyset, \emptyset), (\emptyset, \{f_1\}), (\{f_2\}, \{f_1\})\big\}$, and $\mathsf{CF}_{\mathcal{X}_1}(\mathcal{L}) = (L/\mathcal{X}_1) \rhd_{\mathcal{L}, \mathcal{X}_1, \mathbf{S}} \mathcal{B} = wf_{\mathcal{B}}\big(\{\emptyset, \{e_1\}, \{f_2\}, \{e_1, f_2\}\}\big) = \big\{\emptyset, \{e_1\}, \{f_2\}, \{e_1, f_2\}\big\}$. The obtained configuration structure still violates $\mathcal{P}$, hence $\mathcal{X}_1$ is not a necessary cause for the violation of $\mathcal{P}$. Intuitively, *even if the first component had behaved correctly*, $\mathcal{P}$ would have been violated. Simply taking the projection of $L/\mathcal{X}_1$ on the first configuration of the tuples we would have given the set of configurations $\big\{\emptyset, \{f_2\}\big\} \subseteq \mathcal{P}$.

Example 10 (Causal over-determination). Consider a system of two components with the same specifications and behavioral model as in Example 9, the property $\mathcal{P} = \neg(f_1 \vee f_2)$, and the log $L = (\mathbb{E}, \mathcal{L})$ with $\mathcal{L} = \big\{\emptyset, \{f_1\}, \{f_1, f_2\}\big\}$. In order to check whether $\mathcal{X}_1 = f_1$ is a necessary cause for the violation of $\mathcal{P}$ we compute $L/\mathcal{X}_1 = \big\{(\emptyset, \emptyset), (\emptyset, \{f_1\}), (\{f_2\}, \{f_1\})\big\}$ and $\mathsf{CF}_{\mathcal{X}_1}(\mathcal{L}) = (L/\mathcal{X}_1) \rhd_{\mathcal{L}, \mathcal{X}_1, \mathbf{S}} \mathcal{B} = wf_{\mathcal{B}}\big(\{\emptyset, \{e_1\}, \{f_2\}, \{e_1, f_2\}\}\big) = \big\{\emptyset, \{e_1\}, \{f_2\}, \{e_1, f_2\}\big\}$. This configuration structure still violates $\mathcal{P}$, hence $\mathcal{X}_1$ is not a necessary cause for the violation of $\mathcal{P}$. Symmetrically, $\mathcal{X}_2 = f_2$ is not a necessary cause either. On the other hand, as $\mathsf{CF}_{\mathcal{X}_1}(\mathcal{L})$ (resp. $\mathsf{CF}_{\mathcal{X}_2}(\mathcal{L})$) is inevitably faulty with respect to $\mathcal{P}$, the failures of the second (resp. first) component are found to be a sufficient cause for the violation of $\mathcal{P}$.

Example 11 (Joint causation). Consider the same component specifications, behavioral model, and log as in Example 10, and the property $\mathcal{P} = \neg(f_1 \wedge f_2)$. In order to check whether $\mathcal{X}_1 = f_1$ is a necessary cause for the violation of $\mathcal{P}$ we compute, as above, $\mathsf{CF}_{\mathcal{X}_1}(\mathcal{L})$ that satisfies our new property, hence $\mathcal{X}_1$ is a necessary cause for the violation of $\mathcal{P}$. As $\mathsf{CF}_{\mathcal{X}_1}(\mathcal{L})$ (resp. $\mathsf{CF}_{\mathcal{X}_2}(\mathcal{L})$) is not inevitably faulty with respect to $\mathcal{P}$, the failure of the second (resp. first) component alone is not a sufficient cause for the violation of $\mathcal{P}$.

Example 12 (Use of wf in grafting). Consider two components with specifications $(\mathbb{E}_i, \mathcal{C}_i)$ where $\mathbb{E}_1 = \{f_1, a\}$, $\mathbb{E}_2 = \{f_2\}$, and $\mathcal{C}_1 = \mathcal{C}_2 = \{\emptyset\}$, with observable events $\mathbb{O} = \{f_1, a, f_2\}$, the behavioral model $\mathcal{B} = \big\{\emptyset, \{f_1\}, \{f_1, a\}, \{f_2\}, \{f_1, f_2\}, \{f_1, a, f_2\}\big\}$, the property $\mathcal{P} = \neg(f_1 \wedge a)$, the log $L = (\mathbb{E}_1 \cup \mathbb{E}_2, \mathcal{L})$ with $\mathcal{L} = \big\{\emptyset, \{f_1\}, \{f_1, f_2\}, \{f_1, a, f_2\}\big\}$, and $\mathcal{X} = f_1 \wedge \neg a$: both components produce a fault event f_i; the conjunction of f_1 and a violates $\mathcal{P}$. We have $L/\mathcal{X} = \big\{(\emptyset, \emptyset), (\emptyset, \{f_1\}), (\{f_2\}, \{f_1\}), (\{f_1, a, f_2\}, \emptyset)\big\}$ and $(L/\mathcal{X}) \rhd_{\mathcal{L}, \mathcal{X}, \mathbf{S}} \mathcal{B} = wf_{\mathcal{B}}(\mathcal{C}) = \big\{\emptyset, \{f_2\}\big\} \subseteq \mathcal{P}$, where $\mathcal{C} = \big\{\emptyset, \{f_2\}, \{f_1, a, f_2\}\big\}$. Hence $\mathcal{X}$ is a necessary cause. The configuration $\{f_1, a, f_2\}$ is not reachable in $H_{\mathcal{B}}$ by any path passing only through the configurations in $\mathcal{C}$, therefore it is removed by $wf_{\mathcal{B}}$. Without applying $wf_{\mathcal{B}}$ we would have obtained the set of configurations $\mathcal{C}$ that still violates $\mathcal{P}$, thus $\mathcal{X}$ would not be found to be a necessary cause.

Example 13 (Comparison with [7]). Consider three components S (scheduler), C_1 and C_2 (clients) with the following event sets and specifications: $\mathbb{E}_S =$

$\{go_1, go_2\}$, $\mathcal{C}_S = \big\{\emptyset, \{go_1\}, \{go_2\}\big\}$, $\mathbb{E}_{C_1} = \{p_1, w_1\}$, $\mathcal{C}_{C_1} = \big\{\emptyset, \{p_1\}, \{w_1\}\big\}$ $\mathbb{E}_{C_2} = \{p_2, w_2, f_2\}$, $\mathcal{C}_{C_2} = \big\{\emptyset, \{p_2\}, \{w_2\}\big\}$, the behavioral model $\mathcal{B} = (p_1 \Rightarrow go_1) \wedge (p_2 \Rightarrow go_2)$, and the property $\mathcal{P} = \neg(p_1 \wedge p_2) \wedge \neg(w_1 \wedge f_2)$. Intuitively, the scheduler grants one of the components access to some critical section. Client C_i may enter with p_i if it has been granted access, or do w_i. The second component may fail with event f_2. The property requires mutual exclusion, and absence of f_2 in conjunction with w_1. We want to analyze causality of $\mathcal{X} = go_1 \wedge go_2$ on the log $L = (\mathbb{E}_S \cup \mathbb{E}_{C_1} \cup \mathbb{E}_{C_2}, \mathcal{L})$ with $\mathcal{L} = \big\{\emptyset, \{f_2\}, \{f_2, go_1\}, \{f_2, go_1, p_1\}, \{f_2, go_2\}, \{f_2, go_2, p_2\}, \{f_2, go_1, go_2, p_1\}, \{f_2, go_1, go_2, p_2\}, \{f_2, go_1, go_2, p_1, p_2\}\big\}$. We have $(L/\mathcal{X}) \rhd_{\mathcal{L},\mathcal{X},\mathbf{S}} \mathcal{B} = \big\{\emptyset, \{f_2\}, \{f_2, go_1\}, \{f_2, go_1, p_1\}, \{f_2, go_2\}, \{f_2, go_2, p_2\}\big\}$. Thus, $\mathcal{X}$ is a necessary cause for the violation of $\mathcal{P}$, and the failure f_2 of C_2 is not a sufficient cause.

The trace-based formalism of [7] cannot express the fact that the log does not distinguish any order among go_1 and go_2; if we fix this by introducing a new fault event go_{12}, then the *unaffected prefixes* — that is, the longest prefixes that could have been observed if go_{12} had not occurred — of the vector of component logs $(go_{12}; p_1; f_2.p_2)$ are $(\epsilon; \epsilon; f_2)$, and the set of counterfactual traces includes the vector of component traces $(go_1; w_1; f_2)$ that still violates $\mathcal{P}$. Hence, the fault of the scheduler is (incorrectly) not recognized as a necessary cause due to the fact that the information that the first client actually performed p_1, is lost.

In contrast, in the approach we present here, the use of configuration structures enables us to represent disjunctive counterfactual scenarios as in the example above that share different (sub-)configurations with the log that are incompatible among each other (here, $\{f_2, go_1, p_1\}$ and $\{f_2, go_2, p_2\}$).

Example 14 (Unobservable failure events). Consider the component specifications $(\mathbb{E}_i, \mathcal{C}_i)$ with $\mathbb{E}_1 = \{f_1, e_1\}$, $\mathbb{E}_2 = \{f_2\}$, $\mathbb{E}_3 = \{f_3, e_3\}$, and $\mathcal{C}_i = \{\emptyset\}$, $i = 1, ..., 3$, with observable events $\mathbb{O} = \{e_1, f_2, e_3\}$, the behavioral model $\mathcal{B} = (e_1 \Rightarrow f_1) \wedge (e_3 \Rightarrow f_3)$, the property $\mathcal{P} = \neg(e_1 \vee f_2)$, the log $L = (\mathbb{O}, \mathcal{L})$ with $\mathcal{L} = \big\{\emptyset, \{e_1\}, \{e_1, e_3\}\big\}$, and $\mathcal{X} = f_1$. Intuitively, the first and third component produce an unobservable violation of their specification; event e_1 following f_1 violates $\mathcal{P}$, whereas the second component behaves correctly. We have $(L/\mathcal{X}) \rhd_{\mathcal{L},\mathcal{X},\mathbf{S}} \mathcal{B} = \big\{\emptyset, \{f_3\}, \{f_3, e_3\}\big\} \subseteq \mathcal{P}$, thus $\mathcal{X}$ is correctly recognized as a necessary cause.

Example 15 (Running example). For Example 3, $L = (\mathbb{O}, \mathcal{L})$ with $\mathcal{L} = \big\{\emptyset, \{\mathtt{f1}\}, \{\mathtt{f1}, \mathtt{f}\}\big\}$, $\mathcal{X}_1 = \mathtt{f1}$, and the property $\mathcal{P} = \neg\mathtt{f}$ we obtain $L/\mathcal{X}_1 = \big\{(\emptyset, \emptyset), (\emptyset, \{\mathtt{f_1}\}), (\emptyset, \{f_1, r\}), (\{f_2\}, \{f_1, r\}), (\{f_2, f\}, \{f_1, r\}\big\}$ and $\mathsf{CF}_{\mathcal{X}_1}(\mathcal{L}) = (L/\mathcal{X}_1) \rhd_{\mathcal{L},\mathcal{X}_1,\mathbf{S}} \mathcal{B} = \big\{\emptyset, \{\mathtt{g}\}, \{\mathtt{g}, \mathtt{f2}\}, \{\mathtt{g}, \mathtt{f2}, \mathtt{f}\}\big\}$ which still violates $\mathcal{P}$, hence $\mathcal{X}_1$ is not a necessary cause for the violation of $\mathcal{P}$. On the other hand, for $\mathcal{X}_2 = \mathtt{f2}$ we have $\mathsf{CF}_{\mathcal{X}_2}(\mathcal{L}) = (L/\mathcal{X}_2) \rhd_{\mathcal{L},\mathcal{X}_1,\mathbf{S}} \mathcal{B} = \big\{\emptyset, \{\mathtt{f1}\}, \{\mathtt{f1}, \mathtt{r}\}, \{\mathtt{f1}, \mathtt{r}, \mathtt{s}\}\big\} \subseteq \mathcal{P}$, hence $\mathcal{X}_2$ is a necessary cause. Conversely, as $\mathsf{CF}_{\mathcal{X}_1}(\mathcal{L})$ is inevitably faulty whereas $\mathsf{CF}_{\mathcal{X}_2}(\mathcal{L})$ is not, the failure of the second component is a sufficient cause but not the failure of the first component.

6 Related work

As we remarked in the introduction, fault diagnosis is an active research field, with diverse questions and techniques drawn from different areas, including concurrency theory, discrete event systems, artificial intelligence, and control theory. We consider in this section only what we believe to be the most relevant works in these areas.

With respect to the techniques we use, our work is clearly related to works on diagnosis in discrete event systems [5, 21] and specifically diagnosis via unfolding [9]. The diagnosis questions in these works are actually very different from ours. They include *diagnosability* questions, which amount to determining the possible occurrence of (types of) hidden faults from the observation of executions, and *explanation* questions, which amount to determining which (prefix of) executions are compatible with observations recorded in a given log. Finding explanations is the key objective in the work by Haar et al. [3, 9]. In the terms of our framework, their goal is to find efficient algorithms (using Petri net unfolding techniques) for computing prefixes of $L \odot B$, where L records observed configurations, and B is the system specification. They also extended their techniques to finding explanations in systems with evolving topology [1], which we do not consider in this paper. To the best of our knowledge, these works do not consider fault ascription as we do here.

Closest to our approach on fault ascription are [20, 7], which also target fault ascription, and share a similar setting of black-box components equipped with specifications, and a log in the form of a vector of component traces. In contrast to the work presented here these works do not consider unobservable events, they are limited to linear component traces and, as pointed out in Example 13, they use a construction of the (sub-)configurations shared between the log and the counterfactuals that may result in either loss of information or inconsistencies in the counterfactual scenarios.

With their definition of *actual causality* based on a model of *structural equations* over a set of propositional variables [11], Halpern and Pearl have proposed the most influential definition of causality in computer science to date. Intuitively, the observed values of a set X of variables is an actual cause for an observed property φ if with different values of X, φ would not hold, and there exists a contingency in which the observed values of X entail φ. At first glance, it would seem that the notion of actual causality does not coincide with our notions of necessary and sufficient causality, but pinpointing the exact reasons for the difference, and characterizing the situations leading to different results, appears non-trivial, and we leave this as a question for further study.

Several approaches use [11] to encode and analyze execution traces. [2] determines potential causes for the first violation of an LTL formula by a trace. As [11] only considers a propositional setting without any temporal connectors, the trace is modeled as a matrix of propositional variables. The structure of the formula is used as a model to determine which events may have caused the violation of the property. The reported causes are, in general, neither necessary nor sufficient. [13] extends the definition of actual causality to totally ordered

sequences of events, and uses this definition to construct from a set of traces a probabilistic fault tree. The accuracy of the diagnostic depends on the number of traces used to construct the model.

The use of a distance metric is explored in [8] to localize, from a counter-example from model-checking, a possible fault as the difference between the error trace and a closest correct trace. This work features a "white box" approach that relies on access to source code, with no component specification.

7 Conclusion

We have presented in this paper a general framework for fault ascription, based on configuration structures. The framework supports the definition of analyses providing notions of necessary and sufficient causes for failures in component-based systems. Analyses in our framework relies on operators $\mathsf{CF}_{\mathcal{X}}$ for constructing counterfactual configurations, which we characterize abstractly via a simple constraint. The key contribution of this framework lies in the definition of notions of necessary and sufficient causality relative to an observed execution, recorded in a log, which we prove to be sound (each necessary or sufficient cause indeed explains an observed failure by some component failures) and complete (each failure has a necessary cause and a sufficient cause). We have also presented an instantiation of the framework that presents pruning and grafting constructions used to define a non-trivial counterfactual operator. Our framework generalizes previous works on fault ascription based on traces [20, 7], and we have shown by means of an example that our pruning and grafting constructions help solve the problem of inaccurate counterfactuals — leading to inconsistencies or loss of information — inherent in the trace-based approach.

Much work remains to be done however. For a start, we intend to formalize a symbolic algorithm implementing our definitions of fault ascription directly on Petri nets and synchronized products of transition systems, similar to the symbolic approach to fault ascription in real-time systems of [6] based on timed automata. For increased precision, we intend to leverage in our analysis techniques developed for fault diagnosis, especially those relying on unfolding [9]. Finally, following up the work of Baldan et al. on fault diagnosis in systems with evolving topology [1], we intend to extend our framework and causal analysis for fault ascription to dynamically configurable systems.

References

1. P. Baldan, T. Chatain, S. Haar, and B. König. Unfolding-based diagnosis of systems with an evolving topology. In *CONCUR 2008*, volume 5201 of *LNCS*. Springer, 2008.
2. I. Beer, S. Ben-David, H. Chockler, A. Orni, and R.J. Trefler. Explaining counterexamples using causality. *Formal Methods in System Design*, 40(1):20–40, 2012.
3. A. Benveniste, S. Haar, E. Fabre, and C. Jard. Distributed Monitoring of Concurrent and Asynchronous Systems. In *CONCUR 2003*, volume 2761 of *LNCS*. Springer, 2003.

4. A. Brennan. Necessary and sufficient conditions. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2012 edition, 2012.
5. C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer, 2nd edition, 2008.
6. G. Gössler and L. Astefanoaei. Blaming in component-based real-time systems. In *2014 International Conference on Embedded Software, EMSOFT*. IEEE, 2014.
7. G. Gössler and D. Le Métayer. A general framework for blaming in component-based systems. *Science of Computer Programming (in Press)*, 2015.
8. A. Groce, S. Chaki, D. Kroening, and O. Strichman. Error explanation with distance metrics. *STTT*, 8(3):229–247, 2006.
9. S. Haar and E. Fabre. Diagnosis with petri net unfoldings. In *Control of Discrete-Event Systems*, volume 433 of *LNCIS*, chapter 15. Springer, 2013.
10. J. Y. Halpern and C. Hitchcock. Graded causation and defaults. *CoRR*, abs/1309.1226, 2013.
11. J. Y. Halpern and J. Pearl. Causes and explanations: A structural approach. part i: Causes. *British Journal for the Philosophy of Science*, 56(4), 2005.
12. I. Hwang, S. Kim, Y. Kim, and C. E. Seah. A survey of fault detection, isolation and reconfiguration methods. *IEEE Trans. on Control Systems Technology*, 18(3), 2010.
13. M. Kuntz, F. Leitner-Fischer, and S. Leue. From probabilistic counterexamples via causality to fault trees. In *SAFECOMP*, volume 6894 of *LNCS*. Springer, 2011.
14. D. Lewis. *Counterfactuals*. Blackwell, 2nd edition, 2000.
15. D. Le Métayer and M. Maarek et al. Liability issues in software engineering: the use of formal methods to reduce legal uncertainties. *Commun. ACM*, 54(4), 2011.
16. J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2nd edition, 2009.
17. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1), 1987.
18. R. Stalnaker. A theory of conditionals. *Studies in Logical Theory*, pages 98–112, 1968.
19. R J. van Glabbeek and G. D. Plotkin. Configuration structures, event structures and petri nets. *Theoretical Computer Science*, 410(41), 2009.
20. S. Wang, A. Ayoub, R. Ivanov, O. Sokolsky, and I. Lee. Contract-based blame assignment by trace analysis. In *2nd ACM Int. Conf. HiCoNS*. ACM, 2013.
21. J. Zaytoon and S. Lafortune. Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37(2), 2013.