Causality Analysis and Fault Ascription in Component-Based Systems

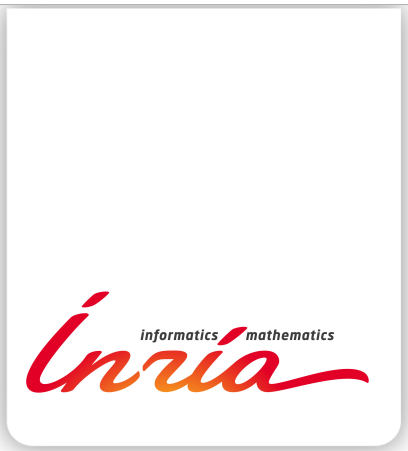
Gregor Gössler, Jean-Bernard Stefani

Causality Analysis and Fault Ascription in Component-Based Systems

Gregor Gössler, Jean-Bernard Stefani

RESEARCH REPORT
N° 9279
June 2019
Project-Team Spades

ISSN 0249-6399 ISRN INRIA/RR--9279--FR+ENG



Causality Analysis and Fault Ascription in Component-Based Systems*†

Gregor Gössler, Jean-Bernard Stefani

Project-Team Spades

Research Report n° 9279 — version 2 — initial version June 2019 — revised version August 2020 — 28 pages

Abstract: This article introduces a general framework for *fault ascription*, which consists in identifying, within a multi-component system, the components whose faulty behavior has caused the failure of said system. Our framework uses *configuration structures* as a general semantical model to handle truly concurrent executions, partial and distributed observations in a uniform way. As a first contribution, and in contrast with most of the current literature on counterfactual analysis which relies heavily on a set of toy examples, we first define a set of expected formal properties for *counterfactual builders*, i.e. operators that build counterfactual executions. We then show that causality analyses that satisfy our requirements meet a set of elementary soundness and completeness properties. Finally we present a concrete causality analysis meeting all our requirements, and we show that it behaves well under refinement. We present several examples illustrating various phenomena such as causal over-determination or observational determinism, and we discuss the relationship of our approach with Halpern and Pearl's actual causality analysis.

Key-words: causality, counterfactual analysis, formal requirements, components, hyperproperties

* This work has been partially supported by the French ANR project DCore (ANR-18-CE25-0007).
† Publisher version: https://doi.org/10.1016/j.tcs.2020.06.010

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Analyse causale et attribution de fautes dans les systèmes à composants

Résumé : Cet article introduit un cadre général pour l'*attribution de fautes* qui consiste à identifier, dans un système à composants, les composants dont le comportement incorrect a causé le dysfonctionnement du système. Nous définissons un ensemble de propriétés attendues de l'analyse contrefactuelle, et nous présentons une analyse raffinée qui satisfait ces besoins. Ceci contraste avec la pratique courante d'évaluer les définitions de causalité contrefactuelle *a posteriori* sur un ensemble d'exemples jouets. Nous établissons la monotonie de notre analyse sous différentes notions de raffinement.

Mots-clés : causalité, analyse contrefactuelle, besoins formels, composants, hyperpropriétés

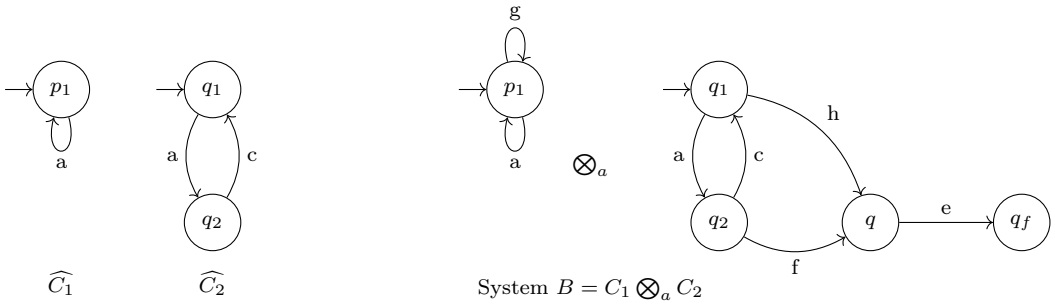


Figure 1: An example system specification: components C_1 and C_2, system B

1 Introduction

1.1 Motivation

Fault diagnosis and *fault isolation* [22, 35, 38] are important concerns for safety critical systems, as well as for large digital systems such as telecommunications networks, cloud computing and social networking platforms. In this paper, we are concerned with *fault ascription* which aims to identify, in presence of evidence of a failure of a multi-component system, the components whose faulty behavior have caused this failure. As defined, fault ascription can be seen as a precise form of fault diagnosis and fault isolation, where the actual cause and origin of a system failure is sought, instead of simple explanations in the form of possible behaviors conducive to the observed system failure.

Example 1. *Consider the mutlicomponent system B described on the right hand side of Figure 1. System B is formed by the product of two components C_1 and C_2 (synchronization product $\bigotimes_a$ requiring synchronization of components on event label a), whose correct specifications, $\widehat{C_1}$ and $\widehat{C_2}$ respectively, are shown on the left hand side of Figure 1.*

In system B, components C_1 and C_2 may exhibit faulty behaviors, manifested by g-labelled transitions for C_1, and f-labelled, h-labelled and e-labelled transitions for C_2. State q_f corresponds to a failure of system B: B's correct behavior is to repeatedly emit label c, driven by its internal clock C_1. Assume now that only events labelled a, c, f and e are visible and can be recorded in a log of B's execution. An occurrence of an e-labelled event signals the failure of B since it leads to the failure state q_f. Consider the following log L of a failed B execution: $L = \{\emptyset, \{a_1\}, \{a_1, c_1\}, \{a_1, c_1, a_2\}, \{a_1, c_1, a_2, f\}, \{a_1, c_1, a_2, f, e\}\}$. In L are recorded all the observable events of B's execution. In log L, event a_i corresponds to the ith occurrence of a synchronization event on label a, event c_i corresponds to the ith occurrence of a transition labelled c, and events f and e correspond to the occurrences of transitions labelled f and e.

Fault ascription asks the questions: what is the cause, or origin, of B's failure, manifested by the occurrence of event e? *and* what components are at the origin of B's failure? *Explanations, i.e. executions conducive to B's failure that accord with log L are clear: they are executions of the form*

$$(p_1, q_1) \xrightarrow{g^*} (p_1, q_1) \xrightarrow{a} (p_1, q_2) \xrightarrow{g^*} (p_1, q_2) \xrightarrow{c} (p_1, q_1) \xrightarrow{g^*} (p_1, q_1) \xrightarrow{a} (p_1, q_2) \xrightarrow{g^*} (p_1, q_2) \xrightarrow{f} (p_1, q) \xrightarrow{e} (p_1, q_f)$$

where $\xrightarrow{g^*}$ denotes the possible occurrence of zero or more g-labelled transitions. However, these possible explanations for B's failure do not reveal the fact that possible faults in component C_1, i.e. g-labelled transitions, play no role in B's failure. Indeed, reasoning counterfactually, it is easy to see that if C_2 never fails, then B never fails: it takes an f-labelled or an h-labelled transition of C_2 for B to fail, and that fault occurrences in C_1 have no incidence on the overall behavior of B (in particular, there are executions of B in which C_1 fails but B does not). The central question we consider in this paper is how to formalize such counterfactual reasoning.

Remark 1. *As should be clear from the above example, the notion of causality considered in this paper is finer than the usual notion of causal dependency in event structures [37]. In particular, an event e in the above example may causally depend on any number of events g_i, but no g_i would be a cause of e.*

1.2 Related Work

Diagnosis 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. The seminal work of [32] formalizes a theory of diagnosis for black-box components in first-order logic. A *diagnosis* for an observed incorrect behavior is essentially defined as a minimal set of components whose failure explains the observation. With respect to the techniques we use, our work is clearly related to works on model-based diagnosis in discrete event systems [6, 38] and specifically diagnosis via unfolding [17]. Our filtering operation is basically a reformulation of failure diagnosis [33] in the framework of configuration structures we use. In contrast, 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. [5, 17]. In the terms of our framework, their goal is to find efficient algorithms (using Petri net unfolding techniques) for computing prefixes of the filtered log. 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.

Counterfactual analysis and actual causality Existing work on defining and analyzing causality can roughly be divided into works based on counterfactual analysis, and works focusing on time series, such as [16]. As pointed out in [31], counterfactual reasoning is strictly more powerful than mere association, as done e.g. by analyzing time series. We will therefore focus on counterfactual analyses of causality.

Counterfactual causality analyses are a much researched topic, see e.g. the pioneering work of [34, 28, 2]. Some of the most influential contributions have been proposed by Pearl, Halpern and coworkers with their definitions of actual causality based on interventions on structural equations models (SEM) [2, 30, 20, 19]. We sketch a comparison between [19] and our work in Section 5.3. It has been pointed out in [11] that Halpern and Pearl's definitions of actual causality poorly support reasoning about state changes. Their inability to distinguish between states and events, and between presence and absence of an event, has been noted e.g. by Hopkins and Pearl [21]. In order for causality analysis to cope with system dynamics and temporal dependencies, [21] outlines the use of situation calculus [29] as an underlying model for counterfactual analysis. Counterfactuals are computed by enforcing or disabling events. However, no formal definition of causality is provided.

A more recent line of work [25, 27, 3] extends Halpern and Pearl's definition of actual causality with a logic capturing orders of events in order to enable causality checking on execution traces. An application of the approach is to construct probabilistic fault trees from a set of counterexample traces. Similarly, [8] propose a definition of actual causality inspired by [20] for actions in a program. [4] adapts actual causality to localize the causes for an execution trace to violate an LTL formula. Closer to our approach are [12, 13] which target the more specific case of fault ascription for black-box components equipped with specifications, with a log in the form of a vector of component traces. All events are considered observable.

However, all of these approaches to causality analysis suffer from two limitations [11, 14]. First, research on counterfactual causality analysis has been driven by a small set of simple examples, which serve as benchmarks for pitting proposed definitions of causality against human intuition. We call this approach TEGAR, *textbook example guided analysis refinement*. We do not find TEGAR, and its endless stream of refinements introduced for dealing with specific examples, very appealing. We think a sound approach for counterfactual causality analysis should first try to answer the question *what formal requirements should this analysis meet?*, before turning to the question *how to implement it?*. To the best of our knowledge there is little existing work proposing principled ways of defining counterfactual causality. [24] lists a set of requirements on definitions and analyses of causality that have been discussed in the literature. [9] specifies formal constraints on the behavior of preemption and overdetermination in SEM. In a similar spirit to our definition of counterfactuals, [26] defines the semantics of counterfactual analysis for traces of events of stochastic rule-based models. [23] proposes a general categorical formalization of interventions on a single variable.

Second, causality analysis should compose with application constraints such as partial observability, incremental analysis, and use of abstractions. The result of causality analysis is only meaningful if the model on which the analysis is performed, and the implementation generating the observations, satisfy a (refinement) relation required by the analysis. Theories of causation should be able to track causation across these levels of abstraction, and must therefore have a well-defined behavior under abstraction and refinement.

Our work opens an alternative approach to reasoning about causation. Rather than proposing definitions of causality whose properties remain implicit and are discussed a posteriori, our goal in this paper is to explicitly state a set of formal requirements *before* designing a counterfactual analysis satisfying them.

1.3 Contributions

In this paper, we aim to develop a formal framework for counterfactual analyses of component-based concurrent systems that avoids the pitfalls of the TEGAR approach. Specifically, using a general notion of component-based concurrent systems, we define formal requirements a counterfactual analysis of such systems should satisfy. Our framework uses configuration structures [36] as a general semantical model to handle truly concurrent executions, as well as partial and distributed observations, in a uniform way. We then present a specific counterfactual analysis that satisfies these requirements, and as an early study of the behaviour of this analysis under abstraction we establish its monotony under two notions of refinement.

In more detail, we make the following contributions:

1. We formalize a set of general properties a counterfactual function should satisfy in order to enable the construction of meaningful causality analyses. These requirements will serve as firm ground for guiding the definition of concrete analyses and reasoning about their properties. This approach contrasts with the usual way of defining counterfactual analyses whose behavior is validated *a posteriori* on a set of toy examples [11].

2. Our formal requirements include well-behavedness under varying observability (better observability improves precision of the analysis) and incrementality (adding new observations to a log improves precision). The first requirement is crucial for causality analyses to work on abstractions. The second requirement is essential to support incremental and on-the-fly analysis. We show that our instantiated counterfactual function is well-behaved under several notions of refinement. To the best of our knowledge, none of these properties, while being crucial for applying causality analysis to non-trivial problems in computer science, has been studied before in the context of causation.

3. We state many of our results for hyper-properties, i.e. properties that accrue not just to sets of executions, but to sets of sets of executions. Hyperproperties have been shown necessary to deal with properties such as security and quality of service [7], and taking them into account in our framework ensures our causality analysis can deal with violation of these kinds of properties as well. We discuss the use of analyzing the causation (or violation) of hyper-properties on motivating examples.

4. We propose a concrete well-formed counterfactual function and discuss our design choices on several examples.

The above contributions considerably extend our preliminary work on this topic reported in [15]. Further improvements with respect to this earlier work include an improved reconstruction of the non-observable behavior from a log, generalization to the causality analysis of arbitrary configurations that are not necessarily violations of component specifications, an improved concrete counterfactual function, and additional examples.

1.4 Outline

The remainder of the article is structured as follows. Section 2 introduces a set of notations and gathers basic definitions and constructions on configuration structures and hyperproperties. Section 3 presents our modeling framework, including our notions of systems, components, failures and logs. Section 4

introduces a set of general formal requirements a well-formed (or "usable") counterfactual function should satisfy, and discusses the properties of definitions of necessary and sufficient causality built on a well-formed counterfactual function. The section ends with a discussion of the special case of causality analysis for fault ascription. Section 5 proposes a concrete counterfactual function and discusses its properties such as well-formedness and behavior under several notions of refinement. We informally discuss how the proposed approach relates to Halpern and Pearl's actual causality. A set of examples illustrates our design choices of the counterfactual function. Section 6 concludes the paper.

2 Configurations structures and hyperproperties

We gather in this section notations, background definitions on configuration structures [36], and on properties and hyperproperties [7].

2.1 Notations

We use $\mathbb{N}$ to denote the set of naturals, i.e. positive integers, and $\mathbb{N}^*$ to denote $\mathbb{N} \setminus \{0\}$, i.e. the set of strictly positive integers. We use $[n]$ to denote the finite set of naturals $\{1, \ldots, n\}$ if $n > 0$. If $n = 0$, then $[n]$ denotes the empty set $\emptyset$. We use boldface to denote tuples of elements taken from a given set, as in $\mathbf{s} = \langle s_1, \ldots, s_n \rangle$ where all s_i belong to some set S. We use $\bigcup S$ to denote $\bigcup_{s \in S} s$. A predicate $\mathcal{P}$ that applies to elements of a set S is identified with a subset of 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. We denote by 2^S the powerset of set S, i.e. the set of subsets of set S. We use min and max to denote the set of minimal and maximal elements of a subset of an ordered set, respectively. Thus, when dealing with the powerset 2^S ordered by set inclusion, we have, for $A \subseteq 2^S$, $\mathsf{min}\, A = \{x \in A \mid \forall a \in A, a \subseteq x \implies a = x\}$, and $\mathsf{max}\, A = \{x \in A \mid \forall a \in A, x \subseteq a \implies a = x\}$.

2.2 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 configuration structure $(\mathbb{E}, \mathcal{C})$ is called* rooted *whenever $\emptyset \in \mathcal{C}$.*

Definition 2 (Step transition relation)**.** *The* step transition relation $\rightarrow_{\mathcal{C}}$ *associated with a configuration structure $C = (\mathbb{E}, \mathcal{C})$ is defined as follows:*

- *Synchronous interpretation: $c \rightarrow_{\mathcal{C}} d$ if $c \subseteq d$.*
- *Asynchronous interpretation: $c \rightarrow_{\mathcal{C}} d$ if $c \subseteq d$ and $\forall z \subseteq \mathbb{E}$, $c \subseteq z \subseteq d \implies z \in \mathcal{C}$.*

The atomic step transition relation $\mapsto_{\mathcal{C}}$ *associated with a configuration structure $C = (\mathbb{E}, \mathcal{C})$ is defined as follows: $c \mapsto_{\mathcal{C}} d$ if $c \rightarrow_{\mathcal{C}} d$ and $\forall c' \in \mathcal{C}, c \subseteq c' \subseteq d \implies c' = c \vee c' = d$.*

Remark 2. *When the configuration structure C is clear, we just write $c \rightarrow d$ and $c \mapsto d$ for $c \rightarrow_{\mathcal{C}} d$ and $c \mapsto_{\mathcal{C}} d$, respectively.*

Remark 3. *Most of the results of this paper do not depend on a particular interpretation. When they do depend on the synchronous or the asynchronous interpretation of the step transition, this dependency is explicitly stated.*

We now define some operations and predicates on configuration structures.

Definition 3 (Operations and predicates on configuration structures)**.** *Operations and predicates on configuration structures used in this paper are defined as follows:*

- *Composition: $(\mathbb{E}_1, \mathcal{C}_1) \parallel (\mathbb{E}_2, \mathcal{C}_2) = (\mathbb{E}, \mathcal{C})$ where $\mathbb{E} = \mathbb{E}_1 \cup \mathbb{E}_2$ and $\mathcal{C} = \{c \in 2^{\mathbb{E}} \mid c \cap \mathbb{E}_i \in \mathcal{C}_i, i = 1, 2\}$*
- *Intersection: $(\mathbb{E}_1, \mathcal{C}_1) \cap (\mathbb{E}_2, \mathcal{C}_2) = (\mathbb{E}_1 \cap \mathbb{E}_2, \mathcal{C}_1 \cap \mathcal{C}_2)$*
- *Inclusion: $(\mathbb{E}_1, \mathcal{C}_1) \subseteq (\mathbb{E}_2, \mathcal{C}_2) \iff \mathbb{E}_1 \subseteq \mathbb{E}_2 \wedge \mathcal{C}_1 \subseteq \mathcal{C}_2$*

- *Maximal elements:* $\mathsf{max}\,(\mathbb{E},\mathcal{C}) = \mathsf{max}\,\mathcal{C} = \{c \in \mathcal{C} \mid \forall c' \in \mathcal{C} : c \subseteq c' \Rightarrow c = c'\}$
- *Projection:* $(\mathbb{E},\mathcal{C})_{\downarrow\mathbb{F}} = (\mathbb{E} \cap \mathbb{F}, \mathcal{C}_{\downarrow\mathbb{F}})$ *where* $\mathcal{C}_{\downarrow\mathbb{F}} = \{c_{\downarrow\mathbb{F}} \mid c \in \mathcal{C}\}$, *and* $c_{\downarrow\mathbb{F}} = c \cap \mathbb{F}$. *For a set* S *of configuration structures on the same set of events, we define* $S_{\downarrow\mathbb{F}} = \{C_{\downarrow\mathbb{F}} \mid C \in S\}$.
- *Expansion: 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* $c^{\uparrow\mathbb{F}} = \{c' \subseteq \mathbb{F} \mid c' \cap \mathbb{E} = c\}$, $\mathcal{C}^{\uparrow\mathbb{F}} = \{c^{\uparrow\mathbb{F}} \mid c \in \mathcal{C}\}$, *and* $(\mathbb{E},\mathcal{C})^{\uparrow\mathbb{F}} = (\mathbb{F},\mathcal{C}^{\uparrow\mathbb{F}})$. *For a set* S *of configuration structures on the same set of events* $\mathbb{E}$*, we define* $S^{\uparrow\mathbb{F}} = \{C^{\uparrow\mathbb{F}} \mid C \in S\}$.
- *Comparison:* $(\mathbb{E}_1,\mathcal{C}_1) \leq (\mathbb{E}_2,\mathcal{C}_2) \iff \mathbb{E}_1 = \mathbb{E}_2 \wedge \forall c_2 \in \mathcal{C}_2\ \exists c_1 \in \mathcal{C}_1 : c_1 \subseteq c_2$*;*
  $\mathcal{C}_1 > \mathcal{C}_2 \iff \forall (c_1,c_2) \in \mathcal{C}_1 \times \mathcal{C}_2 : c_2 \subsetneq c_1$.

The above definitions of comparison will be used to define an exhaustiveness requirement in Section 4 and to state two results in Section 5, respectively.

Remark 4. *When* $\mathbb{E} \subseteq \mathbb{F}$*, we have by definition:* $\forall d \in c^{\uparrow\mathbb{F}}, d_{\downarrow\mathbb{E}} = c$.

2.3 Properties and hyperproperties of configuration structures

Definition 4 (Property and hyperproperty)**.** *A* property $\mathcal{P}$ *of configuration structures on events* $\mathbb{E}$ *is a set of subsets of* $\mathbb{E}$*, i.e.* $\mathcal{P} \subseteq 2^{\mathbb{E}}$*. A* hyperproperty $\mathbf{P}$ *of configuration structures on events* $\mathbb{E}$ *is a set of properties, i.e.* $\mathbf{P} \subseteq 2^{2^{\mathbb{E}}}$.

As an abuse of notation, when $\mathcal{P}$ is a property on events $\mathbb{E}$, we also denote $\mathcal{P}$ the configuration structure $(\mathbb{E},\mathcal{P})$. Likewise, given a hyperproperty $\mathbf{P}$ on events $\mathbb{E}$, we can consider it as a set of configuration structures $\{(\mathbb{E},\mathcal{P}) \mid \mathcal{P} \in \mathbf{P}\}$.

Definition 5 (Satisfaction)**.** *Given a set of events* $\mathbb{E}$*, a hyperproperty* $\mathbf{P}$ *(resp. property* $\mathcal{P}$*) is said to be satisfied by a configuration structure* $C = (\mathbb{E},\mathcal{C})$*, noted* $C \models \mathbf{P}$ *(resp.* $C \models \mathcal{P}$*), if* $\mathcal{C} \in \mathbf{P}$ *(resp.* $\mathcal{C} \subseteq \mathcal{P}$*).*

When adopting the asynchronous interpretation of the step transition relation induced by configuration structures, these notions of satisfaction of properties and hyperproperties of configuration structures coincide with the classical notions of trace-based properties and hyperproperties satisfaction, as defined in [7]. More precisely, we have the following proposition.

Definition 6 (Trace set of a configuration structure)**.** *The set of traces, or* trace set*,* $\mathsf{Tr}(\mathcal{C})$ *of a set of configurations is defined as the set* $\mathsf{Tr}(\mathcal{C}) = \mathsf{Tr}_{fin}(\mathcal{C}) \cup \mathsf{Tr}_{inf}(\mathcal{C})$*, where*

$$\mathsf{Tr}_{fin}(\mathcal{C}) = \{\sigma \in [n] \to \mathcal{C} \mid n \in \mathbb{N}^* \ \wedge\ \forall i \in [n-1],\, \sigma(i) \mapsto \sigma(i+1)\}$$
$$\mathsf{Tr}_{inf}(\mathcal{C}) = \{\sigma \in \mathbb{N}^* \to \mathcal{C} \mid \forall i \in \mathbb{N}^*,\, \sigma(i) \mapsto \sigma(i+1)\}$$

The trace set $\mathsf{Tr}(C)$ *of a configuration structure* $C = (\mathbb{E},\mathcal{C})$ *is defined to be the trace set of its configuration set, i.e.* $\mathsf{Tr}(C) \triangleq \mathsf{Tr}(\mathcal{C})$.

Definition 7 (Configuration structure of a trace)**.** *A (finite or infinite) trace* $\sigma \in \mathsf{Tr}(C)$ *over* $C = (\mathbb{E},\mathcal{C})$ *defines the configuration structure* $\mathtt{CS}(\sigma) = \big(\mathbb{E}, \mathtt{ran}(\sigma)\big)$ *where* $\mathtt{ran}(\sigma) = \{c \in \mathcal{C} \mid \exists i : \sigma(i) = c\}$.

Proposition 1. *For any configuration structure* $C = (\mathbb{E},\mathcal{C})$*, hyperproperty* $\mathbf{P}$ *on* $\mathbb{E}$*, and property* $\mathcal{P}$ *on* $\mathbb{E}$*, we have* $C \in \mathbf{P} \iff \mathsf{Tr}(C) \in \{\mathsf{Tr}(\mathcal{P}) \mid \mathcal{P} \in \mathbf{P}\}$*, and* $\mathsf{Tr}(C) \subseteq \mathsf{Tr}(\mathcal{P}) \implies C \subseteq \mathcal{P}$.

With the asynchronous interpretation of the step transition relation on configuration structures, for any property $\mathcal{P}$ *on* $\mathbb{E}$*,* $C \subseteq \mathcal{P} \iff \mathsf{Tr}(C) \subseteq \mathsf{Tr}(\mathcal{P})$.

Proof. We note first that any element c of a configuration structure $C = (\mathbb{E},\mathcal{C})$ occurs in some trace in $\mathsf{Tr}(C)$: in the extreme case that a configuration c has no predecessor in the atomic step relation, then the trace $\sigma : [1] \to \mathcal{C}$ such that $\sigma(1) = c$ is indeed an element of $\mathsf{Tr}(C)$. The first part of the proposition thus holds immediately for hyperproperties and properties.

For the second part of the proposition, assume $C = (\mathbb{E},\mathcal{C}) \subseteq \mathcal{P}$, where $\mathcal{P}$ is a property of configuration structures. Let $\sigma \in \mathsf{Tr}(C)$. By definition, for some index set I, for all $i \in I$, $\sigma(i) \in \mathcal{C}$, and thus $\sigma(i) \in \mathcal{P}$. Assume for the sake of contradiction that for some $i \in I$ we have $\neg(\sigma(i) \mapsto_{\mathcal{P}} \sigma(i+1))$. It would mean there exists $z \in \mathcal{P}$ such that $\sigma(i) \subset z \subset \sigma(i+1)$. However, since $\sigma(i) \to_{\mathcal{C}} \sigma(i+1)$, with the asynchronous interpretation we have $z \in \mathcal{C}$, which contradicts the fact that $\sigma(i) \mapsto_C \sigma(i+1)$. Hence, for all $i \in I$, $\sigma(i) \mapsto_{\mathcal{P}} \sigma(i+1)$ and thus $\sigma \in \mathsf{Tr}(\mathcal{P})$. The converse is clear, from the initial remark. □

The second part of Proposition 1 fails with the synchronous interpretation of the step transition. Consider the configuration structure C with configurations $\mathcal{C} = \{\emptyset, \{a\}, \{a, b, c\}\}$, and the property $\mathcal{P} = \{\emptyset, \{a\}, \{a, b\}, \{a, b, c\}\}$. We have $\mathcal{C} \subset \mathcal{P}$, but $\mathsf{Tr}(C)$ contains the trace $\sigma \triangleq \emptyset \mapsto_{\mathcal{C}} \{a\} \mapsto_{\mathcal{C}} \{a, b, c\}$, which is not present in $\mathsf{Tr}(\mathcal{P})$ for we do not have $\{a\} \mapsto_{\mathcal{P}} \{a, b, c\}$.

Remark 5. *A property $\mathcal{P}$ gives rise to an associated hyperproperty $[\mathcal{P}]$, defined as $[\mathcal{P}] \triangleq 2^{\mathcal{P}}$, and called its* lift hyperproperty*. Satisfying a property $\mathcal{P}$ is equivalent to satisfying its lift hyperproperty $[\mathcal{P}]$.*

The notions of safety properties and liveness properties extend to hyperproperties [7]. Of particular interest to us in this paper are safety hyperproperties, i.e. hyperproperties whose violations can be detected with finite observations and cannot be rectified by future events. We recall the definition of safety hyperproperty below.

Definition 8 (Witness)**.** *A* witness *is a finite configuration structure.*

Definition 9 (Safety Hyperproperty)**.** *A hyperproperty $\mathbf{P}$ is a* safety hyperproperty *if for any configuration structure C that violates $\mathbf{P}$, i.e. such that $C \not\models \mathbf{P}$, there exists a witness $M \subseteq C$ such that for any configuration structure C' with $M \subseteq C'$, we have $C' \not\models \mathbf{P}$.*

Remark 6. *Any safety hyperproperty $\mathbf{P}$ is* subset-closed*, i.e. for any configuration structures C and C', if $C \models \mathbf{P}$ and $C' \subseteq C$, then $C' \models \mathbf{P}$. By construction, the lift $[\mathcal{P}]$ of property $\mathcal{P}$ is a subset-closed hyperproperty (though not necessarily a safety hyperproperty).*

Because they are a source of useful examples, of particular interest to us in this paper will be k-safety hyperproperties.

Definition 10 (k-safety hyperproperty)**.** *A* k-safety hyperproperty *is a safety hyperproperty such that every violating configuration structure has a witness with at most k traces in its trace set.*

Remark 7. *The 1-safety hyperproperties are the lifted safety properties, i.e. $\mathbf{P}$ is a 1-safety hyperproperty iff $\mathbf{P} = [\mathcal{P}]$, where $\mathcal{P}$ is a safety property.*

3 Components, systems, failures and logs

In this section, we introduce our modeling framework, which relies on notions of component and system specifications, as well as notions of failures and logs.

3.1 Components and systems

A *component specification* is intended to capture the expected correct behavior of a component. A *system specification* is intended to capture the actual behavior of a system composed of a set of interacting components:

Definition 11 (Component and System specification)**.** *A* component specification *is a rooted configuration structure. 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 $\mathbb{B} = \bigcup_{i \in I} \mathbb{E}_i$, called a* system behaviour specification *or* behaviour specification *for brevity.*

We use the word "component" in a broad sense to denote part of a system behavior. The configuration structure B, as its name implies, is a specification of the system behaviour: it is used to express assumptions and constraints on the possible (correct and incorrect) behaviors of the composition of components that constitute the system. In particular, B can be used to model synchronization and coordination between components. The component specifications define the *correct* behavior of components, in the sense of *normality* of [18]. The actual component behavior in a system may violate those specifications: for instance, B may contain behaviors not in $S = \|_{i \in I} S_i$, for instance events in $\mathbb{E} \setminus \left(\bigcup_{i \in I} \bigcup \mathcal{C}_i \right)$. Conversely, part of the behaviors of S may not be feasible according to B.

Example 2. *An example system specification is illustrated in Figure 1. In this example, component specifications are given by the configuration structures* $(\mathbb{E}_1, \mathcal{C}_1)$ *and* $(\mathbb{E}_2, \mathcal{C}_2)$ *associated with automata* $\widehat{C_1}$ *and* $\widehat{C_2}$*, respectively:*

$$\mathbb{E}_1 = \{a_i \mid i \in \mathbb{N}^*\} \cup \{g_i \mid i \in \mathbb{N}^*\}$$
$$\mathcal{C}_1 = \{\emptyset\} \cup \{\{a_i \mid i \in [n]\} \mid n \in \mathbb{N}^*\}$$
$$\mathbb{E}_2 = \{a_i \mid i \in \mathbb{N}^*\} \cup \{c_i \mid i \in \mathbb{N}^*\} \cup \{f, h, e\}$$
$$\mathcal{C}_2 = \{\emptyset\} \cup \big\{\{a_i \mid i \in [n]\} \cup \{c_i \mid i \in [n]\} \mid n \in \mathbb{N}^*\big\} \cup \big\{\{a_i \mid i \in [n+1]\} \cup \{c_i \mid i \in [n]\} \mid n \in \mathbb{N}^*\big\}$$

In the above specifications, events a_i*,* g_i *and* c_i *denote the* i*th occurrence of an* a*-labelled transition, of a* g*-labelled transition, and of a* c*-labelled transition, respectively. Events* e*,* f *and* h *denote the occurrence of an* e*-labelled, of an* f*-labelled, and of an* h*-labelled transition, respectively.*

The system specification is given by the configuration structure $(\mathbb{E}, \mathcal{B})$ *associated with the product with synchronization on action* a *(noted* $\otimes_a$ *in Figure 1) of the two* C_1 *and* C_2 *automata (*$\widehat{C_1}$ *and* $\widehat{C_2}$ *automata extended with* g*-labelled, and* f*-labelled and* h*-labelled transitions, respectively):*

$$\mathbb{E} = \{a_i \mid i \in \mathbb{N}^*\} \cup \{g_i \mid i \in \mathbb{N}^*\} \cup \{c_i \mid i \in \mathbb{N}\} \cup \{f, h, e\} \tag{1}$$
$$\mathcal{B} = \big\{\{a_i \mid i \in [n]\} \cup \{c_i \mid i \in [n]\} \mid n \in \mathbb{N}^*\big\} \tag{2}$$
$$\cup \big\{\{a_i \mid i \in [n+1]\} \cup \{c_i \mid i \in [n]\} \mid n \in \mathbb{N}^*\big\} \tag{3}$$
$$\cup \big\{\{a_i \mid i \in [n]\} \cup \{c_i \mid i \in [n]\} \cup \{g_j \mid j \in [m]\} \mid n, m \in \mathbb{N}^*\big\} \tag{4}$$
$$\cup \big\{\{a_i \mid i \in [n+1]\} \cup \{c_i \mid i \in [n]\} \cup \{g_j \mid j \in [m]\} \mid n, m \in \mathbb{N}^*\big\} \tag{5}$$
$$\cup \big\{\{a_i \mid i \in [n]\} \cup \{c_i \mid i \in [n]\} \cup \{h\} \mid n \in \mathbb{N}^*\big\} \tag{6}$$
$$\cup \big\{\{a_i \mid i \in [n]\} \cup \{c_i \mid i \in [n]\} \cup \{h, e\} \mid n \in \mathbb{N}^*\big\} \tag{7}$$
$$\cup \big\{\{a_i \mid i \in [n+1]\} \cup \{c_i \mid i \in [n]\} \cup \{f\} \mid n \in \mathbb{N}^*\big\} \tag{8}$$
$$\cup \big\{\{a_i \mid i \in [n+1]\} \cup \{c_i \mid i \in [n]\} \cup \{f, e\} \mid n \in \mathbb{N}^*\big\} \tag{9}$$
$$\cup \big\{\{a_i \mid i \in [n]\} \cup \{c_i \mid i \in [n]\} \cup \{g_j \mid j \in [m]\} \cup \{h\} \mid n, m \in \mathbb{N}^*\big\} \tag{10}$$
$$\cup \big\{\{a_i \mid i \in [n]\} \cup \{c_i \mid i \in [n]\} \cup \{g_j \mid j \in [m]\} \cup \{h, e\} \mid n, m \in \mathbb{N}^*\big\} \tag{11}$$
$$\cup \big\{\{a_i \mid i \in [n+1]\} \cup \{c_i \mid i \in [n]\} \cup \{g_j \mid j \in [m]\} \cup \{f\} \mid n, m \in \mathbb{N}^*\big\} \tag{12}$$
$$\cup \big\{\{a_i \mid i \in [n+1]\} \cup \{c_i \mid i \in [n]\} \cup \{g_j \mid j \in [m]\} \cup \{f, e\} \mid n, m \in \mathbb{N}^*\big\} \tag{13}$$

Each line in the above definition of $\mathcal{B}$ *corresponds intuitively to a series of executions of system* B*. Lines (2) and (3) correspond to executions with no faults from* C_1 *or* C_2*. Lines (4) and (5) correspond to executions featuring only faults from* C_1 *(occurrences of* g*-labelled transitions). Lines (6) and (7) correspond to executions featuring a faulty transition* h *from* C_2*. Lines (8) and (9) correspond to executions featuring a faulty transition* f *from* C_2*. Lines (10) and (11) correspond to executions featuring faults from* C_1 *and a faulty transition* h *from* C_2*. Lines (12) and (13) correspond to executions featuring faults from* C_1 *and a faulty transition* f *from* C_2*.*

Figure 2 shows a subset of the configuration structure $(\mathbb{E}, \mathcal{B})$*.*

Our naming of events in the specification above is adequate for the asynchronous interpretation of the step transition relation. Dealing with the synchronous interpretation would have required a more complex naming scheme for events arising from terminal f*-labelled,* h*-labelled and* e*-labelled transitions of the* C_2 *component, or the use of the event naming scheme in [10]. For the sake of simplicity, we have preferred to opt for the ad-hoc naming scheme above, and we will use only the asynchronous interpretation of the step transition relation for this running example throughout the paper.*

A few comments on our model choices are in order.

Remark 8. *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* $S_i = (\mathbb{E}_i, \mathcal{C}_i)$

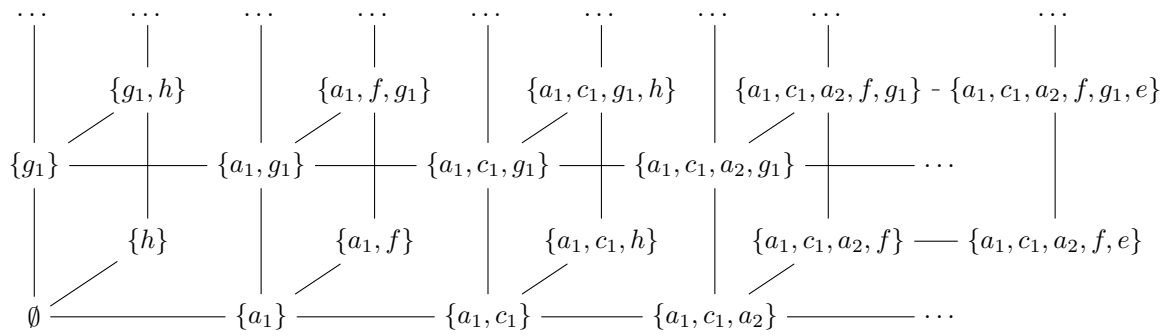


Figure 2: Subset of the configuration structure $(\mathbb{E}, \mathcal{B})$ of Example 2. Edges represent the atomic step transition relation.

- $B = (\mathbb{B}, \mathcal{B})$ *is a rooted configuration structure where* $\mathbb{B} = \mathbb{E} \cup \mathbb{E}^*$, $\mathbb{E} = \bigcup_{i \in I} \mathbb{E}_i$ *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 11: let* $\mathbf{S} = \langle S_1, \ldots, S_n \rangle$*, it suffices to define* $\mathtt{A}(\mathbf{S}, B) = (\mathbf{S}', B)$*, where* $\mathbf{S}' = \langle S_1, \ldots, S_n, \top_{E^*} \rangle$ *and* $\top_{E^*} = (\mathbb{E}^*, 2^{\mathbb{E}^*})$.

Remark 9. *Our definition of system specification places no constraint on the the relationship between the system behavioral model and the component specifications. This affords us a lot of freedom in formalizing system specifications and notably in specifying component composition and system failures. However, as will become apparent in Section 4, the intent is that events in a system specification reflect events from components executions, and that a system specification has some correlation with the composition of component specifications. Accordingly, a meaningful system specification* $(\mathbf{S}, B)$ *should satisfy* $B \cap \|_{i \in [n]} S_i \neq \emptyset$*, i.e. B should allow for some correct behavior of its components. The configuration structures in our running example 2 illustrate this: we have* $C_1 \parallel C_2 \subseteq B$*, although this is a simple situation where the system specification can be described as a product of modified composition specifications. Our implicit assumption on the relationship between system and component specifications relies on the simple form of configuration structure composition we have adopted, where possible synchronizations between components take the form of common events in their event sets. At the expense of additional complexity, one could have opted for instance for a composition based on a parallel composition of configuration structures based on the categorical product, restriction and relabelling as in [37]. This would have led to more complex conditions in Section 4, e.g. for characterizing consistent specifications for fault ascription, or in the definition of necessary and sufficient causes. We believe our present setting to be both simple and general enough to accommodate many forms of component composition and failure models, even if it is at the expense of some ingenuity in the naming of events (as illustrated in our running example with f and h events).*

3.2 Faults, failures, and logs

Given a system specification $(\mathbf{S}, B)$ with events in $\mathbb{E}$, a *fault* is an incorrect behavior with respect to $\mathbf{S}$. A *failure* with respect to a (hyper)property $\mathbf{P}$ over $\mathbb{E}$ is a violation of the latter. In the general case, $\mathbf{P}$ may be violated even though all components satisfy their component specifications. In Section 4.3 we will consider the special case where satisfaction of all component specifications implies satisfaction of $\mathbf{P}$. Conversely, the violation of a component specification does not necessarily entail a violation of $\mathbf{P}$. This is useful e.g. to model systems that tolerate certain component faults.

Remark 10. *As remarked above, a meaningful specification of a system should satisfy* $\|_{i \in [n]} S_i \cap B \neq \emptyset$*, but the analysis described below does not depend on this assumption.*

We start with a notion of consistency that generalizes the classical notion of absence of (binary) conflict in event structures [37].

Definition 12 (Consistency $\bigcirc$)**.** *Given sets* $\mathcal{C}$ *and* $\mathcal{B}$ *of configurations,* $\mathcal{C}$ *is* consistent *with respect to* $\mathcal{B}$*, written* $\bigcirc_{\mathcal{B}} \mathcal{C}$*, if* $\exists c \in \mathcal{B} : \bigcup \mathcal{C} \subseteq c$.

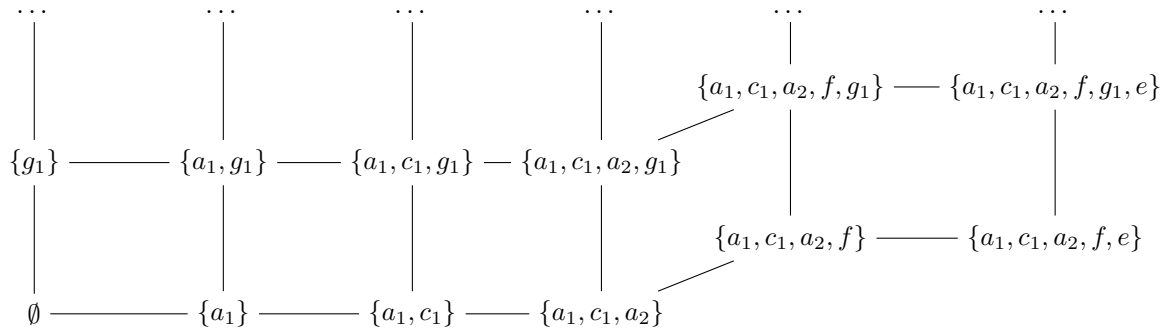


Figure 3: Subset of the filtered log $L \odot B$ of the running example.

Observations of the execution of a system take the form of *logs* and *log systems*.

Definition 13 (Log and log system)**.** *A* log *L of a system specified by $(\mathbf{S}, B)$ with $B = (\mathbb{B}, \mathcal{B})$, relative to* a set of observable events *$\mathbb{O}$, is a rooted configuration structure $(\mathbb{O}, \mathcal{L})$ that is* consistent with respect to *B, i.e. such that $\mathbb{O} \subseteq \mathbb{B}$, $(\mathbb{O}, \mathcal{L}) \subseteq B_{\downarrow \mathbb{O}}$, and $\bigcirc_{\mathcal{B}} \mathcal{L}$.*

A log system *is a set of logs.*

The use of consistency will become clear with the definition of filtering and the examples below. Intuitively, when $\mathcal{B}$ is a behaviour specification and $\mathcal{C}$ is a log, then consistency of $\mathcal{C}$ with respect to $\mathcal{B}$ means that the set of all observed events in the log $\mathcal{C}$ can be extended to some configuration in $\mathcal{B}$. In other terms, if $\mathcal{C}$ is consistent with respect to $\mathcal{B}$, then events in $\mathcal{C}$ are not mutually in conflict and can appear in some configuration of $\mathcal{B}$.

Definition 14 (Detected failure)**.** *A violation of a hyperproperty $\mathbf{P}$ is detected by a log system $\mathbf{L} = \{(\mathbb{O}, \mathcal{L}_i) \mid i \in I\}$ whenever $\bigcup_{i \in I} \mathcal{L}_i \not\models \mathbf{P}_{\downarrow \mathbb{O}}$.*

As is standard practice in fault diagnosis [6], we now introduce a notion of filtering, that retrieves possible explanations for the observed behaviour recorded in a log.

Definition 15 (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:*

$$L \odot B = \left\{ c \in \mathcal{L}^{\uparrow \mathbb{B}} \cap \mathcal{B} \mid \bigcirc_{\mathcal{B}} (\mathcal{L} \cup \{c\}) \right\}$$

The filtering operation extracts configurations from B that are consistent with observations provided by the log L.

Example 3. *For $B = (\mathbb{B}, \mathcal{B})$ with $\mathbb{B} = \{\tau, a, b\}$, $\mathcal{B} = \big\{\emptyset, \{\tau\}, \{a\}, \{a, b\}\big\}$, $\mathbb{O} = \{a\}$, and a log $L = (\mathbb{O}, \mathcal{L})$ with $\mathcal{L} = \big\{\emptyset, \{a\}\big\}$ we have $L \odot B = \big\{\emptyset, \{a\}, \{a, b\}\big\}$. The configuration $\{\tau\}$ is consistent with the observed configuration $\emptyset \in \mathcal{L}$ but inconsistent with the observation $\{a\}$ (since $\{\tau, a\} \notin \mathcal{B}$). Hence we do not have $\{\tau\}$ in $L \odot B$. The configurations $\{b\}$ and $\{\tau, b\}$ are also consistent with the observed configuration $\emptyset \in \mathcal{L}$ but they are not present in $\mathcal{B}$, hence we do not have them in $L \odot B$.*

Example 4 (Filtering on the running example)**.** *Applied to Example 2, where $L = (\mathbb{O}, \mathcal{L})$, with $\mathbb{O} = \{a_i, c_i \mid i \in \mathbb{N}\} \cup \{f, e\}$, $\mathcal{L} = \big\{\emptyset, \{a_1\}, \{a_1, c_1\}, \{a_1, c_1, a_2\}, \{a_1, c_1, a_2, f\}, \{a_1, c_1, a_2, f, e\}\big\}$, the configuration structure $L \odot B$ is shown in Figure 3. The consistency constraint of Definition 15 eliminates, in particular, all configurations of B that contain event h, since they are not consistent with the observed event f.*

Remark 11. *To simplify notations in the following sections, given a system specification and its behavioral model $(\mathbb{B}, \mathcal{B})$, we often write sets of configurations $\mathcal{X} \subseteq \mathcal{B}$ using logical formulas with events as propositional variables indicating the occurrence of these events. For instance, $\mathcal{X} = f$ stands for $\mathcal{X} = \{c \in \mathcal{B} \mid f \in c\}$.*

We gather below the definitions of auxiliary functions and predicates which will be useful in the definition of our causality analysis framework.

Definition 16 (Predecessor closure: $\mathtt{pc}$, $\mathtt{lpc}$)**.** *A set of configurations $\mathcal{C}$ is* predecessor-closed *with respect to $\mathcal{C}'$, written $\mathtt{pc}_{\mathcal{C}'}(\mathcal{C})$, if for any $c \in \mathcal{C}$, $c \in \min \mathcal{C}'$ or $\max\{c' \in \mathcal{C}' \mid c' \subsetneq c\} \cap \mathcal{C} \neq \emptyset$. We write $\mathtt{lpc}_{\mathcal{C}'}(\mathcal{C})$ to designate the largest predecessor-closed subset of $\mathcal{C}$ with respect to $\mathcal{C}'$.*

Hence, $\mathtt{pc}_{\mathcal{C}'}(\mathcal{C}) \iff \mathcal{C} = \mathtt{lpc}_{\mathcal{C}'}(\mathcal{C})$. Intuitively, $\mathcal{C}$ is predecessor-closed with respect to $\mathcal{C}'$ if each element of $\mathcal{C}$ has some immediate predecessor with respect to $\subseteq$ in $\mathcal{C}'$ that is in $\mathcal{C}$. Note that, by definition of the atomic step transition relation, if $c \in \mathcal{C}'$ we have $\max\{c' \in \mathcal{C}' \mid c' \subsetneq c\} = \{c' \in \mathcal{C} \mid c' \mapsto_{\mathcal{C}'} c\}$.

Example 5. *For $\mathcal{C} = \{\emptyset, \{a,b\}, \{d\}, \{c,d\}\}$ and $\mathcal{C}' = \{\emptyset, \{a\}, \{a,b\}, \{c\}, \{d\}, \{c,d\}\}$ we have $\mathtt{lpc}_{\mathcal{C}'}(\mathcal{C}) = \{\emptyset, \{d\}, \{c,d\}\}$.*

Definition 17 (Prefix inclusion $\sqsubseteq$)**.** *$C' = (\mathbb{E}', \mathcal{C}')$ refines $C = (\mathbb{E}, \mathcal{C})$ by* prefix inclusion*, noted $C' \sqsubseteq C$, if $C' \subseteq C$ and $\mathtt{pc}_{\mathcal{C}}(\mathcal{C}')$.*

Definition 18 (Immediate predecessors $\mathtt{pre}$)**.** $\mathtt{pre}_{\mathcal{C}}(\mathcal{S}) = \bigcup_{c \in \mathcal{S}} \max\{c' \in \mathcal{C} \setminus \mathcal{S} \mid c' \subseteq c\}$

The immediate predecessors of a configuration structure $\mathcal{S}$ thus correspond to the set of configurations not in $\mathcal{S}$ that can evolve directly into a configuration of $\mathcal{S}$. This informal explanation is backed up by the following proposition:

Proposition 2. $\mathtt{pre}_{\mathcal{C}}(\mathcal{S}) = \bigcup_{c \in \mathcal{S}} \{c' \in \mathcal{C} \setminus \mathcal{S} \mid c' \mapsto_{\mathcal{C}} c\}$

Proof. Given configuration structures $\mathcal{C}$, $\mathcal{S}$, and $c \in \mathcal{C}$, define $Z(c) = \max\{c' \in \mathcal{C} \setminus \mathcal{S} \mid c' \subseteq c\}$ and $Y(c) = \{c' \in \mathcal{C} \setminus \mathcal{S} \mid c' \mapsto_{\mathcal{C}} c\}$. We show that for all $c \in \mathcal{S}$, and $c' \in Z(c)$, there exists $d \in \mathcal{S}$ such that $c' \in Y(d)$. Let $c' \in Z(c)$. If $c' \mapsto_{\mathcal{C}} c$, then $c' \in Y(c)$ and we are done. If $c' \not\mapsto_{\mathcal{C}} c$, then there must be some $d \in \mathcal{C}$ such that $c' \mapsto_{\mathcal{C}} d \subseteq c$ and $d \neq c$. Now, we cannot have $d \in \mathcal{C} \setminus \mathcal{S}$ for $c' \in Z(c)$, hence $d \in \mathcal{S}$ and $c \in Y(d)$. This establishes that $\mathtt{pre}_{\mathcal{C}}(\mathcal{S}) \subseteq \bigcup_{c \in \mathcal{S}} Y(c)$. The converse is clear since for any $c' \in Y(c)$ we have $c' \mapsto c$, and thus, by definition of the step transition relation, $c' \in Z(c)$. □

Definition 19 (Invariance between configuration structures $\mathtt{inv}_{\mathbf{S}}$, $\mathtt{inv}^{+}_{\mathbf{S}}$)**.** *Let $\mathtt{inv}_{\mathbf{S}}(\mathcal{C}, \mathcal{C}') = \bigwedge_{i \in I} \left(\mathcal{C}_{\downarrow \mathbb{E}_i} \subseteq \mathcal{C}_i \implies \mathcal{C}'_{\downarrow \mathbb{E}_i} \subseteq \mathcal{C}_i\right)$ define* invariance *of $\mathbf{S} = ((\mathbb{E}_i, \mathcal{C}_i))_{i \in I}$ between $\mathcal{C}$ and $\mathcal{C}'$. Given a log $L = (\mathbb{O}, \mathcal{L})$, $\mathtt{inv}^{+}_{\mathbf{S}}(L, \mathcal{C}') \iff \exists \mathcal{C} \subseteq B : \mathcal{C}_{\downarrow \mathbb{O}} = \mathcal{L} \wedge \mathtt{inv}_{\mathbf{S}}(\mathcal{C}, \mathcal{C}')$ defines* possible invariance *of $\mathbf{S}$ between L and $\mathcal{C}'$.*

Intuitively, a set of component specifications is invariant between two configuration structures $\mathcal{C}$ and $\mathcal{C}'$ if the fact that all the components behave correctly in $\mathcal{C}$ is also true in $\mathcal{C}'$. A set of component specifications $\mathbf{S}$ is possibly invariant between a log L and a configuration structure $\mathcal{C}'$, if L is a projection of a configuration $\mathcal{C}$ such that $\mathbf{S}$ is invariant between $\mathcal{C}$ and $\mathcal{C}'$.

4 Counterfactual Configurations Analysis

In this section we define a notion of causality analysis of component behaviors for the violation of a system-level property. Our notion of causality analysis is dependent on a notion of *counterfactual builder* for constructing *counterfactuals*, which we require to obey certain key properties. A counterfactual is a configuration set that does not contain configurations from a given exclusion set, corresponding to possible executions of a system that avoid the configurations in the exclusion set.

A causality analysis takes as inputs the following elements:

- A system specification $\sigma = (\mathbf{S}, B)$, with component specifications $\mathbf{S} = ((\mathbb{E}_i, \mathcal{C}_i))_{i \in I}$ and a behaviour specification $B = (\mathbb{B}, \mathcal{B})$, with $\mathbb{B} = \bigcup_{i \in I} \mathbb{E}_i$.
- A set $\mathbb{O} \subseteq \mathbb{B}$ of observable events.
- A property $\mathcal{P} \subseteq 2^{\mathbb{B}}$ or hyperproperty $\mathbf{P} \subseteq 2^{2^{\mathbb{B}}}$.
- A log $L = (\mathbb{O}, \mathcal{L})$ or log system $\mathbf{L} = \{(\mathbb{O}_j, \mathcal{L}_j) \mid j \in J\}$.
- A set $\mathcal{X} \subseteq \mathcal{B}$ of non empty configurations, called an *exclusion set*.

The exclusion set $\mathcal{X}$ is a candidate configuration set to be checked for causality with respect to violation of $\mathcal{P}$, i.e. causality analysis checks whether $\mathcal{X}$ is a cause for the violation of $\mathcal{P}$.

As discussed above, the set of faulty configurations $(L \odot B) \setminus \|_i \mathcal{C}_i$ is, in general, incomparable with the violation of $\mathcal{P}$: the latter does not need to occur simultaneously with the violation of component specifications. Similarly, as illustrated in Example 1, the mere fact that the violation of $\mathcal{P}$ is preceded by the violation of a component specification $\mathcal{S}_k$, is not sufficient to establish that the latter is a *logical cause* of the former. We now turn to the formal definition of a causality analysis, and its attendant notion of causality.

In order to verify whether the configurations in $\mathcal{X}$ are a cause for the violation of $\mathcal{P}$ in L, a causality analysis constructs the (hypothetical) system behavior *where the configurations 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.* It then verifies whether all obtained behaviors satisfy $\mathcal{P}$.

Remark 12. *The property $\mathcal{P}$ may encompass both safety and liveness requirements. There are two reasons in favor of not restricting the analysis to safety. First, the "log" may represent an infinite counterexample trace, constructed for instance by a model-checker, that violates a liveness property, and we want to find out why. Second, even for finite logs, the counterfactuals may be infinite, such that liveness can be a concern.*

The main element of a causality analysis in our framework is an operation for building counterfactuals. Formally, a *counterfactual builder* is an operation on configuration sets, $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}} : 2^{2^{\mathbb{E}}} \to 2^{2^{\mathbb{E}}}$, that, given the configuration set of a log $L = (\mathbb{O}, \mathcal{L})$, in the context of a system specification σ, an exclusion set $\mathcal{X}$, and a set of observable events $\mathbb{O}$, builds a set of configurations $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L})$ that is "counterfactual with respect to $\mathcal{X}$ in σ, consistent with log L". Intuitively, the set of configurations $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L})$ models the system behavior "if $\mathcal{X}$ had not happened". We omit the subscripts σ and $\mathbb{O}$ in $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}$ when there is no risk of ambiguity. The action of counterfactual builder $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}$ is extended to log systems as follows:

$$\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\{\mathcal{L}_i \mid i \in I\}) \triangleq \bigcup_{i \in I} \mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L}_i)$$

Remark 13. *The above definition of counterfactual with respect to a set of logs may give the impression that spurious behaviors are introduced in the log system counterfactual when taking the union of log counterfactuals. Consider the following log configuration sets: $\mathcal{L}_1 = \{\emptyset, \{a\}\}$ and $\mathcal{L}_2 = \{\emptyset, \{b\}, \{b, a\}\}$. We have $a.b \in \mathsf{Tr}(\mathcal{L}_1 \cup \mathcal{L}_2)$ but $a.b \notin (\mathsf{Tr}(\mathcal{L}_1) \cup \mathsf{Tr}(\mathcal{L}_2))$, so are we not introducing the unobserved behavior $a.b$ in the counterfactual? In fact since $\{a\}$ is a configuration that occurs in an execution of the system ($\mathcal{L}_1$), then configurations $\{a\}$ and $\{b\}$ are concurrent and the trace $a.b$ is thus a valid trace in the configuration set of the system. The example just illustrates the fact that logs record only partial information on the execution of a system: here log $\mathcal{L}_2$ does not record the occurrence of 'a' prior to that of 'b', despite the fact that events 'a' and 'b' are concurrent.*

4.1 Requirements on Counterfactual Builders

We define in this section a set of formal properties we require of a counterfactual builder. We then define the notion of a well-formed counterfactual builder as a counterfactual builder meeting several of our requirements. A well-formed counterfactual builder is used as a basis for our notion of causality analysis in the following section. We begin by defining three requirements which provide lower and upper bounds for the set of counterfactuals computed by a counterfactual builder, given a log, a behaviour specification, and an exclusion set.

The first requirement is that counterfactual builders be *sound*, namely that they do produce counterfactuals, i.e. possible executions of a system that do not encounter the given exclusion set. This is the most basic requirement one can impose on counterfactual builders: an execution can be deemed counterfactual with respect to an exclusion set $\mathcal{X}$ only if it does not contain any of the configurations in the exclusion set. Formally:

Requirement 1 (Soundness Snd)**.** *A counterfactual builder $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}$ is* sound *if it satisfies the property* Snd *below, for any log $L = (\mathbb{O}, \mathcal{L})$, system specification $\sigma = (\mathbf{S}, B)$, $B = (\mathbb{B}, \mathcal{B})$, and exclusion set $\mathcal{X}$:*

$$\textsc{Snd} \triangleq \mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L}) \subseteq \mathcal{B} \setminus \mathcal{X}$$

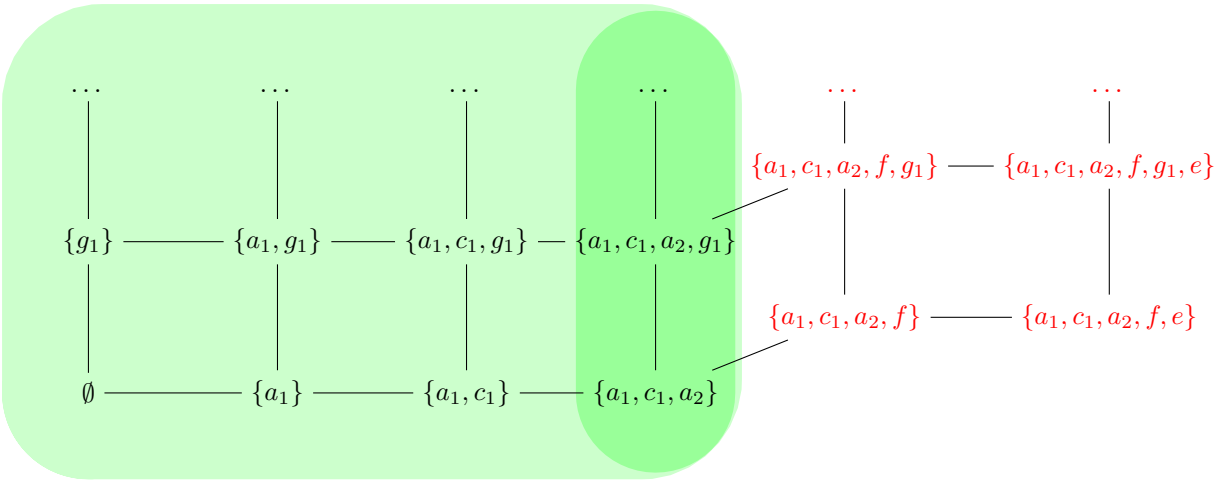


Figure 4: On the filtered log of the running example, the configurations containing event f (printed in red) are removed by counterfactual builders satisfying SND, for the exclusion set $\mathcal{X} = f \vee h$. Independence correctness CI requires the configurations highlighted in bright green to be included in counterfactuals built by the counterfactual builder. Dependence correctness CD requires that the configurations added by the counterfactual builder be bounded from above by the supersets of the configurations $\{a_1, c_1, a_2\}$, $\{a_1, c_1, a_2, g_1\}$, $\{a_1, c_1, a_2, g_1, g_2\}$, etc, which are highlighted in darker green.

The second requirement is that counterfactual builders be *independence correct*, namely that they keep, in the counterfactuals they construct, configurations in the filtered log that are independent (in the sense of true concurrency) from the configurations in the filtered log that touch the exclusion set. Again, this requirement seems intuitively clear: if we have a configuration c in the filtered log $L \odot B$ that does not contain a configuration c' that touches the exclusion set $\mathcal{X}$ (i.e. such that $c' \in \mathcal{X}$), then it must be part of a counterfactual execution. Formally:

Requirement 2 (Independence correctness CI)**.** *A counterfactual builder $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}$ is* independence correct *if it satisfies the property* CI *below, for any log $L = (\mathbb{O}, \mathcal{L})$, system specification $\sigma = (\mathbf{S}, B)$, $B = (\mathbb{B}, \mathcal{B})$, and exclusion set $\mathcal{X}$:*

$$\text{CI} \triangleq \{c \in L \odot B \mid \forall c' \in (L \odot B) \cap \mathcal{X} : \neg(c' \subseteq c)\} \subseteq \mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L})$$

The third requirement is that counterfactual builders be *dependence correct*, namely that they build counterfactuals from the immediate predecessors of the exclusion set in the log in order to preserve as much of the logged execution as possible. In other terms, dependence correctness ensures that configurations not in the filtered log $L \odot B$ can be added in a counterfactual only if they are greater than the last correct observation preceding some configuration in the exclusion set. Formally:

Requirement 3 (Dependence correctness CD)**.** *A counterfactual builder $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}$ is* dependence correct *if it satisfies the property* CD *below, for any log $L = (\mathbb{O}, \mathcal{L})$, system specification $\sigma = (\mathbf{S}, B)$, $B = (\mathbb{B}, \mathcal{B})$, and exclusion set $\mathcal{X}$:*

$$\text{CD} \triangleq \mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L}) \setminus (L \odot B) \subseteq \{c \in \mathcal{B} \mid \exists c' \in \mathtt{pre}_{L \odot B}(\mathcal{X}) : c' \subseteq c\}$$

Example 6 (SND, CI, and CD on the running example)**.** *The definitions of requirements* SND*,* CI*, and* CD *are illustrated in Figure 4 for the exclusion set $\mathcal{X} = f \vee h$. Independence correctness provides a lower bound for the set of computed counterfactuals, while soundness and dependence correctness provide upper bounds for the set of computed counterfactuals.*

The first three requirements above apply to any behaviour specification, irrespective of its relation with component specifications. The next requirement, called *correctness invariance*, insists that a counterfactual builder preserve in counterfactuals the correctness of components that had a correct behavior during the logged execution. One can see correctness invariance as a consistency requirement for fault ascription. Formally:

Requirement 4 (Correctness invariance CINV)**.** *A counterfactual builder* $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}$ *is* correctness invariant *if it satisfies the property* CINV *below, for any log* $L = (\mathbb{O}, \mathcal{L})$*, behaviour specification* $\sigma = ((\mathbb{E}_i, \mathcal{C}_i)_{i\in I}, B)$*,* $B = (\mathbb{B}, \mathcal{B})$*, and exclusion set* $\mathcal{X}$*:*

$$\textsc{Cinv} \triangleq \forall i \in I,\ \big((L \odot B)_{\downarrow \mathbb{E}_i} \subseteq \mathcal{C}_i \implies \mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L})_{\downarrow \mathbb{E}_i} \subseteq \mathcal{C}_i\big)$$

The last two requirements below are of a different nature from the previous ones: they are concerned with the behavior of causality analysis under variations in what is observed through a log. Requirement OBS asks for anti-monotony with respect to variation in the set of observable events: more observable events entail less uncertainty and thus smaller counterfactuals. Formally:

Requirement 5 (Observation anti-monotony)**.** *A counterfactual builder* $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}$ *is* observation anti-monotonic *if it satisfies the property* OBS *below, for any log* $L = (\mathbb{O}, \mathcal{L})$*, behaviour specification* $\sigma = ((\mathbb{E}_i, \mathcal{C}_i)_{i\in I}, B)$*,* $B = (\mathbb{B}, \mathcal{B})$*, and exclusion set* $\mathcal{X}$*:*

$$\textsc{Obs} \triangleq \big(\mathcal{X} \subseteq L \odot B \wedge \mathtt{pc}_{B_{\downarrow\mathbb{O}}}(L) \wedge \forall \mathbb{O}' \subseteq \mathbb{O}, \mathcal{X}_{\downarrow\mathbb{O}'}^{\uparrow\mathbb{B}} = \mathcal{X}\big) \implies \mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L}) \subseteq \mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}'}(\mathcal{L}_{\downarrow\mathbb{O}'})$$

Requirement INC asks for *incrementality* of the analysis: a log with more observations leads to a more precise diagnostic. Formally:

Requirement 6 (Incrementality INC)**.** *A counterfactual builder* $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}$ *is* incremental *if it satisfies the property* INC *below, for any log* $L = (\mathbb{O}, \mathcal{L})$*, behaviour specification* $\sigma = ((\mathbb{E}_i, \mathcal{C}_i)_{i\in I}, B)$*,* $B = (\mathbb{B}, \mathcal{B})$*, and exclusion set* $\mathcal{X}$*:*

$$\begin{aligned}\textsc{Inc} \triangleq \forall\ \textit{log}\ L',\ L \sqsubseteq L' \wedge L \odot B \subseteq L' \odot B \wedge \mathit{inv}_{\mathbf{S}}(L \odot B, L' \odot B) \implies \\ \mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L}') \setminus (L' \odot B) \subseteq \mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L}) \setminus (L \odot B)\end{aligned}$$

Definition 20 (Well-formed counterfactual builders)**.** *A counterfactual builder* $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}$ *— where* $\sigma = (\mathbf{S}, B)$ *is a system specification with* $B = (\mathbb{B}, \mathcal{B})$ *and* $\mathcal{X} \subseteq \mathcal{B}$ *is an exclusion set with* $\emptyset \notin \mathcal{X}$ *— is well-formed if for any log* $L = (\mathbb{O}, \mathcal{L})$ *it satisfies the property* WF *below*

$$\textsc{Wf} \triangleq \textsc{Snd} \wedge \textsc{Ci} \wedge \textsc{Cd} \wedge \textsc{Cinv} \wedge \textsc{Obs} \wedge \textsc{Inc}$$

Proposition 3. *Let* $L = (\mathbb{O}, \mathcal{L})$*. If* $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}$ *satisfies Requirements* CI *and* CD *and* $(L \odot B) \cap \mathcal{X} = \emptyset$ *then* $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L}) = L \odot B$*.*

Proof. If $(L \odot B) \cap \mathcal{X} = \emptyset$, then $\mathtt{pre}_{L\odot B}(\mathcal{X}) = \emptyset$. From CI we get $L \odot B \subseteq \mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L})$, and from CD we get $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L}) \setminus (L \odot B) \subseteq \emptyset$, hence $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L}) \subseteq (L \odot B)$. □

The following property follows directly from the observation that by Proposition 1, the behavioral models, traces, and specifications on which both counterfactual sets are computed are identical.

Proposition 4 (Stability of CF under trace equivalence)**.** *If* $\mathsf{Tr}(B) = \mathsf{Tr}(B')$*,* $\mathsf{Tr}(\mathcal{L}) = \mathsf{Tr}(\mathcal{L}')$*, and* $\forall i \in [n] : \mathsf{Tr}(S_i) = \mathsf{Tr}(S_i')$ *under the asynchronous interpretation, then* $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L}) = \mathsf{CF}_{\sigma',\mathcal{X},\mathbb{O}}(\mathcal{L}')$ *where* $\sigma = (\mathbf{S}, B)$ *and* $\sigma' = (\mathbf{S}', B')$*.*

Finally we introduce a last requirement that is not part of the requirements for well-formed counterfactual builders, but that is used to avoid well-formed but degenerate counterfactual builders by ensuring the presence of enough configurations. For instance, the simple counterfactual builder $\mathsf{CF}^0_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L}) \triangleq \mathtt{lpc}_{L\odot B}\big((L \odot B) \setminus \mathcal{X}\big)$ can be shown to be well-formed, but it does not contain any configurations that are not already present in the filtered log. The *exhaustiveness* requirement ensures counterfactuals are not limited to this simple case. A counterfactual builder is *exhaustive* if the counterfactuals it builds include all configurations not in the exclusion set, provided they meet three conditions: (1) they are greater than the observed predecessors of the faulty configurations in the expanded log, (2) they are consistent with, and invariant with respect to, the set R (the simple counterfactual builder above) defined as the largest predecessor-closed subset of the expanded log from which $\mathcal{X}$ has been removed, and (3) they are, together with R, predecessor-closed in the behavioral model. Intuitively, counterfactual executions expand on the non-faulty prefixes of faulty configurations that appear in the log, provided their faulty or non-faulty status be the same as in these prefixes. Note that components that have violated their specifications in R may still do so in $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L})$. Formally:

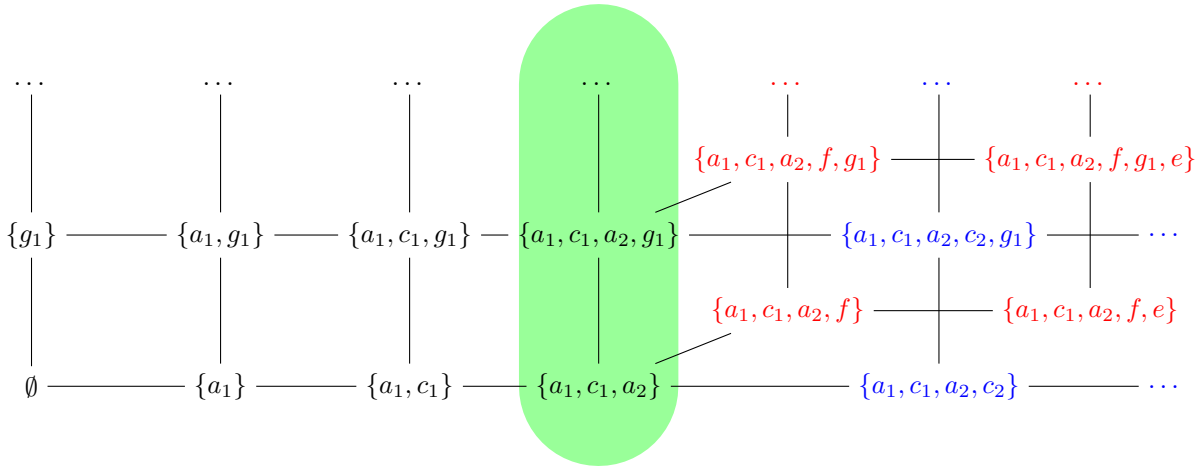


Figure 5: On the filtered log of the running example, requirement Exh ensures the configurations $\{a_1, c_1, a_2, c_2\}$, $\{a_1, c_1, a_2, c_2, g_1\}$, etc (printed in blue) are present in the counterfactual configuration structures for the exclusion set $\mathcal{X} = f \vee h$. The meaning of the green and red colors is as in Figure 4.

Requirement 7 (Exhaustiveness Exh)**.** *A counterfactual builder $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}$ is* exhaustive *if it satisfies the property* Exh *below, for any log $L = (\mathbb{O}, \mathcal{L})$, system specification $\sigma = (\mathbf{S}, B)$, $B = (\mathbb{B}, \mathcal{B})$, and exclusion set $\mathcal{X}$:*

$$\textsc{Exh} \triangleq \forall \mathcal{C}' \subseteq \mathcal{B} \setminus \mathcal{X},\ \mathtt{pre}_{L \odot B}\big((L \odot B) \cap \mathcal{X}\big) \leq \mathcal{C}' \wedge \bigcirc_{\mathcal{B}}(R \cup \mathcal{C}') \wedge \mathtt{inv}_{\mathbf{S}}(R, \mathcal{C}') \wedge \mathtt{pc}_{\mathcal{B}}(\mathcal{C}' \cup R) \implies \mathcal{C}' \subseteq \mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L})$$

where: $R \triangleq \mathtt{lpc}_{L \odot B}((L \odot B) \setminus \mathcal{X})$

Example 7 (Exh on the running example)**.** *Figure 5 illustrates the purpose of requirement* Exh *on the running example for the exclusion set $\mathcal{X} = f \vee h$.*

4.2 Causality Analysis

We now proceed to define a causality analysis based on counterfactual builders. For a given $\mathsf{CF}_{\mathcal{X}}$ we define the notions of necessary and sufficient causality. Intuitively, a necessary cause is some configuration set that must be present whenever a property violation is detected. A sufficient cause is some configuration set that inevitably leads to the violation of a property. In this section, we consider only well-formed counterfactual builders. We consider first the notion of necessary causality.

Definition 21 (Necessary causality)**.** *Let $(\mathbf{S}, B)$ be a system with component specifications $\mathbf{S} = \langle S_1, ..., S_n \rangle$ and $S_i = (\mathbb{E}_i, \mathcal{C}_i)$, $\mathbf{P}$ be a hyperproperty, $\mathbf{L} = \{(\mathbb{O}, \mathcal{L}_j) \mid j \in J\}$ be a log system in which a violation of $\mathbf{P}$ is detected, and $\mathcal{X} \subseteq \mathcal{B}$ be an exclusion set. $\mathcal{X}$ is a* necessary cause *of the violation of $\mathcal{P}$ in $\mathbf{L}$ if $\mathsf{CF}_{\mathcal{X}}(\{\mathcal{L}_j \mid j \in J\}) \models \mathbf{P}$.*

The faults of a subset $\mathcal{I}$ of components are a necessary cause of the violation of $\mathbf{P}$ in $\mathbf{L}$ if $\mathcal{X} \triangleq \{c \in \mathcal{B} \mid \exists i \in \mathcal{I} : c_{\downarrow \mathbb{E}_i} \notin \mathcal{C}_i\}$ is a necessary cause of the violation of $\mathbf{P}$ in $\mathbf{L}$.

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

Remark 14. *$\mathcal{X}$ is not required to be minimal in order to qualify as a cause. This is useful to reason about* group causality*, for instance, to determine the liability of component providers that are responsible for more than one component. In this example one could take $\mathcal{X}$ as the set of faulty configurations of all components of a system that have been developed by the same provider.*

Remark 15. *To apply the definition of necessary causality to a property $\mathcal{P}$ and single log $L = (\mathbb{O}, \mathcal{L})$, one needs only apply it to the lift hyperproperty $[\mathcal{P}]$. By definition, we have that $\mathcal{X}$ is a necessary cause of the violation of $[\mathcal{P}]$ in $\mathbf{L} = \{L\}$ if $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}) \in [\mathcal{P}] \iff \mathsf{CF}_{\mathcal{X}}(\mathcal{L}) \subseteq \mathcal{P} \iff \mathsf{CF}_{\mathcal{X}}(\mathcal{L}) \models \mathcal{P}$.*

We now prove that this notion of necessary causality is sound, in the sense that any necessary cause contains some configuration appearing in the log or log system. We first prove it for properties, and then for hyperproperties.

Proposition 5 (Soundness of necessary causality for properties)**.** *If $\mathcal{X}$ is a necessary cause for the violation of property $\mathcal{P}$ in the log $L = (\mathbb{O}, \mathcal{L})$, and the counterfactual builder satisfies Requirements* CI *and* CD*, then $(L \odot B) \cap \mathcal{X} \neq \emptyset$.*

Proof. By contradiction. Let $\mathcal{X}$ be such that $(L \odot B) \cap \mathcal{X} = \emptyset$. Because property $\mathcal{P}$ is violated in log L, we have $\mathcal{L} \not\models \mathcal{P}_{\downarrow\mathbb{O}}$. By Proposition 3 we have $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}) = L \odot B$. Hence $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}) \not\models \mathcal{P}$, and $\mathcal{X}$ is not a cause for the violation of $\mathcal{P}$ in L. □

Proposition 6 (Soundness of necessary causality for hyperproperties)**.** *If $\mathcal{X}$ is a necessary cause for the violation of property $\mathbf{P}$ in the log system $\mathbf{L} = \{L_j = (\mathbb{O}, \mathcal{L}_j) \mid j \in J\}$, and the counterfactual builder satisfies Requirements* CI *and* CD*, then for some $j \in J$, $(L_j \odot B) \cap \mathcal{X} \neq \emptyset$.*

Proof. By definition of necessary causality, we have $\mathsf{CF}_{\mathcal{X}}\big(\bigcup_{j\in J} \mathcal{L}_j\big) \in \mathbf{P}$. Hence there exists a property $\mathcal{P} \in \mathbf{P}$ such that $\mathsf{CF}_{\mathcal{X}}\big(\bigcup_{j\in J} \mathcal{L}_j\big) = \mathcal{P}$. Thus we have, for all $j \in J$, $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}_j) \subset \mathcal{P}$. By definition of violation in a log system, we have $\bigcup_{j\in J} \mathcal{L}_j \notin \mathbf{P}_{\downarrow\mathbb{O}}$. Hence there must exist some $k \in J$ such that $\mathcal{L}_k \not\subseteq \mathcal{P}_{\downarrow\mathbb{O}}$, which means that $\mathcal{P}$ is violated in L_k. Since $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}_k) \subset \mathcal{P}$, this means $\mathcal{X}$ is a necessary cause of the violation of $\mathcal{P}$ in L_k. Hence by Proposition 5, we have $(L_k \odot B) \cap \mathcal{X} \neq \emptyset$. □

Remark 16. *Our definition of necessary causality applies to arbitrary hyperproperties, and the above result on the soundness of necessary causality holds for arbitrary hyperproperties. However, as we will see below, other results such as monotonicity with respect to observability and completeness of necessary causality hold a priori only for subset-closed hyperproperties.*

Proposition 7. *Verifying necessary causality of configuration set $\mathcal{X}$ with respect to a property $\mathcal{P}$ and a log $L = (\mathbb{O}, \mathcal{L})$ amounts to verifying a safety property on the counterfactual configuration set $\mathsf{CF}_{\mathcal{X}}(\mathcal{L})$.*

Proof. The test of Definition 21 amounts to verifying whether $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}) \subseteq \mathcal{P}$. Any configuration $c \in \mathsf{CF}_{\mathcal{X}}(\mathcal{L}) \setminus \mathcal{P}$ is a finite witness for the violation of $\mathcal{P}$ by $\mathsf{CF}_{\mathcal{X}}(\mathcal{L})$; any violation of $\mathcal{P}$ by $\mathsf{CF}_{\mathcal{X}}(\mathcal{L})$ has such a finite witness. The claim follows. □

Proposition 8 (Necessary causality is monotonic wrt. observability for subset-closed hyperproperties)**.** *For any behavioral model $(\mathbb{B}, \mathcal{B})$, log set $\mathbf{L} = \{(\mathbb{O}, \mathcal{L}^j) \mid j \in J\}$ with $\forall j \in J : \mathtt{pc}_{B_{\downarrow\mathbb{O}}}(L)$, $\mathbb{O}' \subseteq \mathbb{O}$, and $\mathcal{X} \subseteq L \odot B$ with ${\mathcal{X}_{\downarrow\mathbb{O}'}}^{\uparrow\mathbb{B}} = \mathcal{X}$, and counterfactual builder $\mathsf{CF}_{\mathcal{X}}$ that satisfies Requirement* OBS*, if $\mathcal{X}$ is a necessary cause for the violation of subset-closed hyperproperty $\mathbf{P}$ in $\mathbf{L}' = \{(\mathbb{O}', \mathcal{L}^j_{\downarrow\mathbb{O}'}) \mid j \in J\}$ then it is a necessary cause for the violation of $\mathbf{P}$ in $\mathbf{L}$.*

Proof. Let $\mathcal{L} = \{\mathcal{L}^j \mid j \in J\}$ and $\mathcal{L}' = \{\mathcal{L}^j_{\downarrow\mathbb{O}'} \mid j \in J\}$. Note first that if a violation of $\mathbf{P}$ is detected in L' ($\bigcup \mathcal{L}' \not\models \mathbf{P}_{\downarrow\mathbb{O}'}$), then it is also detected in L since $\mathbb{O}' \subseteq \mathbb{O}$. Since $\mathcal{X}$ is a necessary cause for the violation of $\mathbf{P}$ in $\mathbf{L}'$, then $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}') \models \mathbf{P}$. But then because $\mathsf{CF}_{\mathcal{X}}$ verifies OBS we have, for all $j \in J$, $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}^j) \subseteq \mathsf{CF}_{\mathcal{X}}(\mathcal{L}^j_{\downarrow\mathbb{O}'})$. Hence $\mathsf{CF}_{\mathcal{X}}(\mathbf{L}) = \bigcup_{i\in J} \mathsf{CF}_{\mathcal{X}}(\mathcal{L}^j) \subseteq \bigcup_{i\in J} \mathsf{CF}_{\mathcal{X}}(\mathcal{L}^j_{\downarrow\mathbb{O}'}) = \mathsf{CF}_{\mathcal{X}}(\mathbf{L}')$, and thus $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}) \models \mathbf{P}$ for $\mathbf{P}$ is subset-closed. □

We now turn to the definition of sufficient causality. Intuitively, sufficient causality ought to be a kind of dual of necessary causality. Unfortunately, we do not have a formal dual for the notion of necessary causality. Instead, we settle for a notion of sufficient causality appropriate for properties only.

Definition 22 (Inevitable)**.** *The violation of a property $\mathcal{P}$ in a configuration structure $\mathcal{C}$ is* inevitable *if*

$$\forall tr \in \mathsf{max}\,\mathsf{Tr}(\mathcal{C}) : \mathtt{CS}(tr) \not\models \mathcal{P}$$

Definition 23 (Sufficient cause)**.** *Consider a system $(\mathbf{S}, B)$ with $\mathbf{S} = \langle S_1, ..., S_n\rangle$ and $S_i = (\mathbb{E}_i, \mathcal{C}_i)$, a property $\mathcal{P}$, a log $L = (\mathbb{O}, \mathcal{L})$ in which a violation of $\mathcal{P}$ is detected, and $\mathcal{Y} \subseteq \mathcal{B}$. $\mathcal{Y}$ is a sufficient cause for the violation of $\mathcal{P}$ in L if a violation of $\mathcal{P}$ in $\mathsf{CF}_{\mathcal{B}\setminus\mathcal{Y}}(\mathcal{L})$ is inevitable.*

The faults of a subset $\mathcal{I}$ of components are a sufficient cause for the violation of $\mathcal{P}$ in L, if $\mathcal{Y} \triangleq \{c \in \mathcal{B} \mid \forall i \in [n] \setminus \mathcal{I} : c_{\downarrow\mathbb{E}_i} \in \mathcal{C}_i\}$ is a sufficient cause for the violation of $\mathcal{P}$ in L.

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

The following results show that our causality analysis does not blame any set of innocent components, and that it finds a necessary and a sufficient cause for every system-level failure.

Proposition 9 (Completeness of sufficient causality)**.** *If the counterfactual builder satisfies Requirements* Ci *and* Cd *then each inevitable violation of $\mathcal{P}$ in $L \odot B$ has a sufficient cause.*

Proof. 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 $\mathcal{Y} = \mathcal{B}$ and $\mathcal{X} = \mathcal{B} \setminus \mathcal{Y} = \emptyset$. Because $\mathsf{CF}_{\mathcal{X}}$ satisfies Requirements Ci and Cd, it follows with Proposition 3 that $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}) = L \odot B$. Since $L \odot B$ is inevitably faulty with respect to $\mathcal{P}$, $\mathcal{X}$ is a sufficient cause for the violation of $\mathcal{P}$ in L. □

Proposition 10. *Let $\mathcal{P}$ be a property that is violated in log $L = (\mathbb{0}, \mathcal{L})$. If the faults of a subset $\mathcal{I}$ of components are a necessary (resp. sufficient) cause for the violation of $\mathcal{P}$ in L, then the faults of components $[n] \setminus \mathcal{I}$ are not a sufficient (resp. necessary) cause for the violation of $\mathcal{P}$ in L.*

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 $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}) \models \mathcal{P}$, thus $\mathsf{CF}_{\mathcal{X}}(\mathcal{L})$ is not inevitably faulty, thus the faults of components $[n] \setminus \mathcal{I}$ are not a sufficient cause (since $\mathcal{Y} = \mathcal{B} \setminus \mathcal{X} = \{c \in \mathcal{B} \mid \forall i \in \mathcal{I} : c_{\downarrow \mathbb{E}_i} \in \mathcal{C}_i\}$).

Conversely, if the faults of the components in $\mathcal{I}$ are 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 $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}) \not\models \mathcal{P}$, and the faults of components $[n] \setminus \mathcal{I}$ are not a necessary cause. □

4.3 Fault Ascription

In this section we apply the causality analysis for properties defined in the previous section to the problem of fault ascription. As in the previous section, we consider only well-formed counterfactual builders. Also, we consider the case of system specifications that are *consistent with respect to the given (hyper)property*, which amounts to say that when all the components behave according to their specification, the system does not fail. Formally:

Definition 24 (Consistent system specification)**.** *A consistently specified system is a tuple $(\sigma, \mathbf{P})$ where $\sigma = (\mathbf{S}, B)$ is a system specification with $\mathbf{S} = ((\mathbb{E}_i, \mathcal{C}_i))_{i \in [n]}$, and $\mathbf{P}$ is a hyperproperty such that $\|_{i \in [n]}\ \mathcal{C}_i \cap B \models \mathbf{P}$.*

With a consistent specification, hyperproperty $\mathbf{P}$ may be violated only if at least one of the components violates its specification. Our definition of necessary causality is complete with consistently specified systems, in the following sense, provided $\mathbf{P}$ is subset-closed.

Proposition 11 (Completeness of necessary causality for subset-closed hyperproperties)**.** *In a consistently specified system, if $\mathsf{CF}_{\mathcal{X}}$ satisfies Requirement* Snd *then each violation of $\mathbf{P}$ in log system $\mathbf{L} = \{(\mathbb{0}, \mathcal{L}^j) \mid j \in J\}$ has a necessary cause in $\mathcal{B} \setminus \|_{i \in [n]} \mathcal{C}_i$.*

Proof. Let $\mathcal{X} = \mathcal{B} \setminus \|_{i \in [n]} \mathcal{C}_i$, $\mathcal{D}^j = \mathsf{CF}_{\mathcal{X}}(\mathcal{L}^j)$, and $\mathcal{D} = \bigcup_{j \in J} \mathcal{D}^j$. Because $\mathsf{CF}_{\mathcal{X}}$ verifies Snd, we have for all $j \in J$ $\mathcal{D}^j \subseteq \mathcal{B} \setminus \mathcal{X} = \mathcal{B} \cap \|_{i \in [n]} \mathcal{C}_i$. Hence we have $\mathcal{D} \subseteq \mathcal{B} \cap \|_{i \in [n]} \mathcal{C}_i$ that is, $\mathcal{D}$ contains only observations consistent with executions where all components behave correctly. By consistent specification, because $\mathbf{P}$ is subset-closed, we have $\mathsf{CF}_{\mathcal{X}}(\mathcal{L}) \models \mathbf{P}$, thus $\mathcal{X}$ is a necessary cause for the violation of $\mathbf{P}$ in $\mathbf{L}$. □

In the case of a property $\mathcal{P}$, any sufficient cause contains some configuration of the log where at least one component has violated its specification:

Proposition 12 (Soundness of sufficient causality)**.** *If the faults of components $\mathcal{I}$ are a sufficient cause for the violation of $\mathcal{P}$ in the log $L = (\mathbb{0}, \mathcal{L})$, and $\mathsf{CF}_{\mathcal{X}}$ verifies Requirements* Snd *and* Cinv*, then $(L \odot B)_{\downarrow \mathbb{E}_i} \not\subseteq \mathcal{C}_i$ for some $i \in \mathcal{I}$.*

Proof. Let $\mathcal{I}$ be such that $\forall i \in \mathcal{I}$, $(L \odot B)_{\downarrow \mathbb{E}_i} \subseteq \mathcal{C}_i$. In order to check for sufficient causality of the faults of the components in $\mathcal{I}$ we compute $\mathcal{Y} = \{c \in \mathcal{B} \mid \forall i \in [n] \setminus \mathcal{I} : c_{\downarrow \mathbb{E}_i} \in \mathcal{C}_i\}$ and $\mathcal{X} = \mathcal{B} \setminus \mathcal{Y} = \{c \in \mathcal{B} \mid \exists i \in [n] \setminus \mathcal{I} : c_{\downarrow \mathbb{E}_i} \notin \mathcal{C}_i\}$. Let $\mathcal{C} = \mathsf{CF}_{\mathcal{X}}(\mathcal{L})$. Because $\mathsf{CF}_{\mathcal{X}}$ verifies Snd and Cinv we have

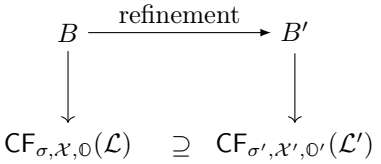


Figure 6: Anti-monotony of the counterfactual builder under refinement of the behavioral model.

$\mathcal{C} \subseteq \mathcal{B} \setminus \mathcal{X} \ \wedge \ \forall i \in [n] : \big((L \odot B)_{\downarrow \mathbb{E}_i} \subseteq \mathcal{C}_i \Rightarrow \mathsf{CF}_{\mathcal{X}}(\mathcal{L})_{\downarrow \mathbb{E}_i} \subseteq \mathcal{C}_i\big)$, 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 faults of components in $\mathcal{I}$ are not a sufficient cause for the violation of $\mathcal{P}$. □

4.4 Refinement

In Section 4.1 we considered intrinsic requirements on counterfactual builders. We now consider different properties relating causality analysis and refinement. The idea is that counterfactuals built from a refinement of a given behavioral model B are finer than the counterfactuals built from B, as illustrated in Figure 6.

The following property characterizes monotony of CF under refinement of B, for the case where component specifications are always satisfied and $\mathcal{X}$ is included in the refined behavioral model. Proposition 1 ensures that causality analysis composes with abstractions that ensure trace inclusion.

Requirement 8 (Refinement by model inclusion)**.** *Given system specifications $\sigma = (\mathbf{S}, B)$ and $\sigma' = (\mathbf{S}, B')$ with $B' \subseteq B$, and a log $(\mathbb{O}, \mathcal{L})$ in which a violation of a property $\mathcal{P}$ is detected. Then $\mathsf{CF}_{\sigma',\mathcal{X},\mathbb{O}}(\mathcal{L}) \subseteq \mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L})$, and necessary causality of the violation of $\mathcal{P}$ is invariant under refinement.*

Definition 25 (Alphabet inclusion $\preceq$)**.** *$C' = (\mathbb{E}', \mathcal{D})$ refines $C = (\mathbb{E}, \mathcal{C})$ by* alphabet inclusion*, noted $C' \preceq C$, if $C = C'_{\downarrow \mathbb{E}}$.*

Requirement 9 (Refinement by alphabet inclusion)**.** *Given system specifications $\sigma = (\mathbf{S}, B)$ and $\sigma' = (\mathbf{S}', B')$ with $\mathbf{S}' = ((\mathbb{E}'_i, \mathcal{C}'_i))_{i \in I}$ and $\mathbf{S} = ((\mathbb{E}_i, \mathcal{C}_i))_{i \in I} \triangleq \mathbf{S}'_{\downarrow \mathbb{B}}$ (where projection is applied component-wise), $B' = (\mathbb{B}', \mathcal{B}') \preceq B = (\mathbb{B}, \mathcal{B})$, a log $L' = (\mathbb{O}', \mathcal{L}')$ with $\mathbb{O}' \subseteq \mathbb{B}'$, exclusion sets $\mathcal{X} \subseteq \mathcal{B}$ and $\mathcal{X}' \subseteq \mathcal{B}'$ such that $\mathcal{B} \setminus \mathcal{X} = (\mathcal{B}' \setminus \mathcal{X}')_{\downarrow \mathbb{B}}$, and a property $\mathcal{P}$ such that a violation of $\mathcal{P}$ is detected by $L = (\mathbb{O}, \mathcal{L}) \triangleq L'_{\downarrow \mathbb{B}}$, then*

$$\mathsf{CF}_{\sigma',\mathcal{X}',\mathbb{O}'}(\mathcal{L}')_{\downarrow \mathbb{B}} \subseteq \mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L})$$

If in addition $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L}) \models \mathcal{P}$ then $\mathsf{CF}_{\sigma',\mathcal{X}',\mathbb{O}'}(\mathcal{L}') \models \mathcal{P}^{\uparrow \mathbb{B}'} \cap \mathcal{B}'$, that is, necessary causality is invariant under refinement.

Notice that there is a qualitative difference between the requirement Obs and the above property on refinement by alphabet inclusion: while the former varies the observable events but keeps the behavioral model constant, the latter allows us to reason about refinement of the behavioral model, in the case of full observability.

5 An Instantiation

In this section we propose a concrete definition $\mathsf{CF}^*_{\sigma,\mathcal{X},\mathbb{O}}$ of a counterfactual builder that we call WfE counterfactuals (for Wf and Exh). This instantiation constructs from a log L a counterfactual configuration structure where the exclusion set $\mathcal{X}$ to be checked for causality and its effects are eliminated and replaced with alternative behaviors.

5.1 WfE Counterfactuals

We now develop a concrete and constructive definition of a counterfactual builder $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}$.

Definition 26 ($\mathsf{CF}^*_{\sigma,\mathcal{X},\mathbb{O}}$)**.** *For a system specification $\sigma = (\mathbf{S}, B)$ with $B = (\mathbb{B}, \mathcal{B})$, a log $L = (\mathbb{O}, \mathcal{L})$ with $\mathbb{O} \subseteq \mathbb{B}$, and a set $\mathcal{X} \subseteq \mathcal{B}$, let $\mathcal{C} = L \odot B$, $R = \mathtt{lpc}_{\mathcal{C}}(\mathcal{C} \setminus \mathcal{X})$, and*

$$\begin{aligned} P_c &= \max\left\{c' \mid c' \subseteq c \wedge \exists r \in R : c' \subseteq r \wedge \neg\mathcal{X}(c') \wedge \forall i \in [n] : c'_{\downarrow\mathbb{E}_i} \in \mathcal{B}_{\downarrow\mathbb{E}_i}\right\} \\ Z_c &= \left\{d \in \mathcal{B} \mid c \subseteq d \wedge \textstyle\bigcirc_{\mathcal{B}}(\{d\} \cup R_{\downarrow\mathbb{O}})\right\} \\ G_c &= \mathtt{lpc}_{Z_c}\left(\{c' \in Z_c \setminus \mathcal{X} \mid inv^+_{\mathbf{S}}(L, \{c'\})\}\right) \\ \mathsf{CF}^*_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L}) &= R \cup \bigcup_{c \in \mathcal{C} \cap \mathcal{X}} \bigcup_{c' \in P_c} G_{c'} \end{aligned}$$

As for CF we omit the subscripts σ and $\mathbb{O}$ in $\mathsf{CF}^*_{\sigma,\mathcal{X},\mathbb{O}}$ when there is no ambiguity.

In Definition 26, $\mathcal{C}$ is the set of configurations that are consistent with the log. An ascending chain of configurations from $\emptyset$ to a configuration $c \in \mathcal{C}$ can be seen as an *explanation* of how c may have been reached in L. Intuitively, configurations of $\mathcal{C} \setminus \mathcal{X}$ that cannot be explained in $\mathcal{C}$ represent effects of $\mathcal{X}$ that would not have occurred in the absence of $\mathcal{X}$. The role of $\mathtt{lpc}_{\mathcal{C}}$ in the definition of R is to remove those configurations from $\mathcal{C} \setminus \mathcal{X}$. The construction of P_c, for a faulty configuration $c \in \mathcal{C} \cap \mathcal{X}$, restricts c to the observations of the maximal non-faulty sub-configurations that are consistent, for each component, with the projection of the behavioral model, provided they are smaller than some element of R. Intuitively, the definition of P_c implements "back-tracking" to the maximal configurations smaller than c that are still explainable without $\mathcal{X}$. For each "pruned" configuration $c' \in P_c$, $Z_{c'}$ is the set of extensions that are consistent with the preserved part $R_{\downarrow\mathbb{O}}$ of the log. Finally, G_c is the maximal prefix-closed subset of Z_c that satisfies correctness with respect to $\mathcal{X}$ and invariance of the component specifications, that is, no new component faults are introduced. The counterfactual configuration structure $\mathsf{CF}^*_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L})$ is obtained as the union of the preserved configurations R from the filtered log and the counterfactual configurations in G_c. Notice that, while all configurations in $\mathsf{CF}^*_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L})$ are consistent with the portion R of the expanded log, they may not be consistent among each other, that is, they may represent traces of alternative, non-confluent executions.

5.2 Properties

We first state a lemma that establishes a condition for monotony of the maximal prefix-closed subset of configuration structures. This lemma will allow us to reason about the behavior of CF^* under varying observability, logs, and refinement.

Lemma 1. *Let $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$ be sets of configurations. If $\mathcal{C}_1 \subseteq \mathcal{C}_2$, $(\mathcal{C}_2 \setminus \mathcal{C}_1) \cap \mathcal{C}_4 > \mathtt{lpc}_{\mathcal{C}_1}(\mathcal{C}_1 \setminus \mathcal{C}_3)$, and $\mathcal{C}_4 \subseteq \mathcal{C}_3$ then $\mathtt{lpc}_{\mathcal{C}_1}(\mathcal{C}_1 \setminus \mathcal{C}_3) \subseteq \mathtt{lpc}_{\mathcal{C}_2}(\mathcal{C}_2 \setminus \mathcal{C}_4)$.*

Proof. Let $c \in \mathtt{lpc}_{\mathcal{C}_1}(\mathcal{C}_1 \setminus \mathcal{C}_3)$, hence $c \in \mathcal{C}_2$. If $c \in \min \mathcal{C}_2$ the claim follows with $\mathcal{C}_4 \subseteq \mathcal{C}_3$. Otherwise let $M = \max\{c' \in \mathcal{C}_2 \mid c' \subsetneq c\}$. By $c \in \mathtt{lpc}_{\mathcal{C}_1}(\mathcal{C}_1 \setminus \mathcal{C}_3)$ it follows that $\exists c' \in M$ such that either $c' \in \mathtt{lpc}_{\mathcal{C}_1}(\mathcal{C}_1 \setminus \mathcal{C}_3)$, or $c' \in \mathcal{C}_2 \setminus \mathcal{C}_1$ such that $\exists c'' \in \mathtt{lpc}_{\mathcal{C}_1}(\mathcal{C}_1 \setminus \mathcal{C}_3) : c'' \subseteq c'$. By hypothesis we have that $(\mathcal{C}_2 \setminus \mathcal{C}_1) \cap \mathcal{C}_4 > \{c\}$, hence $(\mathcal{C}_2 \setminus \mathcal{C}_1) \cap \mathcal{C}_4 > M$. It follows that $c' \in \mathtt{lpc}_{\mathcal{C}_1}(\mathcal{C}_1 \setminus \mathcal{C}_3) \cup \big(\mathcal{C}_2 \setminus (\mathcal{C}_1 \cup \mathcal{C}_4)\big)$. With $\mathcal{C}_1 \subseteq \mathcal{C}_2$ and $\mathcal{C}_4 \subseteq \mathcal{C}_3$ it follows that $c' \in \mathcal{C}_2 \setminus \mathcal{C}_4$. We recursively apply the same reasoning to construct a descending chain from c in $\mathcal{C}_2$. It follows that there exists a descending chain from c to $\min \mathcal{C}_2$ in $\mathcal{C}_2$ that remains in $\mathcal{C}_2 \setminus \mathcal{C}_4$, that is, $c \in \mathtt{lpc}_{\mathcal{C}_2}(\mathcal{C}_2 \setminus \mathcal{C}_4)$. □

Lemma 2. *Given logs $L = (\mathbb{O}, \mathcal{L})$ and $L' = (\mathbb{O}, \mathcal{L}')$, if $L \sqsubseteq L'$ then $L \odot B \sqsubseteq L' \odot B$.*

Proof. $L \odot B \subseteq L' \odot B$ follows with the property $\bigcirc_{\mathcal{B}}\mathcal{L}'$ of L' from $L \sqsubseteq L'$. Let $c \in L \odot B$ and $c' \in (L' \odot B) \setminus (L \odot B)$, hence $c'_{\downarrow\mathbb{O}} \in L' \setminus L$. By $L \sqsubseteq L'$ it follows that $\neg(c' \subseteq c)$. Therefore, $L \odot B \sqsubseteq L' \odot B$. □

Lemma 3. *For a behavioral model $B = (\mathbb{B}, \mathcal{B})$, log $L = (\mathbb{O}, \mathcal{L})$, $\mathcal{X} \subseteq \mathcal{B}$, and $\mathbb{O}' \subseteq \mathbb{O}$, if $\mathcal{X}_{\downarrow\mathbb{O}'}{}^{\uparrow\mathbb{B}} = \mathcal{X}$ then $\big((L \odot B) \setminus \mathcal{X}\big)_{\downarrow\mathbb{O}'} = \big((L_{\downarrow\mathbb{O}'} \odot B) \setminus \mathcal{X}\big)_{\downarrow\mathbb{O}'}$.*

Proof. "$\subseteq$" follows from the Definition of filtering and monotony of set difference and projection. "$\supseteq$": suppose that $c \in (L_{\downarrow\mathbb{O}'} \odot B) \setminus \mathcal{X}$. Thus, $c \in \mathcal{B} \setminus \mathcal{X}$ and $c_{\downarrow\mathbb{O}'} \in L_{\downarrow\mathbb{O}'}$. As L is a log we have that $\bigcirc_{\mathcal{B}} L$. Therefore there is some $\hat{c} \in L \odot B$ such that $\hat{c}_{\downarrow\mathbb{O}'} = c_{\downarrow\mathbb{O}'}$. From $c \notin \mathcal{X}$ it follows with $\mathcal{X}_{\downarrow\mathbb{O}'}{}^{\uparrow\mathbb{B}} = \mathcal{X}$ that $\hat{c} \notin \mathcal{X}$. Therefore, $c_{\downarrow\mathbb{O}'} \in \big((L \odot B) \setminus \mathcal{X}\big)_{\downarrow\mathbb{O}'}$. □

Theorem 1. *CF^* is well-formed.*

Proof. Let $\sigma = (\mathbf{S}, B)$ with $B = (\mathbb{B}, \mathcal{B})$ and $L = (\mathbb{O}, \mathcal{L})$. We have to show the following properties:

- SND: by Definition 26, all elements of $\mathsf{CF}^*_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L})$ are either in R or in some G_c, any of which is a subset of $\mathcal{B} \setminus \mathcal{X}$. Hence, $\mathsf{CF}^*_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L}) \subseteq \mathcal{B} \setminus \mathcal{X}$.
- CI: we have by Definition 16 $\{c \in \mathcal{C} \mid \forall c' \in \mathcal{C} \cap \mathcal{X} : \neg(c' \subseteq c)\} \subseteq \mathtt{lpc}_{\mathcal{C}}(\mathcal{C} \setminus \mathcal{X})$ where $\mathcal{C} = L \odot B$ and by Definition 26, $\mathtt{lpc}_{\mathcal{C}}(\mathcal{C} \setminus \mathcal{X}) \subseteq \mathsf{CF}^*_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L})$.
- CD: by Definition 26 we have $\mathsf{CF}^*_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L}) \setminus (L \odot B) \subseteq \bigcup_{c \in \mathcal{C} \cap \mathcal{X}} \bigcup_{p \in P_c} G_p \subseteq \bigcup_{c \in \mathcal{C} \cap \mathcal{X}} \bigcup_{p \in P_c} \{c \in \mathcal{B} \mid p \subseteq c\}$. By definition of P_c, for any $c' \in P_c$ we have that $\exists c'' \in \mathtt{pre}_{L \odot B}(\mathcal{X})$ such that $c'' \subseteq c'$. Therefore $\bigcup_{c \in \mathcal{C} \cap \mathcal{X}} \bigcup_{p \in P_c} \{c \in \mathcal{B} \mid p \subseteq c\} \subseteq \{c \in \mathcal{B} \mid \exists c' \in \mathtt{pre}_{L \odot B}(\mathcal{X}) : c' \subseteq c\}$. The claim follows.
- CINV: in Definition 26, $(L \odot B)_{\downarrow \mathbb{E}_i} \subseteq \mathcal{C}_i$ implies that the projections of R and all G_c are in $\mathcal{C}_i$. Hence, $\forall i \in [n] : \big((L \odot B)_{\downarrow \mathbb{E}_i} \subseteq \mathcal{C}_i \implies \mathsf{CF}^*_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L})_{\downarrow \mathbb{E}_i} \subseteq \mathcal{C}_i\big)$.
- OBS: this proof is slightly more involved. If $L = (\mathbb{O}_2, \mathcal{L})$ and $\mathbb{O}_1 \subseteq \mathbb{O}_2$ then with $\mathcal{C}_1 = L_{\downarrow \mathbb{O}_1} \odot B$ and $\mathcal{C}_2 = L \odot B$ it follows that $\mathcal{C}_2 \subseteq \mathcal{C}_1$ and $\mathcal{C}_2 \setminus \mathcal{X} \subseteq \mathcal{C}_1 \setminus \mathcal{X}$. By $\mathcal{X} \subseteq \mathcal{C}_2$ and Lemma 1 (with $(\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4)$ of Lemma 1 instantiated with $(\mathcal{C}_2, \mathcal{C}_1, \mathcal{X}, \mathcal{X})$), $R_2 \subseteq R_1$, where we use indices to distinguish the intermediate results computed with $\mathbb{O}_1$ and $\mathbb{O}_2$ in Definition 26.

  Next we show that $P_{c,2} \subseteq P_{c,1}$. To this end we first show that between any two configurations $c_1, c_2 \in R_2$ such that $\forall c \in R_2 : c_1 \subseteq c \subseteq c_2 \implies c = c_1 \vee c = c_2$, there is no configuration $c \in R_1$ "between" both, formally: $\forall c \in R_1 : c_1 \subseteq c \subseteq c_2 \implies c = c_1 \vee c = c_2$. Suppose on the contrary that for some c_1 and c_2 such a $c \in R_1$ strictly between c_1 and c_2 exists. As $c \notin R_2$ it follows that either (a) $c \notin \mathcal{C}_2$ or (b) $c \notin \mathtt{lpc}_{\mathcal{C}_2}(\mathcal{C}_2 \setminus \mathcal{X})$ due to a configuration $c' \in \mathcal{X}$ with $c_1 \subsetneq c' \subseteq c$. (a) is excluded since by hypothesis $\mathtt{pc}_{B_{\downarrow \mathbb{O}_2}}(L)$, such that either $c_{\downarrow \mathbb{O}_2} = (c_1)_{\downarrow \mathbb{O}_2}$ or $c_{\downarrow \mathbb{O}_2} = (c_2)_{\downarrow \mathbb{O}_2}$. In both cases it follows that $c \in \mathcal{C}_2$, which contradicts the assumption of (a). (b) is excluded by $\mathcal{X}_{\downarrow \mathbb{O}_1}^{\uparrow \mathbb{O}_2} = \mathcal{X}$ and $c_1 \in R_2$. It follows that for any $c \in \mathcal{C}_2 \cap \mathcal{X}$ and $c' \in P_{c,2}$ satisfying the condition of set $P_{c,2}$ for some $r \in R_2$ in Definition 26, $c' \in P_{c,1}$ using the same r, hence $P_{c,2} \subseteq P_{c,1}$.

  By Lemma 3, $(R_1)_{\downarrow \mathbb{O}_1} = (R_2)_{\downarrow \mathbb{O}_1}$, and thus $\bigcup (R_1)_{\downarrow \mathbb{O}_1} \subseteq \bigcup (R_2)_{\downarrow \mathbb{O}_1} \subseteq \bigcup (R_2)_{\downarrow \mathbb{O}_2}$. It follows, by the definition of consistency, that $Z_{c,2} \subseteq Z_{c,1}$. By Lemma 1 with $\bar{\mathcal{C}}_1 = \{c' \in Z_{c,2} \mid inv^+_{\mathbf{S}}(\mathcal{C}_2, \{c'\})\}$, $\bar{\mathcal{C}}_2 = \{c' \in Z_{c,1} \mid inv^+_{\mathbf{S}}(\mathcal{C}_1, \{c'\})\}$, and $\bar{\mathcal{C}}_3 = \bar{\mathcal{C}}_4 = \mathcal{X}$ (where $\bar{\mathcal{C}}_1 \subseteq \bar{\mathcal{C}}_2$ follows from $Z_{c,2} \subseteq Z_{c_1}$ and $\mathcal{C}_2 \subseteq \mathcal{C}_1$) we have that $G_{c,2} \subseteq G_{c,1}$ and thus $\mathsf{CF}^*_{\sigma,\mathcal{X},\mathbb{O}_2}(\mathcal{L}) \subseteq \mathsf{CF}^*_{\sigma,\mathcal{X},\mathbb{O}_1}(\mathcal{L}_{\downarrow \mathbb{O}_1})$.
- INC: suppose that $L \sqsubseteq L'$, $L \odot B \subseteq L' \odot B$, and $inv_{\mathbf{S}}(L \odot B, L' \odot B)$. Let $\mathcal{C}$, R, P_c, Z_c, and G_c be defined as in Definition 26, and let $\mathcal{C}'$, R', P'_c, Z'_c and G'_c denote the analog sets defined on L' instead of L. Using Lemma 2 we obtain $\mathcal{C} \sqsubseteq \mathcal{C}'$. By Lemma 1 with $(\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4) = (\mathcal{C}, \mathcal{C}', \mathcal{X}, \mathcal{X})$ and using $\mathcal{C} \sqsubseteq \mathcal{C}'$ it follows that $\mathtt{lpc}_{\mathcal{C}}(\mathcal{C} \setminus \mathcal{X}) \subseteq \mathtt{lpc}_{\mathcal{C}'}(\mathcal{C}' \setminus \mathcal{X})$, hence $R \subseteq R'$. With $\mathcal{C} \sqsubseteq \mathcal{C}'$ it follows that $R' \setminus R \subseteq \mathcal{C}' \setminus \mathcal{C}$, and therefore $R \sqsubseteq R'$. Using this result it follows that $P_c \subseteq P'_c$. For any $c \in \bigcup_{c' \in \mathcal{C} \cap \mathcal{X}} P'_{c'}$ and $d \in Z'_c$, $c \subseteq d$ and $\bigcirc_{\mathcal{B}}(\{d\}, R'_{\downarrow \mathbb{O}})$, hence $\bigcirc_{\mathcal{B}}(\{d\}, R_{\downarrow \mathbb{O}})$ (by $R \subseteq R'$), and $Z'_c \subseteq Z_c$. Since $Z_c \setminus Z'_c$ consists of configurations that are consistent with $R_{\downarrow \mathbb{O}}$ but inconsistent with $R'_{\downarrow \mathbb{O}}$, no configuration of the difference is contained in any configuration of Z'_c, hence $Z'_c \sqsubseteq Z_c$. Furthermore, by Lemma 1 where $\mathcal{C}_1 = \{c' \in Z'_c \mid inv^+_{\mathbf{S}}(\mathcal{C}', \{c'\})\}$, $\mathcal{C}_2 = \{c' \in Z_c \mid inv^+_{\mathbf{S}}(\mathcal{C}, \{c'\})\}$, and $\mathcal{C}_3 = \mathcal{C}_4 = \mathcal{X}$, using $Z'_c \sqsubseteq Z_c$, and $inv_{\mathbf{S}}(\mathcal{C}, \mathcal{C}')$, we obtain $G'_c \subseteq G_c$. It follows that $\mathsf{CF}^*_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L}') \setminus (L' \odot B) \subseteq \mathsf{CF}^*_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L}) \setminus (L \odot B)$.

□

Theorem 2 (EXH)**.** *CF^* satisfies Requirement 7.*

Proof. Let $\sigma = (\mathbf{S}, B)$ with $B = (\mathbb{B}, \mathcal{B})$ and $L = (\mathbb{O}, \mathcal{L})$. Let $\mathcal{X}$ and $\mathcal{C}' \subseteq \mathcal{B} \setminus \mathcal{X}$ be such that $\mathtt{pre}_{\mathcal{C}}(\mathcal{C} \cap \mathcal{X}) \leq \mathcal{C}'$, $\bigcirc_{\mathcal{B}}(R \cup \mathcal{C}')$, $inv_{\mathbf{S}}(R, \mathcal{C}')$, and $\mathtt{pc}_{\mathcal{B}}(R \cup \mathcal{C}')$, where $\mathcal{C} = L \odot B$ and $R = \mathtt{lpc}_{\mathcal{C}}(\mathcal{C} \setminus \mathcal{X})$. Let $d \in \mathcal{C}'$. From $\mathtt{pre}_{\mathcal{C}}(\mathcal{C} \cap \mathcal{X}) \leq \mathcal{C}'$ it follows that there is some $\hat{c} \in \mathcal{C} \cap \mathcal{X}$ and $c_0 \in \mathtt{pre}_{\mathcal{C} \setminus \mathcal{X}}(\mathcal{C} \cap \mathcal{X})$ such that $c_0 \subseteq d$ and $c_0 \subseteq \hat{c}$. Hence $P_{\hat{c}} \neq \emptyset$. Take some $c \in P_{\hat{c}}$. With $\bigcirc_{\mathcal{B}}(R \cup \mathcal{C}')$ it follows that $d \in Z_c$. By hypothesis we have $\mathtt{pc}_{\mathcal{B}}(R \cup \mathcal{C}')$, therefore there is a prefix-closed chain $\mathcal{C}''$ in $\mathcal{B}$ from $\emptyset$ to d that is contained in $R \cup \mathcal{C}'$, such that $\mathcal{C}'' \cup \{c\}$ is again a chain (c may not be an element of $\mathcal{C}''$ since by definition, $P_{\hat{c}}$ is not necessarily included in $\mathcal{B}$). Again by hypothesis we have that $inv_{\mathbf{S}}(R, \mathcal{C}')$, hence $inv^+_{\mathbf{S}}(L, \mathcal{C}')$. With $\mathcal{C}'' \cap \mathcal{X} = \emptyset$ it follows that $\mathcal{C}'' \cap Z_c \subseteq \mathtt{lpc}_{Z_c}\big(\{c' \in Z_c \setminus \mathcal{X} \mid inv^+_{\mathbf{S}}(L, \{c'\})\}\big)$. Therefore $d \in G_c$, and $d \in \mathsf{CF}^*_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L})$. □

Theorem 3 (Refinement by model inclusion)**.** *CF^* satisfies Requirement 8 provided that $(B \setminus B') \cap \mathcal{X} > B'$ and $\{c \in B \mid \exists c' \in L \odot B : c \subseteq c'\} = \{c \in B' \mid \exists c' \in L \odot B' : c \subseteq c'\}$.*

Proof. Let $\sigma = (\mathbf{S}, B)$ and $\sigma' = (\mathbf{S}, B')$ such that (1) $B' \subseteq B$, $(B \setminus B') \cap \mathcal{X} > B'$, and (2) $\{c \in B \mid \exists c' \in \mathcal{C} : c \subseteq c'\} = \{c \in B' \mid \exists c' \in \mathcal{C}' : c \subseteq c'\}$, where $\mathcal{C} = L \odot B$ and $\mathcal{C}' = L \odot B'$. In Definition 26 we have by (2) $R = R'$ and $P_c = P'_c$, where primed variables denote the results obtained by operating on B'. Furthermore, $Z'_c \subseteq Z_c \cap B'$ by (1) and $R = R'$. By Lemma 1 with $(\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4) = (Z'_c, Z_c, \mathcal{X}, \mathcal{X})$, using $Z'_c \subseteq Z_c \cap B'$ and (2), it follows that $G'_c \subseteq G_c$, and using $P_c = P'_c$, $\mathsf{CF}^*_{\sigma',\mathcal{X},\mathbb{0}}(\mathcal{L}) \subseteq \mathsf{CF}^*_{\sigma,\mathcal{X},\mathbb{0}}(\mathcal{L})$. With $L \not\models \mathcal{P}_{\downarrow\mathbb{0}}$ it follows that every necessary cause in B is also a necessary cause in B'. □

Theorem 4 (Refinement by alphabet inclusion)**.** *CF^* satisfies Requirement 9 provided that $\forall c' \in B' : \big((\exists c \in L' \odot B' : c' \subseteq c) \implies (c'_{\downarrow\mathbb{B}} \in \mathcal{X} \iff c' \in \mathcal{X}')\big)$ and $\forall c' \in L' \odot B'\ \forall i : (c'_{\downarrow\mathbb{B}} \in \mathcal{C}_i \iff c' \in \mathcal{C}'_i)$.*

Proof. Let $\sigma = (\mathbf{S}, B)$ and $\sigma' = (\mathbf{S}, B')$ such that $B' \preceq B$ and (1) $\mathbf{S} = \mathbf{S}'_{\downarrow\mathbb{B}}$, (2) $\mathcal{B} \setminus \mathcal{X} = (\mathcal{B}' \setminus \mathcal{X}')_{\downarrow\mathbb{B}}$, (3) $\forall c' \in B' : \big((\exists c \in L' \odot B' : c' \subseteq c) \implies (c'_{\downarrow\mathbb{B}} \in \mathcal{X} \iff c' \in \mathcal{X}')\big)$, and (4) $\forall c' \in L' \odot B'\ \forall i : (c'_{\downarrow\mathbb{B}} \in \mathcal{C}_i \iff c' \in \mathcal{C}'_i)$. Let $\mathcal{C} = L \odot B$, $\mathcal{C}' = L' \odot B'$, and primed variables denote the results obtained by operating on B'. By (3) we have $\mathcal{C}'_{\downarrow\mathbb{B}} \subseteq \mathcal{C}$. By (2) and (3) we have $\mathcal{C} \setminus \mathcal{X} = (\mathcal{C}' \setminus \mathcal{X}')_{\downarrow\mathbb{B}}$. Thus $R'_{\downarrow\mathbb{B}} = R$. Furthermore, $P_c = (P'_c)_{\downarrow\mathbb{B}}$, and $(Z'_c)_{\downarrow\mathbb{B}} \subseteq Z_c$ due to the stronger consistency constraint on the elements of Z'_c. With $\mathcal{X} \subseteq \mathcal{B}$ and (1), (2), (4) it follows that $\mathtt{lpc}_{Z'_c}\big(\{c' \in Z'_c \setminus \mathcal{X}' \mid inv^+_{\mathbf{S}'}(\mathcal{C}', \{c'\})\}\big)_{\downarrow\mathbb{B}} \subseteq \mathtt{lpc}_{Z_c}\big(\{c' \in Z_c \setminus \mathcal{X} \mid inv^+_{\mathbf{S}}(\mathcal{C}, \{c'\})\}\big)$. Hence, $\mathsf{CF}^*_{\sigma',\mathcal{X}',\mathbb{0}}(\mathcal{L}')_{\downarrow\mathbb{B}} \subseteq \mathsf{CF}^*_{\sigma,\mathcal{X},\mathbb{0}}(\mathcal{L})$, and if $\mathsf{CF}^*_{\sigma,\mathcal{X},\mathbb{0}}(\mathcal{L}) \models \mathcal{P}$ then $\mathsf{CF}^*_{\sigma',\mathcal{X}',\mathbb{0}}(\mathcal{L}') \models \mathcal{P}^{\uparrow\mathbb{B}'} \cap \mathcal{B}'$. □

Remark 17 (Comparison with the definitions of [13, 15])**.** *Formalizations of counterfactual analyses for concurrent systems have been proposed in previous work [13, 15]. None of them is satisfactory: [13] assumes all events to be observable, whereas [15] does not guarantee well-formedness nor monotony under refinement.*

We can show that CF^* is not comparable with the counterfactual function of Definition 9 of [13] and the definition of CF in [15]. Intuitively, property Exh *entails a form of exhaustiveness of CF^* whereas an (informally stated) objective of the definitions in [13, 15] was closeness of the counterfactual behaviors to the observed behavior.*

Vice versa, the counterfactuals of Definition 9 of [13] are not included in CF^*. This is because CF^* preserves more information about the observed behavior than [13]. Consider for instance the set $L \odot B = \big\{c_1 \cup c_2 \mid c_1 \in \{\emptyset, \{p_1\}, \{p_1, v_1\}\} \wedge c_2 \in \{\emptyset, \{p_2\}, \{p_2, v_2\}\big\} =: \mathcal{C}$ modeling a situation where two components have taken and then released a shared resource, temporarily violating mutual exclusion in $\mathcal{X} := \big\{\{p_1, p_2\}\big\}$. According to Definition 26 the set of preserved configurations is $R = \mathcal{C} \setminus \mathcal{X}$, that is, all non-faulty configurations are preserved. In contrast, in [13] the "unaffected prefix" is conservatively computed as the empty trace, as the formalism used there is not able to represent alternative correct prefixes. Extending R is then subject to stricter consistency constraints than extending the empty prefix.

Comparing CF^* with [15], CF^* is not a superset of the definition of [15] in general, due to the requirement that the configurations in Z_c (Definition 26) are consistent with all preserved configurations of the expanded log in $R_{\downarrow\mathbb{0}}$, for which there is no counterpart in [15].

5.3 Comparison with Halpern and Pearl's Actual Causality

While a formal comparison is beyond the scope of this paper, we will informally discuss how our approach relates to [19] where actual causality is defined on a recursive structural equations model (SEM). Intuitively, a SEM $M = (\mathcal{S}, \mathcal{F})$ with $\mathcal{S} = (\mathcal{U}, \mathcal{V}, \mathcal{R})$ consists of a set of variables partitioned into *exogenous* variables $\mathcal{U}$ modeling inputs from the environment and *endogenous* variables $\mathcal{V}$, with the values of each variable $X \in \mathcal{V}$ being defined, on the domain $\mathcal{R}_X$, by function $\mathcal{F}_X$ depending on a subset of $\mathcal{U} \cup \mathcal{V}$. For recursive SEM this dependency relation is acyclic.

A direct comparison between [19] and our work is made difficult by the following fundamental difference: recursive SEM are deterministic, and each intervention defines a unique counterfactual scenario. In contrast, our approach supports non-determinism. In particular, faulty behavior is not fixed to the observed one: when a faulty component is not fixed by intervention, then by Exh its counterfactual faulty behavior may differ from the observed one unless the variables on which it depends keep their actual values. Another, less fundamental difference lies in the fact that there is no notion of specifications

— hence, of faults — in actual causality, even though extensions of actual causality with a notion of normality have been proposed [18].

In order to compare both frameworks we will therefore proceed as follows. Consider a recursive SEM $M = (\mathcal{S}, \mathcal{F})$ with $\mathcal{S} = (\mathcal{U}, \mathcal{V}, \mathcal{R})$. We translate M into a system model $\sigma(M)$ in our framework. We define the behavioral model to consist of the set of all predecessor-closed partial valuations of $\mathcal{U} \cup \mathcal{V}$ that are consistent with $\mathcal{F}$. Furthermore, we define the universal component specifications $S_X = \{\emptyset, \{X \mapsto \mathcal{F}_X\}\}$, that is, the set of partial valuations of X consistent with $\mathcal{F}_X$. For a given valuation $\vec{u}$ of $\mathcal{U}$ and predicate $\mathcal{P}$ over $\mathcal{U} \cup \mathcal{V}$ satisfied in M for $\vec{u}$, written $(M, \vec{u}) \models \mathcal{P}$, we can now determine the actual causes of $(M, \vec{u}) \models \mathcal{P}$. On the other hand let $\mathcal{E}(\vec{u}) = \{\{X \mapsto x^*\} \mid X \in \mathcal{V}\}$, where x^* is the observed value of X. We use CF^* to determine the necessary causes among the exclusion sets $\mathcal{X} \subseteq \{\{e\} \mid e \in \mathcal{E}(\vec{u})\}$ for $\mathcal{P}$ in $\sigma(M)$ such that whenever $\{X \mapsto x^*\} \in \mathcal{X}$ then the valuations of all variables in $\mathcal{V}$ that transitively depend on X, are also in $\mathcal{X}$. We conjecture that (1) the results of Pearl's "do" operator and of CF^* coincide, and (2) for any minimal necessary cause $\mathcal{X}$ of $\mathcal{P}$ in $\sigma(M)$, $\bigwedge_{\{X_i \mapsto \cdot\} \in \mathcal{X}} X_i = x_i^*$ is an actual cause of $\mathcal{P}$ in M. A formal treatment of this comparison is left for future work.

5.4 Examples

Example 8 (Causal over-determination)**.** *Consider two components with specifications* $(\mathbb{E}_i, \mathcal{C}_i)$ *where* $\mathbb{E}_i = \{e_i, f_i\}$ *and* $\mathcal{C}_i = \{\emptyset\}$*,* $i = 1, 2$*, the behavioral model* $B = (\mathbb{E}, \mathcal{B})$ *with* $\mathbb{E} = \mathbb{E}_1 \cup \mathbb{E}_2$*,* $\mathcal{B} = \mathcal{B}_1 \| \mathcal{B}_2$*, and* $\mathcal{B}_i = \{\emptyset, \{e_i\}, \{e_i, f_i\}\}$*, the property* $\mathcal{P} = \neg(f_1 \vee f_2)$*, and the log* $L = (\mathbb{E}, \mathcal{L})$ *with* $\mathcal{L} = \{\emptyset, \{e_1\}, \{e_2\}, \{e_1, f_1, e_2\}\}$*. For* $\mathcal{X} = e_1$ *we have* $L \odot B = \mathcal{L}$*,*

$$\begin{aligned} R = \mathtt{lpc}_{\mathcal{L}}(\mathcal{L} \setminus \mathcal{X}) &= \{\emptyset, \{e_2\}\} \\ P_{\{e_1\}} &= \{\emptyset\} \\ P_{\{e_1, f_1, e_2\}} &= \{e_2\} \\ Z_\emptyset &= \mathcal{B} \\ Z_{\{e_2\}} &= \{c \in \mathcal{B} \mid e_2 \in c\} \\ G_{\{e_2\}} &= \{\{e_2\}, \{e_2, f_2\}\} \\ G_\emptyset = \mathsf{CF}^*_{\{e_1\}}(\mathcal{L}) &= \{\emptyset, \{e_2\}, \{e_2, f_2\}\} \end{aligned}$$

and $\mathsf{CF}^*_{\{e_2\}}(\mathcal{L}) = \{\emptyset, \{e_1\}, \{e_1, f_1\}\}$*. Both configuration structures still violate* $\mathcal{P}$*, hence none of* $\{e_1\}$ *and* $\{e_2\}$ *is a necessary cause for the violation of* $\mathcal{P}$*. Conversely, both counterfactual configuration structures are inevitably faulty with respect to* $\mathcal{P}$ *hence the faults of both components are sufficient causes for the violation of* $\mathcal{P}$*. Intuitively,* even if one of the components had behaved correctly*,* $\mathcal{P}$ *would still have been violated. Notice that the occurrence of* f_2 *in* $\mathsf{CF}^*_{\{e_1\}}(\mathcal{L})$ *is a consequence of* Exh*: once the second component has violated its specification we have no reason to assume that* f_2 *will* not *happen.*

Example 9 (Joint causation)**.** *Consider two components with specifications* $(\mathbb{E}_i, \mathcal{C}_i)$ *where* $\mathbb{E}_i = \{e_i, f_i\}$ *and* $\mathcal{C}_i = \{\emptyset, \{e_i\}\}$*,* $i = 1, 2$*, the behavioral model* $\mathcal{B} = 2^{\mathbb{E}}$ *with* $\mathbb{E} = \mathbb{E}_1 \cup \mathbb{E}_2$*, the property* $\mathcal{P} = \neg(f_1 \wedge f_2)$*, and the log* $L = (\mathbb{E}, \mathcal{L})$ *with* $\mathcal{L} = \{\emptyset, \{f_1\}, \{f_1, f_2\}\}$*, hence* $L \odot B = \mathcal{L}$*. In order to check whether* $\mathcal{X}_1 = f_1$ *is a necessary cause for the violation of* $\mathcal{P}$ *we compute*

$$\begin{aligned} R = P_{\{f_1\}} = P_{\{f_1, f_2\}} &= \{\emptyset\} \\ Z_\emptyset &= \mathcal{B} \\ \mathsf{CF}^*_{\mathcal{X}_1}(\mathcal{L}) &= \{\emptyset, \{e_1\}, \{e_2\}, \{f_2\}, \{e_1, e_2\}, \{e_1, f_2\}, \{e_2, f_2\}, \{e_1, e_2, f_2\}\} \end{aligned}$$

$\mathsf{CF}^*_{\mathcal{X}_1}(\mathcal{L}) \models \mathcal{P}$*, hence* $\mathcal{X}_1$ *is a necessary cause for the violation of* $\mathcal{P}$*. As none of* $\mathsf{CF}^*_{\mathcal{X}_1}(\mathcal{L})$ *and* $\mathsf{CF}^*_{\mathcal{X}_2}(\mathcal{L})$ *is inevitably faulty with respect to* $\mathcal{P}$*, the fault of a single component alone is not a sufficient cause for the violation of* $\mathcal{P}$*.*

Example 10 (Use of `lpc` in Definition 26)**.** *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* $\mathcal{B} = \{\emptyset, \{f_1\}, \{f_1, a\}, \{f_2\}, \{f_1, f_2\}, \{f_1, a, f_2\}\}$*, the property* $\mathcal{P} = \neg(f_1 \wedge a)$*, and the log* $L = (\mathbb{E}_1 \cup \mathbb{E}_2, \mathcal{L})$ *with* $\mathcal{L} = \{\emptyset, \{f_1\}, \{f_1, f_2\}, \{f_1, a, f_2\}\}$*: both components produce a fault event* f_i*; the conjunction of* f_1

and a violates $\mathcal{P}$. Let $\mathcal{X} = f_1 \wedge \neg a$. We have $\mathcal{C} = L \odot B = \mathcal{L}$ and

$$\begin{aligned} R = P_{\{f_1\}} = P_{\{f_1,f_2\}} &= \{\emptyset\} \\ Z_\emptyset &= \mathcal{B} \\ \mathit{CF}^*_{\mathcal{X}}(\mathcal{L}) = G_\emptyset = \mathtt{lpc}_{Z_\emptyset}(Z_\emptyset \setminus \mathcal{X}) = \mathtt{lpc}_{Z_\emptyset}\big(\{\emptyset, \{f_1, a\}, \{f_2\}, \{f_1, a, f_2\}\}\big) &= \big\{\emptyset, \{f_2\}\big\} \end{aligned}$$

Hence $\mathit{CF}^*_{\mathcal{X}}(\mathcal{L}) \models \mathcal{P}$, and $\mathcal{X}$ is a necessary cause. The configurations $\{f_1, a\}$ and $\{f_1, a, f_2\}$ have no explanation in R, in the sense that they are not reachable in the transition graph $(\mathcal{B}, \mapsto_B)$ by any path passing only through the configurations in R, and are removed by $\mathtt{lpc}_{\mathcal{C}}$. Without applying $\mathtt{lpc}_{\mathcal{C}}$, $\mathit{CF}^*_{\mathcal{X}}(\mathcal{L})$ would in addition contain $\{f_1, a\}$ and $\{f_1, a, f_2\}$ and therefore still violate $\mathcal{P}$, such that $\mathcal{X}$ would not be recognized as a necessary cause.

Example 11 (Unobservable fault 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 $\mathcal{B} = (e_1 \implies f_1) \wedge (e_3 \implies f_3)$, 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*

$$\begin{aligned} \mathcal{C} = L \odot B &= \big\{\emptyset, \{f_1\}, \{f_1, e_1\}, \{f_1, e_1, f_3\}, \{f_1, e_1, f_3, e_3\}\big\} \\ R &= \{\emptyset\} \\ Z_\emptyset &= \mathcal{B} \\ \mathit{CF}^*_{\mathcal{X}}(\mathcal{L}) = G_\emptyset &= \big\{\emptyset, \{f_3\}, \{f_3, e_3\}\big\} \models \mathcal{P} \end{aligned}$$

thus $\mathcal{X}$ is correctly recognized as a necessary cause.

Example 12 (Running example)**.** *Let us get back to our running example and formally analyze the responsibility of component C_2's failure for the violation of $\mathcal{P} = \neg e$ in*

$$\mathcal{L} = \big\{\emptyset, \{a_1\}, \{a_1, c_1\}, \{a_1, c_1, a_2\}, \{a_1, c_1, a_2, f\}, \{a_1, c_1, a_2, f, e\}\big\}$$

With $\mathcal{X} = f \vee h$ we obtain (just giving the form of Z_c and G_c in the basic case):

$$\begin{aligned} \mathcal{C} = L \odot B &= \big\{c \cup c' \mid c \in \mathcal{L} \wedge c' \in \big\{\emptyset, \{g_1\}, \{g_1, g_2\}, ...\big\}\big\} \\ R &= \big\{c \cup c' \mid c \in \big\{\emptyset, \{a_1\}, \{a_1, c_1\}, \{a_1, c_1, a_2\}\big\} \wedge c' \in \big\{\emptyset, \{g_1\}, \{g_1, g_2\}, ...\big\}\big\} \\ Y \triangleq \bigcup_{c \in \mathcal{C} \cap \mathcal{X}} P_c &= \big\{\{a_1, c_1, a_2\} \cup c' \mid c' \in \big\{\emptyset, \{g_1\}, \{g_1, g_2\}, ...\big\}\big\} \\ Z_{\{a_1,c_1,a_2\}} &= \big\{\{a_1, c_1, a_2\} \cup c' \mid c' \in \big\{\{c_2\}, \{f\}, \{g_1\}, \{c_2, h\}, \{c_2, g_1\}, \{f, g_1\}, \{g_1, g_2\}, ...\big\}\big\} \\ G_{\{a_1,c_1,a_2\}} &= \big\{\{a_1, c_1, a_2\} \cup c' \mid c' \in \big\{\{c_2\}, \{g_1\}, \{c_2, g_1\}, \{g_1, g_2\}, ...\big\}\big\} \\ \mathit{CF}^*_{\mathcal{X}}(\mathcal{L}) &= R \cup \bigcup_{z \in Y} G_z \models \mathcal{P} \end{aligned}$$

Hence, $\mathcal{X}$ is a necessary cause for the violation of $\mathcal{P}$ in L.

Example 13 (Refinement by model inclusion)**.** *Let $\sigma = (\mathbf{S}, B)$ with component specifications $S_i = (\mathbb{E}_i, \mathcal{C}_i)$ with $\mathbb{E}_1 = \{a, f_1, f\}$ and $\mathbb{E}_2 = \{b, f_2, f\}$, $\mathcal{C}_1 = \{\emptyset, \{a\}\}$, $\mathcal{C}_2 = \{\emptyset, \{b\}\}$, the behavioral model $B = (\mathbb{B}, \mathcal{B})$ with $\mathbb{B} = \mathbb{E}_1 \cup \mathbb{E}_2$ and $\mathcal{B} = \big\{\emptyset, \{a\}, \{f_1\}, \{b\}, \{f_2\}, \{a, b\}, \{f_1, b\}, \{a, f_2\}, \{f_1, f_2\}, \{f_1, f_2, f\}\big\}$, the property $\mathcal{P} = \neg(f_1 \wedge f_2)$, the log $L = (\mathbb{O}, \mathcal{L})$ with $\mathbb{O} = \mathbb{E}_1 \cup \mathbb{E}_2$ and $\mathcal{L} = \big\{\emptyset, \{f_1\}, \{f_1, f_2\}\big\}$, and $\mathcal{X} = f_1$. We have*

$$\begin{aligned} R = P_{\{f_1\}} = P_{\{f_1,f_2\}} &= \{\emptyset\} \\ Z_\emptyset &= \mathcal{B} \\ \mathit{CF}^*_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L}) = G_\emptyset &= \big\{\emptyset, \{a\}, \{b\}, \{f_2\}, \{a, b\}, \{a, f_2\}\big\} \end{aligned}$$

With $\sigma' = (\mathbf{S}, B')$ where $B' = (\mathbb{E}, \mathcal{B}')$ and $\mathcal{B}' = \big\{\emptyset, \{a\}, \{f_1\}, \{b\}, \{f_2\}, \{a, b\}, \{f_1, f_2\}\big\}$ (hence, $B' \sqsubseteq B$) we obtain $\mathit{CF}^*_{\sigma',\mathcal{X},\mathbb{O}}(\mathcal{L}) = \big\{\emptyset, \{a\}, \{b\}, \{f_2\}, \{a, b\}\big\} \subseteq \mathit{CF}^*_{\sigma,\mathcal{X},\mathbb{O}}(\mathcal{L})$.

Example 14 (Observational determinism)**.** *Consider an example of two processes P_1 and P_2 contending for a shared resource. The processes are specified by the configuration structures $(\mathbb{E}_i, \mathcal{C}_i)$ with $\mathbb{E}_i = \{req_i, w_i, p_i, v_i\}$ and $\mathcal{C}_i = \big\{\emptyset, \{req_i\}, \{req_i, w_i\}, \{req_i, p_i\}, \{req_i, w_i, p_i\}, \{req_i, p_i, v_i\}, \{req_i, w_i, p_i, v_i\}\big\}$, $i \in [2]$ — that is, P_i may request the resource with req_i, wait (w_i), take the resource (p_i), and free it again (v_i). The behavioral model is $B = (\mathbb{B}, \mathcal{B})$ with $\mathbb{B} = \bigcup_{i\in[n]} \mathbb{E}_i$ and*

$$\mathcal{B} = \{c_1 \cup c_2 \mid c_1 \in \mathcal{C}_1 \wedge c_2 \in \mathcal{C}_2 \wedge \bigwedge_{i=1,2} (w_i \in c_i \implies p_{3-i} \in c_{3-i})\},$$

meaning that process i may be waiting (w_i) only if the other process has taken the resource before, the log system $\mathbf{L} = \{L_1, L_2\}$ with $L_i = (\mathbb{O}, \mathcal{L}_i)$ with $\mathbb{O} = \mathbb{E}_2$ — that is, only the behavior of P_2 is observable, whereas that of P_1 is meant to remain secret —, $\mathcal{L}_1 = \big\{\emptyset, \{req_2\}, \{req_2, w_2\}, \{req_2, w_2, p_2\}, \{req_2, w_2, p_2, v_2\}\big\}$ and $\mathcal{L}_2 = \big\{\emptyset, \{req_2\}, \{req_2, p_2\}, \{req_2, p_2, v_2\}\big\}$ and the hyperproperty $\mathbf{P} = \{\mathcal{P}_1, \mathcal{P}_2\}$ with

$$\mathcal{P}_1 = \big\{c_1 \cup c_2 \mid c_1 \in \mathcal{C}_1 \wedge c_2 \in \big\{\emptyset, \{req_2\}, \{req_2, w_2\}, \{req_2, w_2, p_2\}, \{req_2, w_2, p_2, v_2\}\big\} \wedge c_1 \cup c_2 \in \mathcal{B}\big\}$$
$$\mathcal{P}_2 = \big\{c_1 \cup c_2 \mid c_1 \in \mathcal{C}_1 \wedge c_2 \in \big\{\emptyset, \{req_2\}, \{req_2, p_2\}, \{req_2, p_2, v_2\}\big\} \wedge c_1 \cup c_2 \in \mathcal{B}\big\}$$

requiring observational determinism (OD) [39, 7]. Intuitively, the observed behavior of P_2 must be deterministic (either including or excluding w_2 in the maximal trace), and not give away any information about the state of P_1. OD is a 2-safety hyperproperty, that is, it may be invalidated on a set of two traces exhibiting different behaviors of P_2. Clearly, $L_1 \cup L_2 \not\models \mathbf{P}$. Let us check that $\mathcal{X} = w_2$ is a necessary cause for the violation of $\mathbf{P}$. We have $\mathsf{CF}^*_{\mathcal{X}}(\mathbf{L}) = \big\{\emptyset, \{p_1\}, \{p_1, v_1\}, \{p_1, v_1, p_2\}, \{p_1, v_1, p_2, v_2\}\big\} \models \mathbf{P}$, hence $\mathcal{X} = w_2$ is a necessary cause for the violation of $\mathbf{P}$.

6 Conclusion

In this paper we have stated a set of formal requirements that should — all or in part, depending on the intended application — be ensured by "well-formed" counterfactual builders in order to meaningfully reason about causality. Intuitively, a counterfactual builder is a function $\mathsf{CF}_{\sigma,\mathcal{X},\mathbb{O}}$, parameterized by a system specification, featuring a tuple of components, a set $\mathcal{X}$ of configurations to be checked for causality and a set of observables, that takes as input a log L and returns a configuration structure modeling the alternative executions, had the configurations in $\mathcal{X}$ not occurred in the execution observed in L. We have discussed the properties of well-formed counterfactual builders, and considered the special case where $\mathcal{X}$ represents a set of component faults, yielding a fault ascription analysis. In the second part of the paper we have proposed a concrete well-formed counterfactual builder CF^*, and we have shown it to be monotonic under two notions of refinement. A well-understood behavior under refinement is crucial for reasoning about causation in a complex system, via an abstraction of said system.

The importance of formal requirements for the design of counterfactual analyses was confirmed during our writing of this article: previous versions of Definition 26 — that looked "reasonable" and met our intuition on the catalog of examples — have been revisited several times after adding further expected properties that turned out to be violated. The instantiated counterfactual function we have proposed is not intended to be the last word, but rather the first step towards a set of counterfactual analyses whose development will be triggered by formally stated requirements. For instance, for some applications it would make sense to develop an instance of CF satisfying a variant of well-formedness where our requirement Exh is replaced with minimal distance between the counterfactual and the actual executions, reflecting Lewis' "closest worlds" semantics [28].

The framework for causality analysis and fault ascription presented in this paper suffers from several limitations, which need to be lifted in future work. First, our framework provides an analysis of necessary causality in the case of hyperproperties, but most interesting results obtain only for subset-closed hyperproperties. Obtaining similar results for non subset-closed hyperproperties is an item for further study. Second, our notion of sufficient causality is limited to properties. Extending it to hyperproperties is possible but only for weaker forms of inevitability. It remains to be seen how to extend it to hyperproperties while retaining the current intuition of inevitability. Third, our framework currently deals only with unlabelled configuration structures. One can argue that in practice one observes mostly event labels. Also, usual notions of process equivalences such as similarities and bisimilarities are defined

for labelled structures. Extending our framework to labelled configuration structures would allow us to study the relations between causality analysis and non-trivial process equivalences.

More work is required to turn our framework into effective tools for causality analysis. As Definition 26 is constructive, it could be implemented *as is* for the finite case. In order to account for potentially infinite behaviors and avoid the inefficiency of a naive implementation, our next steps will be to study symbolic representations of counterfactual builders. In particular, we are interested in identifying models of computation on which the counterfactuals can be computed efficiently. Additionally, more attention needs to be paid to identify minimal causes, drawing in particular on results related to explanations and counterexamples in model checking.

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] A. Balke and J. Pearl. Probabilistic evaluation of counterfactual queries. In B. Hayes-Roth and R.E. Korf, editors, *AAAI*, pages 230–237. AAAI Press / The MIT Press, 1994.

[3] A. Beer, S. Heidinger, U. Kühne, F. Leitner-Fischer, and S. Leue. Symbolic causality checking using bounded model checking. In B. Fischer and J. Geldenhuys, editors, *Model Checking Software - 22nd International Symposium, SPIN 2015, Stellenbosch, South Africa, August 24-26, 2015, Proceedings*, volume 9232 of *LNCS*, pages 203–221. Springer, 2015.

[4] 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.

[5] 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.

[6] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer, 2nd edition, 2008.

[7] M. R. Clarkson and F. B. Schneider. Hyperproperties. *Journal of Computer Security*, 18(6):1157–1210, 2010.

[8] A. Datta, D. Garg, D.K. Kaynar, D. Sharma, and A. Sinha. Program actions as actual causes: A building block for accountability. In C. Fournet, M.W. Hicks, and L. Viganò, editors, *IEEE 28th Computer Security Foundations Symposium, CSF 2015, Verona, Italy, 13-17 July, 2015*, pages 261–275. IEEE Computer Society, 2015.

[9] S.K. Dyrkolbotn. On preemption and overdetermination in formal theories of causality. In A. Groce and S. Leue, editors, *Proc. 2nd International Workshop on Causal Reasoning for Embedded and safety-critical Systems Technologies, CREST@ETAPS 2017, Uppsala, Sweden, 2017.*, volume 259 of *EPTCS*, pages 1–15, 2017.

[10] J. Esparza and S. Römer. An unfolding algorithm for synchronous products of transition systems. In *Concurrency Theory, 10th International Conference, CONCUR '99*, volume 1664 of *Lecture Notes in Computer Science*. Springer, 1999.

[11] C. Glymour, D. Danks, B. Glymour, F. Eberhardt, J. Ramsey, R. Scheines, P. Spirtes, C.M. Teng, and J. Zhang. Actual causation: a stone soup essay. *Synthese*, 175(2):169–192, 2010.

[12] G. Gössler and L. Aştefănoaei. Blaming in component-based real-time systems. In *EMSOFT'14*, pages 7:1–7:10. ACM, 2014.

[13] G. Gössler and D. Le Métayer. A general framework for blaming in component-based systems. *Science of Computer Programming*, 113(3):223–235, 2015.

[14] G. Gössler, O. Sokolsky, , and J.-B. Stefani. Counterfactual causality from first principles? In A. Groce and S. Leue, editors, Proceedings 2nd International Workshop on *Causal Reasoning for Embedded and safety-critical Systems Technologies,* Uppsala, Sweden, 29th April 2017, volume 259 of *Electronic Proceedings in Theoretical Computer Science*, pages 47–53. Open Publishing Association, 2017.

[15] G. Gössler and J.B. Stefani. Fault ascription in concurrent systems. In P. Ganty and M. Loreti, editors, *Proc. Trustworthy Global Computing - 10th International Symposium, TGC 2015*, volume 9533 of *LNCS*. Springer, 2016.

[16] C.W.J. Granger. Testing for causality. *Journal of Economic Dynamics and Control*, 2:329 – 352, 1980.

[17] S. Haar and E. Fabre. Diagnosis with petri net unfoldings. In *Control of Discrete-Event Systems*, volume 433 of *LNCIS*, chapter 15. Springer, 2013.

[18] J. Y. Halpern and C. Hitchcock. Graded causation and defaults. *CoRR*, abs/1309.1226, 2013.

[19] J.Y. Halpern. A modification of the halpern-pearl definition of causality. In Q. Yang and M. Wooldridge, editors, *Proc. Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 3022–3033. AAAI Press, 2015.

[20] J.Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach. part I: Causes. *British Journal for the Philosophy of Science*, 56(4):843–887, 2005.

[21] M. Hopkins and J. Pearl. Causality and counterfactuals in the situation calculus. *J. Log. Comput.*, 17(5), 2007.

[22] 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.

[23] B. Jacobs, A. Kissinger, and F. Zanasi. Causal inference by string diagram surgery. In M. Boja'nczyk and A. Simpson, editors, *Proc. Foundations of Software Science and Computation Structures - 22nd International Conference, FOSSACS 2019*, volume 11425 of *LNCS*, pages 313–329. Springer, 2019.

[24] S. Kleinberg. *Causality, Probability, and Time.* Cambridge, 2012.

[25] M. Kuntz, F. Leitner-Fischer, and S. Leue. From probabilistic counterexamples via causality to fault trees. In F. Flammini, S. Bologna, and V. Vittorini, editors, *SAFECOMP*, volume 6894 of *LNCS*, pages 71–84. Springer, 2011.

[26] J. Laurent, J. Yang, and W. Fontana. Counterfactual resimulation for causal analysis of rule-based models. In *Proc. 27th International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 1882–1890, 7 2018.

[27] F. Leitner-Fischer and S. Leue. Probabilistic fault tree synthesis using causality computation. *IJCCBS*, 4(2):119–143, 2013.

[28] D. Lewis. *Counterfactuals.* Blackwell, 1973.

[29] J. McCarthy. Situations, actions, and causal laws. Technical report, Stanford University, CA, Dept. of Computer Science, 1963.

[30] J. Pearl. *Causality: Models, Reasoning, and Inference.* Cambridge University Press, 2000.

[31] J. Pearl. Theoretical impediments to machine learning with seven sparks from the causal revolution. In *Proc. Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18)*, pages 3–3. ACM, 2018.

[32] R. Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.

[33] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Failure diagnosis using discrete-event models. *IEEE Trans. Contr. Sys. Techn.*, 4(2):105–124, 1996.

[34] R. Stalnaker. A theory of conditionals. *Studies in Logical Theory*, pages 98–112, 1968.

[35] L. Travé-Massuyès. Bridging control and artificial intelligence theories for diagnosis: A survey. *Engineering Applications of Artificial Intelligence*, 27, 2014.

[36] R J. van Glabbeek and G. D. Plotkin. Configuration structures, event structures and petri nets. *Theoretical Computer Science*, 410(41), 2009.

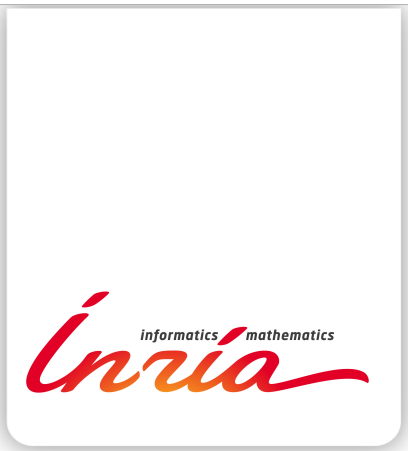
[37] G. Winskel and M. Nielsen. Models for concurrency. In *Handbook of Logic in Computer science*, volume 4. Clarendon Press, 1995.

[38] J. Zaytoon and S. Lafortune. Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37(2), 2013.

[39] S. Zdancewic and A.C. Myers. Observational determinism for concurrent program security. In *16th IEEE Computer Security Foundations Workshop (CSFW-16 2003)*, page 29. IEEE Computer Society, 2003.

A Glossary of Notations and Definitions

notation	definition or meaning	defined where
$[n]$	$\{1,\dots,n\}$	Section 2
$\bigcup S$	$\bigcup_{s\in S} s$	Section 2
$\mathbb{B}$, $\mathbb{E}$, ...	sets of events	
$\mathcal{C}$	set of configurations (sets of events)	
C	configuration structure (CS)	Definition 1
$\rightarrow_{\mathcal{C}}$, $\mapsto_{\mathcal{C}}$	step (resp. atomic step) transition relation	Definition 2
$\parallel$, $\cap$, $\subseteq$, max, $\cdot_{\downarrow\cdot}$, $\cdot^{\uparrow\cdot}$, $\leq$, $>$	operations on CS	Definition 3
$\mathcal{P}$, $\mathbf{P}$	property, hyperproperty	Definition 4
$\models$	satisfaction	Definition 5
Tr	traces of a CS	Definition 6
$\mathtt{CS}$	CS of a trace	Definition 7
$\bigcirc_{\mathcal{B}}\mathcal{C}$	consistency: $\exists c\in\mathcal{B} : \bigcup\mathcal{C}\subseteq c$	Definition 12
$L\odot B$	filtering: $\left\{c\in\mathcal{L}^{\uparrow\mathbb{B}}\cap\mathcal{B} \mid \bigcirc_{\mathcal{B}}(\mathcal{L}\cup\{c\})\right\}$	Definition 15
$\mathtt{pc}_{\mathcal{C}'}(\mathcal{C})$	pred. closed: $\forall c\in\mathcal{C} : c\in\min\mathcal{C}'\vee\max\{c'\in\mathcal{C}' \mid c'\subsetneq c\}\cap\mathcal{C}\neq\emptyset$	Definition 16
$\mathtt{lpc}_{\mathcal{C}'}(\mathcal{C})$	largest pred.-closed subset of $\mathcal{C}$ w.r.t. $\mathcal{C}'$	Definition 16
$C'\sqsubseteq C$	prefix inclusion: $C'\subseteq C\wedge\mathtt{pc}_{\mathcal{C}}(\mathcal{C}')$	Definition 17
$\mathtt{pre}_{\mathcal{C}}(\mathcal{S})$	immediate predecessors: $\bigcup_{c\in\mathcal{S}}\max\{c'\in\mathcal{C}\setminus\mathcal{S} \mid c'\subseteq c\}$	Definition 18
$\mathtt{inv}_{\mathbf{S}}(\mathcal{C},\mathcal{C}')$	invariance: $\bigwedge_{i\in I}\left(\mathcal{C}_{\downarrow\mathbb{E}_i}\subseteq\mathcal{C}_i \implies \mathcal{C}'_{\downarrow\mathbb{E}_i}\subseteq\mathcal{C}_i\right)$	Definition 19
$\mathtt{inv}^{+}_{\mathbf{S}}(L,\mathcal{C}')$	possible invariance: $\exists\mathcal{C}\subseteq B : \mathcal{C}_{\downarrow\mathbb{0}}=\mathcal{L}\wedge inv_{\mathbf{S}}(\mathcal{C},\mathcal{C}')$	Definition 19
Snd	soundness of counterfactual builder	Requirement 1
Ci	independence correctness of counterfactual builder	Requirement 2
Cd	dependence correctness of counterfactual builder	Requirement 3
Cinv	correctness invariance of counterfactual builder	Requirement 4
Obs	observation anti-monotony of counterfactual builder	Requirement 5
Inc	incrementality of counterfactual builder	Requirement 6
Exh	exhaustiveness of counterfactual builder	Requirement 7
$C'\preceq C$	alphabet inclusion	Definition 25
$\mathsf{CF}^{*}_{\sigma,\mathcal{X},\mathbb{0}}$	WFC counterfactual builder	Definition 26



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399