

# Causality and Decision-making: A Logical Framework for Systems and Security Modelling

Pinaki Chakraborty<sup>1\*</sup>, Tristan Caulfield<sup>1\*</sup>, and David Pym<sup>1,2\*</sup>

<sup>1</sup> University College London, England, UK {pinaki.chakraborty.22@ucl.ac.uk, t.caulfield, d.pym@ucl.ac.uk}@ucl.ac.uk

<sup>2</sup> Institute of Philosophy, University of London  
England, UK david.pym@sas.ac.uk

\* corresponding author

**Abstract.** Causal reasoning is essential for understanding decision-making about the behaviour of complex ‘ecosystems’ of systems that underpin modern society, with security — including issues around correctness, safety, resilience, etc. — typically providing critical examples. We present a theory of strategic reasoning about system modelling based on minimal structural assumptions and employing the methods of transition systems, supported by a modal logic of system states in the tradition of van Benthem, Hennessy, and Milner, and validated through equivalence theorems. Our framework introduces an intervention operator and a separating conjunction to capture actual causal relationships between component systems of the ecosystem, aligning naturally with Halpern and Pearl’s counterfactual approach based on Structural Causal Models. We illustrate the applicability through examples of decision-making about microservices in distributed systems. We discuss localized decision-making through a separating conjunction. This work unifies a formal, minimalistic notion of system behaviour with a Halpern-Pearl-compatible theory of counterfactual reasoning, providing a logical foundation for studying decision making about causality in complex interacting systems.

**Keywords:** Logic · Transition systems · Decision-making · Strategic reasoning · System models · Causality · Influence · Interface · Separation · Security · Microservices

## 1 Introduction

Causal modelling is a multidisciplinary field spanning computer science, econometrics, epidemiology, philosophy, and statistics, providing a robust framework for understanding and reasoning about cause-and-effect relationships in systems. Such reasoning is of particular significance in things like root-cause analysis and strategy formulation for security.

One influential approach to understanding causality is counterfactual analysis [26], which stipulates that an event qualifies as a cause if, counterfactually, its absence would prevent the effect from occurring. In many counterfactual

theories of causation, directed graphs have emerged as a powerful tool for representing causal relationships, as exemplified by seminal contributions from Judea Pearl [30], Hitchcock [21], and Spirtes et al. [38]. A formal representation of these causal relationships is provided by a set of equations known as *structural equations (SE)*, which explicitly encode how each variable depends on its causal predecessors. These can be visualised as directed acyclic graphs in which vertices correspond to variables, while directed edges signify direct causal dependencies. Building on this foundation, Halpern and Pearl [20] use structural equations to define a rigorous notion of actual causation, capturing conditions under which specific events can be identified as causes of given outcomes. Although actual causality — the objective mechanism linking causes and effects — is distinct from our knowledge of it, which is built incrementally through observation, intervention, and counterfactual reasoning, both perspectives are deeply intertwined. This interplay motivates our system modelling approach, in which we formalize causation through precise structural and dynamic relations aligning with Halpern and Pearl’s axioms of actual causation.

In particular, our work leverages Pearl’s Structural Equations approach [20,31] (a detailed discussion is deferred to 4) and its subsequent extension by Halpern [18], integrating modal logic to enable rigorous causal analysis across applications, such as mitigating risks and analysing incidents in complex systems like cyber infrastructure, where interactions among software, hardware, and human actions drive outcomes.

It should be noted that we differ from the setting of do-calculus [30] in the sense that stochastic interpretation of variables, which is essential to ‘type causality’ (see Section 4.1), are not involved. Also, we introduce a uniform syntactic treatment of interventions in the logical language itself unlike the setting of Pearl who formalised interventions on a semantic level. And lastly unlike the setting of Structural Equations in which interventions only change the value of a variable, we allow interventions to alter the structural equation relation.

It is also well established that game-based strategic reasoning about systems can be modelled using the formal technology of transition systems and, consequently, can employ the methods of process algebra — for example, see [39] for an elegant exposition of this relationship, [40] for a reflective overview, and [41] for a discussion of ‘dynamic agent organizations’, noting that ‘agent organizations’ can be described algebraically as systems of process terms — allowing access to the expressivity required to capture decentralized/distributed and concurrent systems. These approaches are well adapted to supporting decision-making about such systems because they naturally support a rich *logical* theory that is tightly integrated with the structure of processes. Here we employ this approach in the setting of a minimalistic, behaviour-based model to discuss actual causality in ‘ecosystems’ of interacting systems (see also [10,17]), providing tools for reasoning — that is, decision-making — about causation and influence between system configurations.

We illustrate our approach with systems’ security examples based on the problems involving root-cause analysis — see, for example, [22,27,42,32] — that

are concerned with mitigating faults arising within distributed microservice architectures in large-scale software systems (cf. [1]). (See also [2] for a sketch of a different approach.)

We also draw inspiration from cybernetics — particularly Simon’s work [37,36] — which emphasizes that simple, local rules and interactions can govern complex system behaviours and dynamics. As Simon notes in [37], ‘All behaviour involves conscious or unconscious selection of particular actions out of all those which are physically possible to the actor and to those persons over whom he exercises influence and authority.’ This observation highlights the pivotal role of ‘influence’ in propagating effects throughout a system. By adopting the term ‘influence’ to describe the rules governing our system’s components, we align with Simon’s cybernetic tradition, viewing systems as entities shaped by local interactions.

At its core, our approach treats a system as a set of vertices — each representing a component with observable behaviours — whose dynamics emerges solely from a set of rules called influence. This echoes the cybernetic insight that local interactions drive broader system dynamics, and also provides a robust platform for exploring (actual) causality in interactive environments.

Section 1 outlines the scope and necessity of causal reasoning in system modelling. Section 2 introduces a minimalist approach to system modelling, followed by Section 3, which develops the logical framework used to describe system models. In Section 4, we demonstrate how this logic formalizes actual causation and captures causal structures. Section 5 explores a substantial example of how we can model decision-making about the dependencies between microservices in distributed systems. Section 6 discusses the logical metatheory of our framework, showing how bisimulation characterizes equivalence. Finally, Section 7 situates our work within the broader landscape of causal modelling and strategic decision-making.

## 2 The system modelling framework

In this section, we adopt a deliberately minimalist view: instead of tabulating every internal state, we specify a component only by the interactions (called its behaviour) an external observer can witness and the concrete influence those interactions have on other components. This choice is not superficial in that it draws a line between state (unobservable, intensional details) and behaviour (observable, extensional facts).

Existing literature offers various frameworks for modelling system interactions, particularly in distributed systems — for example, [10,3,11,13,35], with extensive relevant bibliographies, are pertinent here. Our work builds on this foundation by drawing inspiration from a recent abstraction [17] that adopts a behaviour-centric perspective, though we incorporate dynamic aspects that extend beyond their static view. Our terms *component*, *influence*, *configuration* are inspired from the foundational work by Winskel on event structures [43] and by Simon [37].

Formally, let  $\mathcal{C}$  be the set of components and  $\mathcal{B}$  the set of all possible behaviours. A function  $\mathbb{B} : \mathcal{C} \rightarrow 2^{\mathcal{B}}$  assigns to each component  $c \in \mathcal{C}$  its set of allowable behaviours,  $\mathbb{B}(c)$ . The complete state of a system is described by a configuration that specifies each component's behaviour at a particular instant.

**Definition 1 (Configuration).** *Let  $\mathcal{C}$  be a set of components, and let  $\mathbb{B} : \mathcal{C} \rightarrow 2^{\mathcal{B}}$  assign to each component  $c \in \mathcal{C}$  a set  $\mathbb{B}(c)$  of allowable behaviours.*

*A configuration over  $\mathcal{C}$  is a total function  $f : \mathcal{C} \rightarrow \mathcal{B}$  such that  $f(c) \in \mathbb{B}(c)$  for all  $c \in \mathcal{C}$ . The set of all configurations over  $\mathcal{C}$  is denoted by  $F_{\mathcal{C}}$ . When  $\mathcal{C}$  is understood from the context, we write  $F$  for brevity.  $\square$*

To model how components in a system evolve, we introduce influence rules that specify how component behaviours are determined — formally, functions that, given the current behaviour of a component and the behaviours of selected components in the system, determine its next behaviour.

**Definition 2 (Influence rules and contexts).** *Let  $\mathcal{C}$  be the global set of components. For each component  $c \in \mathcal{C}$ , let  $\text{Inf}(c) \subseteq \mathcal{C} \setminus \{c\}$  be the influence context for  $c$ .  $\text{Inf}(c)$  is the subset of components whose behaviours are relevant for determining the behaviour of  $c$ . An influence rule for  $c$  is then a function  $\mathcal{I}_c : \mathbb{B}(c) \times \prod_{d \in \text{Inf}(c)} \mathbb{B}(d) \rightarrow \mathbb{B}(c)$ , specifying how the behaviour of  $c$  evolves from its current behaviour and the behaviours of components in its influence context. The family of all such functions relative to a set of components  $\mathcal{C}$  is called  $\mathcal{I}_{\mathcal{C}}$ .  $\square$*

We often omit the subscript when the referenced set of components is clear from the context. In our framework, a system is defined by its space of possible configurations, the transition dynamics governing their evolution, and the propositions that hold in each configuration. This view is captured by a system model, which encapsulates the set of configurations, the transitions induced by influence rules, and the mapping of configurations to the atomic propositions that hold within them. First, we define the transition relation which is induced by a family of influence rules.

**Definition 3 (Transition relation).** *Given a component set  $\mathcal{C}$ , a behaviour mapping  $\mathbb{B}$ , and a family of influence rules  $\mathcal{I} = \{\mathcal{I}_c\}_{c \in \mathcal{C}}$  where each  $\mathcal{I}_c : \mathbb{B}(c) \times \prod_{d \in \text{Inf}(c)} \mathbb{B}(d) \rightarrow \mathbb{B}(c)$ , the transition relation  $\Delta_{\mathcal{I}} \subseteq F \times F$  is defined as  $(f, f') \in \Delta_{\mathcal{I}}$  iff there exists  $c \in \mathcal{C}$  such that  $f'(c) = \mathcal{I}_c(f(c), (f(d))_{d \in \text{Inf}(c)})$ , and,  $f'(d) = f(d)$  for all  $d \neq c$ . That is,  $f$  transitions to  $f'$  when exactly one component  $c$  updates its behaviour according to its local influence rule, while all other components remain unchanged.  $\square$*

The definition of a system model follows:

**Definition 4 (System model).** *For each  $c \in \mathcal{C}$ , define  $\Delta_c = \{(f, f') \in F \times F \text{ such that } f'(c) = \mathcal{I}_c(f(c), (f(d))_{d \in \text{Inf}(c)})\}$ . Here,  $\forall d \neq c, f'(d) = f(d)$ . Let  $\Delta_{\mathcal{I}} = \bigcup_{c \in \mathcal{C}} \Delta_c$ . A system model  $\mathcal{M}$  is a tuple  $(\mathcal{C}, \mathcal{B}, \mathcal{I}, F, \Delta_{\mathcal{I}}, \Gamma)$ , where  $F$  is the set of all possible configurations of the system given a set of components  $\mathcal{C}$ , a set of possible behaviours  $\mathcal{B}$ , and a family of rules  $\mathcal{I}$  govern the behaviour change*

of the components.  $\Gamma : \mathcal{P} \rightarrow 2^F$  is a valuation function that assigns a subset of the configurations to each atomic proposition from the set of atomic propositions (from a set of atomic propositions  $\mathcal{P}$ ).

For brevity, we often omit the full notation, and write  $\mathcal{M} = (F, \Delta_{\mathcal{I}}, \Gamma)$ . The transition relation  $\Delta_{\mathcal{I}}$  may be viewed as the edge relation of a directed graph over the configuration space  $F$ , where configurations are vertices and transitions form edges.  $\square$

*Example 1.* Let  $\mathcal{C} = \{c_1, c_2, c_3\}$ ,  $\mathcal{B} = \{b_{11}, b_{12}, b_{13}, b_{21}, b_{22}, b_{31}\}$ , and an assignment of behaviours be  $\mathbb{B}(c_1) = \{b_{11}, b_{12}, b_{13}\}$ ,  $\mathbb{B}(c_2) = \{b_{21}, b_{22}\}$ , and  $\mathbb{B}(c_3) = \{b_{31}\}$ . A configuration  $f_1$  in this context is  $\{(c_1, b_{11}), (c_3, b_{31})\}$ . One possible choice of the influence rules is  $\mathcal{I}_{c_1}(b_{11}) = b_{12}$ ,  $\mathcal{I}_{c_1}(b_{12}) = b_{13}$ ,  $\mathcal{I}_{c_1}(b_{13}) = b_{11}$ , and  $\mathcal{I}_{c_2}(b_{21}) = b_{22}$ ,  $\mathcal{I}_{c_2}(b_{22}) = b_{21}$ . Another configuration  $f_2$  which is ‘reachable’ using these rules can be  $\{(c_1, b_{12}), (c_3, b_{31})\}$ , and so on.  $\square$

To analyse subsystems within a system model, we establish conditions under which a system can be meaningfully decomposed. This requires identifying an interface that mediate dependencies between subsystems. In order to define subsystems we begin with a partial configuration, which is the assignment of behaviours to only a subset of the components that constitute a full configuration.

**Definition 5 (Partial configuration).** Let  $\mathcal{C}' \subseteq \mathcal{C}$  be a subset of components. A partial configuration over  $\mathcal{C}'$  is a function  $f' : \mathcal{C}' \rightarrow \bigcup_{c \in \mathcal{C}'} \mathbb{B}(c)$ , where  $\mathbb{B}(c)$  is the set of possible behaviours for component  $c$ . While a full configuration  $f \in F$  assigns behaviours to all components in  $\mathcal{C}$ , a partial configuration  $f'$  assigns behaviours only to a chosen subset  $\mathcal{C}'$ , leaving the rest undefined. Given a full configuration  $f \in F$ , its restriction to  $\mathcal{C}'$  is denoted by  $f \upharpoonright_{\mathcal{C}'}$ .  $\square$

The following defines an interface among the components by imposing constraints on the influence relationships among components:

**Definition 6 (Interface-admitting system model).** A system model  $\mathcal{M} = (F, \Delta_{\mathcal{I}}, \Gamma)$  over a global component set  $\mathcal{C}$  is said to admit an **interface** if there exist subsets  $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathcal{C}$  satisfying  $\mathcal{C}_1 \cup \mathcal{C}_2 = \mathcal{C}$  such that for every component  $c \in \mathcal{C}_i \setminus (\mathcal{C}_1 \cap \mathcal{C}_2)$  (with  $i \in \{1, 2\}$ ), the influence context  $\text{Inf}(c) \subseteq \mathcal{C}_i$ , and for every component  $c \in (\mathcal{C}_1 \cap \mathcal{C}_2)$ , the influence context  $\text{Inf}(c) \subseteq (\mathcal{C}_1 \cap \mathcal{C}_2)$ . We say that  $\mathcal{M}$  is interface-admitting if such subsets  $\mathcal{C}_1$  and  $\mathcal{C}_2$  exist with interface  $(\mathcal{C}_1 \cap \mathcal{C}_2)$ . The two conditions above ensure that non-interface components depend only on other components within their own partition (including the interface), and that interface components depend only on components in the interface.  $\square$

*Remark 1.* For each  $c \in \mathcal{C}_i$ , its local influence rule is  $\mathcal{I}_c^i : \mathbb{B}(c) \times \prod_{d \in \text{Inf}(c)} \mathbb{B}(d) \rightarrow \mathbb{B}(c)$ . These rules are consistent with the constraints of influence locality specified above. The original influence rule  $\mathcal{I}_c$  is recoverable from the local rules. Specifically, for any  $b \in \mathbb{B}(c)$  and any behaviour assignment  $(b_d)_{d \in \text{Inf}(c)}$ , it holds that  $\mathcal{I}_c(b, (b_d)_{d \in \text{Inf}(c)}) = \mathcal{I}_c^i(b, (b_d)_{d \in \text{Inf}(c)})$ .  $\square$

*Example 2 (Interface).* Let  $\mathcal{C} = \{c_1, c_2, c_3\}$  and  $\mathbb{B}(c_1) = \{b_{11}, b_{12}, b_{13}\}$ ,  $\mathbb{B}(c_2) = \{b_{21}, b_{22}\}$ , and,  $\mathbb{B}(c_3) = \{b_{31}\}$ . The influence contexts are  $E(c_1) = \emptyset, E(c_2) = \{c_1\}, E(c_3) = \emptyset$ . The influence rules are,  $\mathcal{I}_{c_1}(b_{11}) = b_{12}, \mathcal{I}_{c_1}(b_{12}) = b_{13}, \mathcal{I}_{c_1}(b_{13}) = b_{11}, \mathcal{I}_{c_2}(b_{21}, b_{12}) = b_{22}, \mathcal{I}_{c_2}(b_{21}, \_) = b_{21}, \mathcal{I}_{c_2}(b_{22}, \_) = b_{22}, \mathcal{I}_{c_3}(b_{31}) = b_{31}$ . Thus  $c_1$  cycles its behaviours autonomously,  $c_2$  switches from  $b_{21}$  to the  $b_{22}$  *only when*  $c_1$  behaves  $b_{12}$ , and  $c_3$  is inert. Take the partition  $C_1 = \{c_1, c_2\}, C_2 = \{c_2, c_3\}$ , and,  $I = C_1 \cap C_2 = \{c_2\}$ . The locality conditions of Definition 6 hold, and thus  $\{c_2\}$  constitutes an interface.  $\square$

A conjugate decomposition ensures that an interface-admitting system can be partitioned into subsystems in a way that preserves the global system behaviour through consistent interactions at the interface, with local transition dynamics faithfully reflecting the overall system evolution.

**Definition 7 (Conjugate decomposition).** *Let  $\mathcal{M} = (F, \Delta_{\mathcal{I}}, \Gamma)$  be an interface-admitting system model over a global component set  $\mathcal{C}$ . In particular, let  $C_1 \cap C_2$  ( $C_1, C_2 \subseteq \mathcal{C}$ ) form an interface in  $\mathcal{M}$ . A conjugate decomposition of  $\mathcal{M}$  with respect to the interface  $I = C_1 \cap C_2$  is a pair of partial system models  $(F_1, \Delta_{\mathcal{I}_1}, \Gamma_1)$  over  $C_1$ , and  $(F_2, \Delta_{\mathcal{I}_2}, \Gamma_2)$  over  $C_2$ , such that, the following conditions are satisfied:*

1.  $F_1$  and  $F_2$  are the sets of partial configurations over  $C_1$  and  $C_2$ , respectively, with  $\Gamma_1$  and  $\Gamma_2$  being the restrictions of the global valuation  $\Gamma$  to  $F_1$  and  $F_2$ .
2. The global transition relation  $\Delta_{\mathcal{I}}$  is recoverable from the partial transition relations  $\Delta_{\mathcal{I}_1}$  and  $\Delta_{\mathcal{I}_2}$ ; that is, for any full configuration  $f \in F$  with restrictions  $f \upharpoonright_{C_1} = f_1$  and  $f \upharpoonright_{C_2} = f_2$ , if  $f \Delta_{\mathcal{I}} f'$ , the corresponding restrictions satisfy  $f_1 \Delta_{\mathcal{I}_1} f'_1$  and  $f_2 \Delta_{\mathcal{I}_2} f'_2$ , and on the interface,  $I$ , the partial configurations agree.  $\square$

A system can intervene on by triggering changes in how its component's behaviours are altered. This is formalized via interventions which are one-time modifications applied to a specific subset of components replacing their existing influence rules with new ones. After the intervention, the system continues to operate under the new rules.

**Definition 8 (Intervention).** *Consider a system model  $\mathcal{M} = (F, \Delta_{\mathcal{I}}, \Gamma)$ . An intervention  $\theta_{C'}$  consists of a pair  $(C', \mathcal{I}'_{C'})$ , where  $C' \subseteq \mathcal{C}$  is the subset of components targeted by the intervention, and  $\mathcal{I}'_{C'} = \{\mathcal{I}'_c\}_{c \in C'}$  is a new set of influence rules for the components in  $C'$ . Each rule  $\mathcal{I}'_c : \mathbb{B}(c) \times \prod_{d \in \text{Inf}(c)} \mathbb{B}(d) \rightarrow \mathbb{B}(c)$  respects the original influence context  $\text{Inf}(c)$ .*

*An intervention is atomic and one-time: it modifies the influence rules instantaneously and irreversibly at the point of application, after which the system evolves using the new rules. When the intervention  $\theta$  is applied, the system model transforms into  $\mathcal{M}_\theta = (F, \Delta_{\mathcal{I}}^\theta, \Gamma)$ , where the modified influence rules are given by,  $\mathcal{I}'_c = \mathcal{I}'_c$  if  $c \in C'$ ; otherwise it does not change. The transition relation  $\Delta_{\mathcal{I}}^\theta$  is the smallest relation closed under these revised rules, while  $F$  and  $\Gamma$  remain unchanged.  $\square$*

*Example 3 (Intervention).* Let the component set be  $\mathcal{C} = \{c_1, c_2, c_3\}$  and the behaviour domains be  $\mathbb{B}(c_1) = \{b_{11}, b_{12}, b_{13}\}$ ,  $\mathbb{B}(c_2) = \{b_{21}, b_{22}\}$ , and  $\mathbb{B}(c_3) = \{b_{31}\}$ . The influence contexts be,  $\text{Inf}(c_1) = \emptyset$ ,  $\text{Inf}(c_2) = \{c_1\}$ , and,  $\text{Inf}(c_3) = \emptyset$ . The influence rules before intervention are,  $\mathcal{I}_{c_1}(b_{11}) = b_{12}$ ,  $\mathcal{I}_{c_1}(b_{12}) = b_{13}$ ,  $\mathcal{I}_{c_1}(b_{13}) = b_{11}$ ,  $\mathcal{I}_{c_2}(b_{21}, b_{12}) = b_{22}$ ,  $\mathcal{I}_{c_2}(b_{21}, \_) = b_{21}$ ,  $\mathcal{I}_{c_2}(b_{22}, \_) = b_{22}$ ,  $\mathcal{I}_{c_3}(b_{31}) = b_{31}$ . Thus  $c_1$  cycles through three states independently,  $c_2$  switches from  $b_{21}$  to  $b_{22}$  only if  $c_1$  currently shows  $b_{12}$ , and  $c_3$  is inert. Apply the atomic intervention  $\theta = (\{c_1\}, \mathcal{I}'_{c_1})$  with  $\mathcal{I}'_{c_1}(b_{11}) = \mathcal{I}'_{c_1}(b_{12}) = \mathcal{I}'_{c_1}(b_{13}) = b_{11}$ , leaving all other rules unchanged. After  $\theta$  every reachable configuration  $f$  of the intervened model satisfies  $f(c_1) = b_{11}$ . Because  $c_2$  can behave  $b_{22}$  only when  $b_{12}$  is in its influence context, the reset freezes  $c_2$ 's behaviour as  $b_{21}$ .  $\square$

*Remark 2.* If the original system  $\mathcal{M}$  is decomposable via an interface-dependent decomposition, then the intervened model  $\mathcal{M}_\theta$  remains decomposable under the same decomposition structure, as interventions do not alter the influence contexts, which govern the decomposition.  $\square$

In the sequel, a pointed system model is a pair consisting of a system model  $\mathcal{M}$  and a *chosen* configuration  $f$  in the model.

*Remark 3.* Consider an intervention  $\theta = (C', \mathcal{I}'_{C'})$ , where  $C' \subseteq \mathcal{C}$  is the subset of components targeted by the intervention and  $\mathcal{I}'_{C'} = \{I'_c \mid c \in C'\}$  is a set of new influence rules. For two pointed system models  $(\mathcal{M}_1, f) = (F, \Delta_{\mathcal{I}_1}, f)$  and  $(\mathcal{M}_2, f) = (F, \Delta_{\mathcal{I}_2}, f)$ , we say that  $\mathcal{M}_1 R_\theta \mathcal{M}_2$  holds if, for every component  $c \in \mathcal{C}$ ,  $\mathcal{I}_2(c) = I'_c$  if  $c \in C'$ , and  $\mathcal{I}_2(c) = \mathcal{I}_1(c)$  if  $c \notin C'$ , so that the intervention changes only the influence rules for components in  $C'$ . The transition relation  $\Delta_{\mathcal{I}_2}$  is then induced by these updated influence rules. We denote the union of all such relations with  $R_\theta$ .  $\square$

Our terms *component*, *influence*, *configuration* are inspired from the foundational work by Winskel on event structures [43] and by Simon [37].

### 3 A logic for minimal system models

In this section, we introduce a logical language, denoted by  $\mathcal{L}(\langle \theta \rangle, *)$ , tailored to capture the dynamic and structural aspects of minimal system models. Our language integrates standard modal operators  $\Box$  and  $\Diamond$  (with  $\Diamond$  as the dual of  $\Box$ ), a dynamic operator  $\langle \theta \rangle$  that reflects interventions on a set of components, and a structural separation operator,  $*$  — similar in spirit to the multiplicative connective as in, for example, [28, 23, 16], itself in the long tradition of relevance logic (e.g., in a vast literature, [34]) — which enables the decomposition of system configurations.

#### 3.1 Syntax and semantics

The language  $\mathcal{L}(\langle \theta \rangle, *)$  is given by  $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Box\varphi \mid \Diamond\varphi \mid \langle \theta \rangle\varphi \mid \varphi * \varphi$ , where  $p$  ranges over atomic propositions. Implication,  $\rightarrow$ , and disjunction,  $\vee$ ,

are defined in the usual (classical) way. We denote the subset consisting of  $*$ -free formulae by  $\mathcal{L}(\langle\theta\rangle)$ .

The semantics is defined relative to a system model  $\mathcal{M} = (F, \Delta_{\mathcal{I}}, \Gamma)$  over a set of components  $\mathcal{C}$ . Here,  $F$  is the set of full configurations, each assigning a behaviour to every component in  $\mathcal{C}$ ,  $\Delta_{\mathcal{I}}$  is a transition relation based on influence rules  $\mathcal{I}$ ,  $\Gamma$  is a valuation assigning propositions to configurations.

**Definition 9 (Semantics).** *Given a system model  $\mathcal{M} = (F, \Delta_{\mathcal{I}}, \Gamma)$  and a configuration  $f \in F$ , the satisfaction relation  $\models$  is defined as follows:*

$$\begin{aligned}
(\mathcal{M}, f) &\models p \text{ iff } f \in \Gamma(p) \\
(\mathcal{M}, f) &\models \neg\varphi \text{ iff } (\mathcal{M}, f) \not\models \varphi \\
(\mathcal{M}, f) &\models \varphi \wedge \psi \text{ iff } (\mathcal{M}, f) \models \varphi \text{ and } (\mathcal{M}, f) \models \psi \\
(\mathcal{M}, f) &\models \Box\varphi \text{ iff for every } f' \in F \text{ with } f \Delta_{\mathcal{I}} f', \text{ it holds that } (\mathcal{M}, f') \models \varphi \\
(\mathcal{M}, f) &\models \Diamond\varphi \text{ iff there exists some } f' \in F \text{ with } f \Delta_{\mathcal{I}} f' \text{ such that } (\mathcal{M}, f') \models \varphi \\
(\mathcal{M}, f) &\models \langle\theta\rangle\varphi \text{ iff there exists some intervention } \theta_{C'} \text{ (with } C' \subseteq \mathcal{C} \text{) and a} \\
&\quad \text{configuration } f' \text{ satisfying } f \Delta_{\mathcal{I}}^{\theta} f' \text{ such that } (\mathcal{M}_{\theta_{C'}}, f') \models \varphi, \\
&\quad \text{where } \mathcal{M}_{\theta_{C'}} = (F, \Delta_{\mathcal{I}}^{\theta_{C'}}, \Gamma) \text{ is the updated model} \\
(\mathcal{M}, f) &\models \varphi * \psi \text{ iff there exist } C_1, C_2 \subseteq \mathcal{C} \text{ such that } C_1 \cap C_2 \text{ constitutes} \\
&\quad \text{an interface, and both } (\mathcal{M}_{C_1}, f|_{C_1}) \models \varphi \text{ and} \\
&\quad (\mathcal{M}_{C_2}, f|_{C_2}) \models \psi, \text{ where } \mathcal{M}_{C_1} \text{ is the partial model over } C_1, \\
&\quad \text{and } \mathcal{M}_{C_2} \text{ is the partial model over } C_2
\end{aligned}$$

A model  $\mathcal{M}$  satisfies a formula  $\varphi$  at a configuration  $f$  iff  $(\mathcal{M}, f) \models \varphi$ . □

A formula  $\Box\varphi$  is read as ‘necessarily  $\varphi$ ’, meaning that in every configuration accessible from the current configuration via  $\Delta_{\mathcal{I}}$ , the formula  $\varphi$  holds. A formula  $\Diamond\varphi$  is read as ‘possibly  $\varphi$ ’, meaning that there exists a configuration accessible from the current configuration via  $\Delta_{\mathcal{I}}$  in which the formula  $\varphi$  holds. The separating conjunction  $*$  is introduced to partition the system into overlapping subsystems, via a shared interface, enabling modular reasoning about distinct parts of the system. A formula  $\varphi * \psi$  is read as ‘ $\varphi$  separating-conjoined with  $\psi$ ’, meaning that the system can be partitioned into two overlapping subsystems — with their shared interface mediating external influences — such that one subsystem satisfies  $\varphi$  and the other satisfies  $\psi$ . The intervention operator  $\langle\theta\rangle$  allows us to formally represent and evaluate counterfactual modifications. A formula  $\langle\theta\rangle\varphi$  is read as ‘there exists an intervention  $\theta$  such that after its application, the formula  $\varphi$  holds in the resultant model’.

*Remark 4.* Each configuration  $f$  is associated with a *characteristic formula*  $\chi_f = \bigwedge_{c \in \mathcal{C}} p_{c, f(c)}$ , such that  $(\mathcal{M}, f') \models \chi_f$  if and only if  $f' = f$ . This formula uniquely identifies  $f$ . □

## 4 Causal models

While there are other counterfactual approaches to causal modelling (cf. [25]), we align with Pearl and Halpern’s framework because it provides a robust, manipulable framework based on structural equations that clearly formalizes counterfactual reasoning, making it especially well-suited for complex systems.

#### 4.1 The Halpern-Pearl Framework

Pearl’s account [31] formalizes a causal model as a tuple  $M = \langle U, V, F \rangle$ , where  $U$  is a set of exogenous variables capturing external influences,  $V$  is a set of endogenous variables representing the internal state, and  $F$  is a family of functions (structural equations) of the form  $v_i = f_i(pa_i, u_i)$  for  $i = 1, \dots, n$ , with  $pa_i \subseteq V \setminus \{V_i\}$  being the minimal set of parent variables that determine  $V_i$ , and  $u_i \subseteq U$  the corresponding exogenous inputs. For any fixed assignment  $U := u$ , these equations yield a unique solution that defines a distinct causal scenario. Structural equations encode causal relationship by setting the left-hand variable as the effect and right-hand variables as causes, with equality signalling a directional ‘determined by’ relationship.

Building on Pearl’s approach, Halpern extends this foundation [19] by focusing on an event-centric perspective, distinguishing between type causality (general patterns) and token causality (specific instances). While our system modelling framework naturally aligns with Halpern’s analytical framework on actual causality. Since we do not model causality using random variables, we focus on actual causality rather than type causality among configurations.

*Actual causation* concerns retrospective causal claims — asserting that an event  $C$  was a cause of an effect  $E$ . Halpern’s extended framework distinguishes between endogenous and exogenous variables, where a causal model  $M = (S, F)$  consists of a signature  $S$  specifying variables and their possible values, and a set of structural equations  $F$  governing their interactions [20]. A causal setting is a pair  $(M, \vec{u})$ , where  $\vec{u}$  assigns values to exogenous variables, determining the behaviour of endogenous ones. In this framework, an event  $A$  (encoded by some formula  $\varphi$ ) is an actual cause of  $E$  (encoded by another formula  $\psi$ ) if (i) both  $A$  and  $E$  occur in the actual world, (ii) in a counterfactual world where  $A$  is absent but *all else* remains fixed,  $E$  does not occur, and (iii)  $A$  is minimal, meaning no proper subset of  $A$  suffices to bring about  $E$ . The Halpern-Pearl (HP) definition of actual causation [19] formalizes these three criteria of actual causal relationships via three clauses — AC1, AC2, and AC3 (see the appendix for details). In a similar manner, we introduce our notion of *cause* within the context of system models, aligning our approach with the HP criteria while adapting it to the dynamics of configuration-based systems. We use a variant of AC2( $a^m$ ) clause introduced in [18].

**Definition 10 (Cause).** Let  $\mathcal{M} = (F, \Delta_{\mathcal{I}}, \Gamma)$  be a system model, and  $f_1, f_2 \in F$  be configurations over components  $\mathcal{C}$ . Let  $\psi_E = \bigwedge_{c \in \mathcal{C}_E} p_{c=f_2(c)}$  be the effect formula, where  $\mathcal{C}_E \subseteq \mathcal{C}$  is the set of components relevant for determining the outcome. A subset of components  $\mathcal{C}' \subseteq \mathcal{C}$ , whose behaviours are fixed as in  $f_1$  (i.e.,  $\chi_C = \bigwedge_{c \in \mathcal{C}'} p_{c=f_1(c)}$ ), is called a **cause** of  $f_2$  from  $f_1$  (denoted  $\text{Cause}(f_1, f_2)$ ) if the following conditions hold:

1. There exists a sequence of transitions such that  $f_1 \Delta_{\mathcal{I}}^+ f_2$ , where  $\Delta_{\mathcal{I}}^+$  is the transitive closure of  $\Delta_{\mathcal{I}}$ , and  $f_1(c) = f_2(c)$  for all  $c \in \mathcal{C}'$ . This is expressed as  $\Diamond^+(\psi_E \wedge \chi_C)$ , and is an actuality condition (analogous to **AC1** in HP definitions [19]).

2. *There exists a witness set  $\mathcal{W} \subseteq \mathcal{C}$  such that for any configuration  $f'_1$  where  $f'_1(c) = f_1(c)$  for all  $c \in \mathcal{W}$ , but  $f'_1(c) \neq f_1(c)$  for some  $c \in \mathcal{C}' \setminus \mathcal{W}$ , if  $f'_1 \Delta_{\mathcal{I}} f'_2$ , then  $f'_2 \neq f_2$ . This is the counterfactuality condition analogous to **AC2** in HP definitions [19]).*

Let  $\chi_C = \bigwedge_{c \in C} p_{c=f_1(c)}$ ,  $\chi_{\mathcal{W}} = \bigwedge_{c \in \mathcal{W}} p_{c=f_1(c)}$ . Also let

$$\chi'_{C \setminus \mathcal{W}} = \bigvee_{c \in C \setminus \mathcal{W}} \neg p_{c=f_1(c)}$$

be the formula expressing that at least one component in  $C \setminus \mathcal{W}$  has changed relative to  $f_1$  after an intervention. The counterfactual condition can be expressed as  $\langle \theta \rangle (\chi_{\mathcal{W}} * \chi'_{C \setminus \mathcal{W}}) \rightarrow \Box^+ \neg (\psi_E \wedge \chi_C)$

3. *There is no proper subset  $\mathcal{C}'' \subset \mathcal{C}'$  that satisfies both the above conditions. This is the Minimality condition analogous to **AC3** in HP [19]).*  $\square$

The invariance of the candidate cause's behaviours across a transition from configuration  $f_1$  to  $f_2$ , expressed as  $f_1(c) = f_2(c)$  for all  $c \in \mathcal{C}'$ , aligns with Halpern and Pearl's AC1 condition, which requires that both the candidate cause and the effect hold in the actual world. It confirms the candidate cause's presence in the actual system evolution, enabling counterfactual analysis: by altering the candidate cause in a hypothetical scenario and observing the effect's absence, we isolate its causal role. The second condition helps isolate the subset of components which constitute a cause. The third conditions ensures no proper subset of the candidate cause suffices as an actual cause — by enforcing that every component in  $\mathcal{C}'$  is essential; restricting to any proper subset  $\mathcal{C}'' \subset \mathcal{C}'$  disrupts either the invariance in the actual transition or the counterfactual dependence, thus preventing over-attribution.

Understanding how changes propagate through a system is essential for analysing causality. A *causal chain* captures this progression by linking configurations through causal dependencies (refer to the definition of cause), ensuring that each transition satisfies the established criteria of causal relationships.

**Definition 11 (Causal chain).** *A causal chain in a given system model  $\mathcal{M} = (F, \Delta_{\mathcal{I}}, \Gamma)$  is a finite sequence of configurations  $(f_1, f_2, \dots, f_n)$  with  $n \geq 2$  and each  $f_i \in F$ , satisfying the following conditions:*

1. *For each consecutive pair  $(f_i, f_{i+1})$ , there exists a subset of components  $\mathcal{C}_i \subseteq \mathcal{C}$  such that  $\mathcal{C}_i$  is a cause of  $f_{i+1}$  from  $f_i$  according to the three criteria (actuality, counterfactuality, minimality).*
2. *For each  $i$ , it holds true that  $(f_i, f_{i+1}) \in \Delta_{\mathcal{I}}^+$ , meaning that the causal influence is realizable through one or more transitions in the system.*
3. *The chain is minimal in the sense that no configuration  $f_k$  can be removed without violating sequential causality. This ensures that the chain does not include superfluous steps.*

We denote the set of all causal chains in  $\mathcal{M}$  by  $\text{Chain}(\mathcal{M})$ .  $\square$

**Definition 12 (Causal system model).** *A causal projection of a system model  $\mathcal{M} = (F, \Delta_{\mathcal{I}}, \Gamma)$  is a tuple  $(F^c, \Delta^c, \Gamma^c)$  such that  $F^c \subseteq F$  consists of configurations that appear in at least one causal chain in  $\text{Chain}(\mathcal{M})$ , and,  $\Delta^c$  and  $\Gamma^c$  are the restrictions of  $\Delta_{\mathcal{I}}$  and  $\Gamma$  respectively to  $F^c$ . The system model  $\mathcal{M} = (F, \Delta_{\mathcal{I}}, \Gamma)$  is called a causal system model if it has a causal projection.  $\square$*

If the graph  $(F^c, \Delta^c)$  is acyclic, i.e.,  $\Delta^c$  is a partial order then there are no ‘causal loops’ (a ‘causal loop’ is formed when for any two configurations  $f_1$  and  $f_2$ , both  $f_1$  and  $f_2$  are causes of each other).

Lemma 1, below, characterizes how interventions affect causal chains by delineating conditions under which such chains are either preserved or disrupted. It is proved in [12].

**Lemma 1 (Characterization of Interventions in Causal System Models).** *Let  $\mathcal{M} = (F, \Delta_{\mathcal{I}}, \Gamma)$  be a causal system model with causal projection  $\mathcal{M}^c = (F^c, \Delta_{\mathcal{I}}^c, \Gamma^c)$ . Let  $\theta = (C', I_{C'})$  be an intervention yielding the intervened system  $\mathcal{M}_\theta = (F, \Delta_{\mathcal{I}_\theta}, \Gamma)$ . If a causal chain  $(f_1, \dots, f_n) \in \text{Chain}(\mathcal{M})$  does not involve any component in  $C'$  as part of the cause for any transition, then this chain is preserved under intervention  $\theta$ . Formally,  $\forall i(1 \leq i < n), C' \cap \text{Cause}(f_i, f_{i+1}) = \emptyset \Rightarrow (f_1, \dots, f_n) \in \text{Chain}(\mathcal{M}_\theta)$ . Otherwise if  $(f_1, \dots, f_n)$  contains a configuration  $f_i$  whose cause involves components in  $C'$ , and the intervention  $\theta$  modifies the influence rules such that the causal transition to  $f_{i+1}$  is invalidated, then the chain is disrupted. Formally,  $\exists i(1 \leq i < n)$  such that  $C' \cap \text{Cause}(f_i, f_{i+1}) \neq \emptyset$  and  $\langle \theta \rangle \neg (f_i \Delta_{\mathcal{I}_\theta} f_{i+1})$ .  $\square$*

The following theorem, proved in the appendix, relates our approach to modelling causes in systems with the HP framework [19]:

**Theorem 1.** *Let  $\mathcal{M} = (F, \Delta_{\mathcal{I}}, \Gamma)$  be a causal system model over a finite component set  $\mathcal{C}$ , where causes are defined via causal chains satisfying actuality, counterfactual dependence, and minimality (cf. Definition 10). Construct a corresponding HP causal model  $M = \langle U, V, F \rangle$ , where  $V = \{V_c \mid c \in \mathcal{C}\}$  with  $\text{dom}(V_c) = \mathbb{B}(c)$  and the structural equations in  $F$  are induced by the influence rules  $\mathcal{I}$  of  $\mathcal{M}$ . For any configuration  $f_2 \in F$ , define the effect formula  $\varphi_{f_2} = \bigwedge_{c \in \mathcal{C}} (V_c = f_2(c))$ . Then, if there exists a causal chain in  $\mathcal{M}$  from an initial configuration  $f_1$  to  $f_2$ , one can extract a candidate cause; that is, a subset  $\vec{X} \subseteq V$  and an assignment  $\vec{x}$  (with a corresponding context  $\vec{u}$ ) such that the assignment  $\vec{X} = \vec{x}$  satisfies the HP criteria (AC1–AC3) for being an actual cause of  $\varphi_{f_2}$  in  $(M, \vec{u})$ . In other words, the existence of a causal chain from  $f_1$  to  $f_2$  in  $\mathcal{M}$  implies that there is a corresponding actual cause in the HP model.  $\square$*

## 5 Security examples

Modern cloud-native applications often decompose functionality into independently deployable microservices consisting of a very large number of services. Microservices architecture promotes cost optimization and sustainability by enabling selective scaling of components based on demand, minimizing resource use

and waste. It also allows for smaller, independent updates, reducing the need for extensive end-to-end testing compared to monolithic systems. This shift from monolithic to microservice-based architectures has transformed how software is designed, deployed, and maintained [45] (cf. refer to this whitepaper from Amazon Web Services [1] for technical details).

This evolution has also intensified the need for rigorous tools in forensic analysis and audit. A framework for actual causation, grounded in Halpern’s approach [19] but adapted to model system transitions, is well-suited as a first step in addressing these challenges (see also [13,9] on model design perspectives).

### 5.1 Microservices

Since in microservice-based architecture, an application typically consists of a large number of loosely coupled, fine-grained services, accurately reconstructing inter-service call graphs is non-trivial. Dependencies evolve at runtime, and often lack static configurations. [45]. These difficulties have motivated causal-discovery techniques in industry-facing tools [22], stressing the need for a rigorous framework such as the one proposed here.

Decomposition into loosely coupled services with explicit APIs (Application Program Interface) mirrors our formal notions of components, configurations, and interfaces, making microservices an ideal case study. Their ubiquity in large-scale deployments ensures industrial relevance, failures often arise from identifiable interactions among just a few services, and operational practice already employs one-shot mitigations (rolling updates, circuit breakers, traffic re-routes) that correspond to the atomic interventions in our logic. Since failures frequently trace back to a small cluster of inter-service interactions, actual causality is a useful notion in determining the precise chain of responsibility for post-incident audits and forensic analysis in microservice deployments.

### 5.2 Graph-based paradigms

Several existing tools employ *causal dependency graph* to trace how anomalies propagate through microservice ecosystems. For instance, Microscope [27] infers service dependencies in real time to build a *service impact graph*, which it combines with runtime anomalies to derive a causality graph. Groot [42], designed for large-scale systems, constructs a global dependency graph and, upon alerts, extracts a focused subgraph around affected services. It aggregates events (e.g., CPU spikes, HTTP errors, code changes) and uses domain-specific rules to assemble an *event causality graph*. While effective for diagnostics, such tools treat causality observationally and do not support formal reasoning about interventions or counterfactuals [5].

Collectively, these systems illustrate the power of graph-based methods in *heuristically* localizing root causes. Yet, they fall short of providing a *rigorous* foundation for *actual causation*; that is, a precise characterization of ‘which component state (or event) truly *caused* the observable failure’, in the sense of Halpern–Pearl counterfactual dependence [20].

We argue that, in the absence of a formal specification language for expressing actual causality in dynamic systems (in contrast to do-calculus, which is designed for causal inference), such approaches remain inadequate for purposes of audit. This limitation is particularly acute in high-stakes scenarios, such as microservice-based infrastructures in financial exchanges, where root cause analysis is often conducted through ad hoc means. For instance, the consultation paper issued by the UK Financial Conduct Authority [15] exemplifies the use of informal causal chain-based analysis for forensics and audit.

While full empirical validation is beyond the scope of this theoretical development, our framework is conceptually compatible with existing microservice monitoring tools (such as [27,42]), where detected anomalies correspond to particular configurations or behaviour assignments in our model. Future empirical work could involve systematically translating observed anomalies and performance metrics into formal configurations and causal chains.

MAYBE DROP NEXT SECTION TO MAKE SPACE FOR EXAMPLES ABOVE?

### 5.3 Modelling microservices

In this section, we illustrate how to apply the system-modelling framework to a small microservice deployment. We then show how to decompose the system into subsystems with a shared interface. We also show how strategic queries can be formulated in this approach using counterfactuals. For the sake of brevity we defer a thorough discussion of formally identifying an actual cause of a failure (by invoking Definition 10) to [12].

**Components and behaviours** In a typical web application using the microservices architecture, the following design pattern is often used: **Auth** handles user authentication, **UserDB** manages credential storage and lookup, **ProfileSvc** provides user profile information, **Logger** records system events and requests, and **FrontEnd** serves as the user-facing component coordinating interactions among the back-end services.

In our framework, this corresponds to a set of components

$$\mathcal{C} = \{\text{Auth}, \text{UserDB}, \text{ProfileSvc}, \text{Logger}, \text{FrontEnd}\}$$

Each component  $c \in \mathcal{C}$  is associated with a set of permissible behaviours (the behaviour names are self-explanatory):

$$\begin{aligned} \mathbb{B}(\text{Auth}) &= \{\text{idle}, \text{authSucc}, \text{authFail}\} & \mathbb{B}(\text{ProfileSvc}) &= \{\text{idle}, \text{profileOK}, \text{TimeOut}\} \\ \mathbb{B}(\text{UserDB}) &= \{\text{idle}, \text{dbOK}, \text{dbError}\} & \mathbb{B}(\text{Logger}) &= \{\text{idle}, \text{logged}, \text{logFail}\} \\ & & \mathbb{B}(\text{FrontEnd}) &= \{\text{idle}, \text{serving}, \text{error}\} \end{aligned}$$

A *configuration*  $f \in F$  is a function  $f: \mathcal{C} \rightarrow \bigcup_{c \in \mathcal{C}} \mathbb{B}(c)$ , with  $f(c) \in \mathbb{B}(c)$  for each  $c$ .

We now specify, for each component  $c \in \mathcal{C}$ , an *influence context*  $\text{Inf}(c)$  (the subset of other components whose behaviours can affect  $c$ ), and then give a local influence rule  $\mathcal{I}_c$  as in Definition 2:

1.  $E(\text{Auth}) = \{\text{FrontEnd}, \text{UserDB}\}$ . The authentication service first receives a request from the front end; if it reaches out to the user database for credentials, then **UserDB**'s state may induce a success or failure.
2.  $E(\text{UserDB}) = \{\text{Auth}\}$ . The database processes queries only when the auth service requests it.
3.  $E(\text{ProfileSvc}) = \{\text{Auth}, \text{UserDB}\}$ . The profile service fetches user data only after successful authentication and a database read.
4.  $E(\text{Logger}) = \{\text{Auth}, \text{ProfileSvc}, \text{FrontEnd}\}$ . The logger records each request, authentication attempt, and profile lookup.
5.  $E(\text{FrontEnd}) = \{\text{Auth}, \text{ProfileSvc}, \text{Logger}\}$ . The front-end serves pages only after successful authentication and profile data, and logs its own error or serving state.

Accordingly, we define local influence rules  $\mathcal{I}_c : \mathbb{B}(c) \times \prod_{d \in \text{Inf}(c)} \mathbb{B}(d) \rightarrow \mathbb{B}(c)$  for each  $c$ . Below, we write  $\mathcal{I}_c(b_c, (b_d)_{d \in \text{Inf}(c)})$  for the output behaviour, given current behaviour  $b_c$  of  $c$  and behaviours  $b_d$  of each  $d \in \text{Inf}(c)$ . (We omit trivial cases where a component remains idle if nothing relevant changes.)

**Authentication service**  $\mathcal{I}_{\text{Auth}}$  caters to all authentication activities required for interaction with external users.

$$\begin{aligned}\mathcal{I}_{\text{Auth}}(\text{idle}, (\text{serving}, \text{dbOK})) &= \text{authSucc} \\ \mathcal{I}_{\text{Auth}}(\text{idle}, (\text{serving}, \text{dbError})) &= \text{authFail}\end{aligned}$$

That is, when the front end issues a login request (modelled as **FrontEnd** = *serving*) and the database is OK, then **Auth** transitions to *authSucc*; if the database is in *dbError*, then **Auth** transitions to *authFail*.

**User database**  $\mathcal{I}_{\text{UserDB}}$ :

$$\begin{aligned}\mathcal{I}_{\text{UserDB}}(\text{idle}, (\text{authSucc})) &= \text{dbOK} \\ \mathcal{I}_{\text{UserDB}}(\text{idle}, (\text{authFail})) &= \text{dbError}\end{aligned}$$

Thus, if **Auth** has just succeeded, the database returns *dbOK*; if **Auth** failed, the database reports *dbError*.

**Profile service**  $\mathcal{I}_{\text{ProfileSvc}}$ :

$$\begin{aligned}\mathcal{I}_{\text{ProfileSvc}}(\text{idle}, (\text{authSucc}, \text{dbOK})) &= \text{profileOK} \\ \mathcal{I}_{\text{ProfileSvc}}(\text{idle}, (\text{authFail}, \_)) &= \text{TimeOut},\end{aligned}$$

where  $\_$  denotes ‘any database state’. In other words, if authentication succeeds *and* the database is OK, the profile lookup succeeds; if authentication fails, the profile request times out.

**Logger**  $\mathcal{I}_{\text{Logger}}$  acts as a shared interface between other components.

$$\mathcal{I}_{\text{Logger}}(\text{idle}, (b_{\text{Auth}}, b_{\text{ProfileSvc}}, \text{serving})) = \text{logged},$$

whenever  $b_{\text{Auth}} \in \{\text{authSucc}, \text{authFail}\}$ ,  $b_{\text{ProfileSvc}} \in \{\text{profileOK}, \text{TimeOut}\}$

$$\mathcal{I}_{\text{Logger}}(\text{idle}, (b_{\text{Auth}}, \text{error})) = \text{logFail}$$

Thus, if the front end is *serving* and both Auth and ProfileSvc have transitioned to some success/failure state, the logger records it (*logged*). If the front end itself is in *error*, the logger may fail to log (*logFail*).

**Front-end**  $\mathcal{I}_{\text{FrontEnd}}$ :  $\mathcal{I}_{\text{FrontEnd}}(\text{idle}, (b_{\text{Auth}}, b_{\text{ProfileSvc}}, b_{\text{Logger}})) = \text{serving}$ , if  $b_{\text{Auth}} = \text{authSucc}$  and  $b_{\text{ProfileSvc}} = \text{profileOK}$  and  $b_{\text{Logger}} = \text{logged}$ . Otherwise, it equals *error* if  $b_{\text{Auth}} = \text{authFail} \vee b_{\text{ProfileSvc}} = \text{TimeOut} \vee b_{\text{Logger}} = \text{logFail}$ . In other words, the front end will serve the requested page only if authentication and profile lookup succeed and the logger has recorded those events; otherwise it enters an *error* state.

**The system model** Collecting everything, we obtain a *system model*  $\mathcal{M} = (\mathcal{C}, \mathcal{B}, \mathcal{I}, F, \Delta_{\mathcal{I}}, \Gamma)$ , where

1.  $\mathcal{C}$  is the component set above,
2.  $\mathcal{B} = \bigcup_{c \in \mathcal{C}} \mathbb{B}(c)$  is the union of all behaviour sets,
3.  $\mathcal{I} = \{\mathcal{I}_c \mid c \in \mathcal{C}\}$  is the family of influence rules just defined.
4.  $F$  is the set of all full configurations  $f : \mathcal{C} \rightarrow \bigcup_c \mathbb{B}(c)$ ,
5.  $\Delta_{\mathcal{I}} \subseteq F \times F$  is the one-step transition relation induced by  $\mathcal{I}$ ,

$$(f, f') \in \Delta_{\mathcal{I}} \quad \text{iff} \quad \forall c \in \mathcal{C}, f'(c) = \mathcal{I}_c(f(c), (f(d))_{d \in \text{Inf}(c)})$$

6.  $\Gamma : \mathcal{P} \rightarrow 2^F$  is a valuation that assigns, for each atomic proposition in a chosen propositional vocabulary  $\mathcal{P}$ , the set of configurations in which it holds. For example,  $\Gamma(p_{\text{FrontEnd}=\text{error}}) = \{f \in F \mid f(\text{FrontEnd}) = \text{error}\}$ , and similarly for propositions like  $p_{\text{Auth}=\text{authFail}}$ , and so on.

To show that  $\mathcal{M}$  admits a non-trivial interface decomposition, partition  $\mathcal{C}$  into  $C_1 = \{\text{Auth}, \text{UserDB}, \text{Logger}\}$  and  $C_2 = \{\text{ProfileSvc}, \text{FrontEnd}, \text{Logger}\}$ . Note that  $C_1 \cup C_2 = \mathcal{C}$  and  $I = C_1 \cap C_2 = \{\text{Logger}\}$  is the interface. We can check the two conditions of Definition 4 (Interface-admitting System Model).

#### 5.4 Strategic decision queries

We now show how to *formalize decision-making questions* in the microservice example without developing a full game-theoretic apparatus. Note that it could have been formulated in the setting of a multi-agent game.

**Notation** Recall  $\varphi_{\text{fail}} = p_{\text{FrontEnd}=\text{error}}$ , and the three candidate interventions:

$$\begin{aligned} \theta_1 &= (\{\text{UserDB}\}, \{\mathcal{I}'_{\text{UserDB}}\}), \quad \mathcal{I}'_{\text{UserDB}}(\cdot) := \text{dbOK} \\ \theta_2 &= (\{\text{FrontEnd}\}, \{\mathcal{I}'_{\text{FrontEnd}}\}), \quad \mathcal{I}'_{\text{FrontEnd}}(\_, \_) := \text{servingCache} \\ \theta_3 &= (\{\text{ProfileSvc}\}, \{\mathcal{I}'_{\text{ProfileSvc}}\}), \quad \mathcal{I}'_{\text{ProfileSvc}}(\_, \_) := \text{profileStale} \end{aligned}$$

**Guaranteed recovery** Which interventions  $\theta_i$  guarantee  $\neg\varphi_{\text{fail}}$  from configuration  $f_2$ ? Formally,  $(\mathcal{M}, f_2) \models \langle \theta_i \rangle \Box \neg\varphi_{\text{fail}}$ . In our example,

$$(\mathcal{M}, f_2) \models \langle \theta_1 \rangle \Box \neg\varphi_{\text{fail}}, (\mathcal{M}, f_2) \models \langle \theta_2 \rangle \Box \neg\varphi_{\text{fail}}, \text{ and } (\mathcal{M}, f_2) \not\models \langle \theta_3 \rangle \Box \neg\varphi_{\text{fail}}$$

Thus  $\theta_1$  (repairing the DB) and  $\theta_2$  (cache-serve) are *valid recovery policies*, while  $\theta_3$  is not.

**Minimal-cost intervention** Suppose we assign costs to each  $\theta_i$ :  $Cost(\theta_1) = 10$ ,  $Cost(\theta_2) = 5$ ,  $Cost(\theta_3) = 2$ , where, for example, repairing the database is more expensive than re-routing to the cache. We wish to choose the  $\theta_i$  that (i) satisfies  $\langle \theta_i \rangle \Box \neg \varphi_{\text{fail}}$  and (ii) minimizes  $Cost(\theta_i)$ . The corresponding formula might be

$$(\mathcal{M}, f_2) \models \langle \theta_i \rangle \Box \neg \varphi_{\text{fail}} \text{ and } (\mathcal{M}, f_2) \models \langle \zeta_j \rangle \Box \neg \varphi_{\text{fail}} \text{ implies } (Cost(\theta_i) \leq Cost(\zeta_j))$$

for some  $\theta_i$  and for all  $\zeta_j$  (a predicate version of the logic could be used to internalize the quantifications). In our setting,  $\theta_2$  is chosen, since  $Cost(\theta_2) = 5$  is the lowest cost among  $\{\theta_1, \theta_2\}$  that guarantee recovery.

**Fallback vs. repair trade-off** If  $Utility(\theta_i)$  combines cost and the user-satisfaction penalty (e.g., stale data penalty), we can write  $Utility(\theta_i) = -Cost(\theta_i) - Penalty(\theta_i)$ , and ask for  $(\mathcal{M}, f_2) \models \langle \theta_i \rangle \Box \neg \varphi_{\text{fail}}$  and  $(\mathcal{M}, f_2) \models \langle \zeta_j \rangle \Box \neg \varphi_{\text{fail}}$  implies  $(Utility(\theta_i) \geq Utility(\theta_j))$ , for some  $\theta_i$  and for all  $\zeta_j$ . This yields the ‘best trade-off’ policy under a combined cost-penalty metric.

**Localized decision-making through separation (using \*)** In this example, using the interface  $\{\text{Logger}\}$ , we can ensure that an intervention on one subsystem does not violate invariants in the other. For instance, when applying  $\theta_2$ , we require

$$(\mathcal{M}, f_2) \models \langle \theta_2 \rangle (\varphi_{C_1} * \varphi_{C_2})$$

where  $\varphi_{C_1} = p_{\text{UserDB}=dbError} \wedge p_{\text{Logger}=logged}$  and  $\varphi_{C_2} = p_{\text{FrontEnd}=servingCache} \wedge p_{\text{Logger}=logged}$ . This asserts that after forcing the front-end to *servingCache*, the  $C_1$ -subsystem (DB-Auth-Logger) can continue with  $\text{Logger} = \text{logged}$  and  $\text{UserDB} = dbError$ , while the  $C_2$ -subsystem (ProfileSvc-FrontEnd-Logger) enters a safe ‘cache’ configuration. Thus the intervention respects subsystem locality and prevents cross-subsystem side-effects.

**Strategic perspective.** This framework could have been enriched by viewing an orchestrator (defender) and external failures (attackers) as players: the defender’s strategy set would be  $\{\theta_1, \theta_2, \theta_3, \dots\}$ , while the adversary’s ‘strategy’ is the choice of which component fails next. A natural payoff function rewards the absence of failures minus the cost of interventions. While our logical intervention-queries already suffice to guide practical decision-making without constructing a full game model, the framework naturally suggests a fuller game-theoretic treatment; we therefore leave the explicit formulation of full strategy spaces, payoff functions, and equilibrium concepts to future work.

## 5.5 Example Configurations and an Actual Cause

Given two configurations,  $f_1, f_2 \in F$ , We now exhibit how to identify a set of components  $C'$  as a *cause* of  $f_2$  from  $f_1$  per Definition 10. Intuitively,  $f_1$  will be read as a normal ‘no-error’ configuration, while  $f_2$  exhibits a front-end error. We show that a misconfiguration in *UserDB* (in  $C_1$ ) is the actual cause of  $f_2$ .

**Configuration  $f_1$ :** All components are idle or behave in a successful manner. All services await incoming requests.

$$\begin{aligned} f_1(\text{Auth}) &= \text{idle} & f_1(\text{Logger}) &= \text{idle} \\ f_1(\text{UserDB}) &= \text{idle} & f_1(\text{FrontEnd}) &= \text{idle} \\ f_1(\text{ProfileSvc}) &= \text{idle} \end{aligned}$$

**Configuration  $f_2$ :** (A front-end error due to a database fault.)

$$\begin{aligned} f_2(\text{Auth}) &= \text{authFail} & f_2(\text{Logger}) &= \text{logged} \\ f_2(\text{UserDB}) &= \text{dbError} & f_2(\text{FrontEnd}) &= \text{error} \\ f_2(\text{ProfileSvc}) &= \text{profileTimeout} \end{aligned}$$

Here, the request reached the front end, **Auth** attempted to authenticate, but **UserDB** returned *dbError*, leading to **Auth** = *authFail*, **ProfileSvc** = *profileTimeout*, the logger recorded the events, and finally the front end transitioned to *error*.

**Effect Formula  $\psi_E$ .** We consider the observable failure ‘FrontEnd is in error’ as the effect,  $\psi_E = p_{\text{FrontEnd}=\text{error}}$ . Thus  $\psi_E$  holds exactly in those configurations where  $f(\text{FrontEnd}) = \text{error}$ .

**Candidate Cause  $C' = \{\text{UserDB}\}$ .** We claim that fixing **UserDB** in state *dbError* (as in  $f_2$ ) is an *actual cause* of  $f_2$  from  $f_1$ . To check Definition 10, we let  $\chi_C = p_{\text{UserDB}=\text{dbError}}$ . Intuitively, ‘UserDB is stuck in *dbError*’ is our cause candidate.

**(i) Actuality.** We must show that there is a sequence of transitions from  $f_1$  to  $f_2$  in which **UserDB** remains *dbError*. Indeed, consider the following one-step transitions (written  $f \Delta_I f'$ ):

$$\begin{array}{lll} f_1 & \xrightarrow{\text{FrontEnd: idle} \rightarrow \text{serving}} & f'_1, \text{ where } f'_1(\text{FrontEnd}) = \text{serving}, \text{ others unchanged.} \\ f'_1 & \xrightarrow{\text{UserDB: idle} \rightarrow \text{dbError}} & f'_2, \text{ (misconfiguration injected).} \\ f'_2 & \xrightarrow{\text{Auth: idle} \rightarrow \text{authFail}} & f'_3, \text{ since } E(\text{Auth}) = (\text{serving}, \text{dbError}). \\ f'_3 & \xrightarrow{\text{ProfileSvc: idle} \rightarrow \text{profileTimeout}} & f'_4, \text{ since } E(\text{ProfileSvc}) = (\text{authFail}, \text{dbError}). \\ f'_4 & \xrightarrow{\text{Logger: idle} \rightarrow \text{logged}} & f'_5, \text{ since it sees } (\text{authFail}, \text{profileTimeout}, \text{serving}). \\ f'_5 & \xrightarrow{\text{FrontEnd: serving} \rightarrow \text{error}} & f_2, \text{ since } E(\text{FrontEnd}) = (\text{authFail}, \text{profileTimeout}, \text{logged}). \end{array}$$

Throughout this run, once **UserDB** transitions to *dbError*, it stays in that state. Hence **UserDB** = *dbError* in all intermediate configurations, and eventually  $f_2(\text{FrontEnd}) = \text{error}$ . Thus  $\Diamond^+(\psi_E \wedge \chi_C)$  holds. Moreover, in  $f_1$  we indeed have  $f_1(\text{UserDB}) = \text{idle} \neq \text{dbError}$ , so the cause condition “UserDB is set to *dbError*” is nontrivial.

(ii) *Counterfactual.* We must exhibit a witness set  $\mathcal{W} \subseteq \mathcal{C}$  and show that if UserDB were *not* set to *dbError* (while keeping  $\mathcal{W}$  fixed), then no run leads to  $f_2$  exactly. Take  $\mathcal{W} = \{\text{Auth}, \text{ProfileSvc}, \text{Logger}, \text{FrontEnd}\}$ . In other words, we hold all other components at their post-failure states (in  $f_2$ ) except UserDB. To apply Definition 10, we consider an intervention  $\theta$  that forces all components in  $\mathcal{W}$  to their  $f_2$  values but *does not* force UserDB, allowing it to vary,  $\theta = (\mathcal{W}, \{\mathcal{I}'_c : c \in \mathcal{W}\})$ ,  $\mathcal{I}'_c$  simply sets  $c$  to  $f_2(c)$  immediately and stably.

Under this intervention, UserDB is free, and all other components behave exactly as in  $f_2$ . Now, if we keep  $\text{Auth} = \text{authFail}$ ,  $\text{ProfileSvc} = \text{profileTimeout}$ ,  $\text{Logger} = \text{logged}$ ,  $\text{FrontEnd} = \text{error}$  but let UserDB deviate from *dbError* (i.e.  $f'_1(\text{UserDB}) = \text{idle}$  or *dbOK*), then **Auth** could not have arrived at *authFail* via  $\mathcal{I}_{\text{Auth}}$  as defined, nor could **ProfileSvc** reach *profileTimeout*, nor could **FrontEnd** become *error* under the same influence rules. Concretely, with  $\text{UserDB} = \text{dbOK}$ , one would get  $\text{Auth} = \text{authSuccess}$  and  $\text{ProfileSvc} = \text{profileOK}$ , forcing  $\text{FrontEnd} = \text{serving}$ . Hence no run  $(f'_1) \Delta_{\mathcal{I}} f'_2$  can yield  $f'_2(\text{FrontEnd}) = \text{error}$ . This establishes

$$\langle \theta \rangle (\chi_{\mathcal{W}} * \chi'_{\{\text{UserDB}\} \setminus \mathcal{W}}) \longrightarrow \Box^+ \neg (\psi_E \wedge \chi_C),$$

verifying the counterfactual condition (AC2) of Definition 10.

(iii) *Minimality.* Finally, no proper subset of  $\{\text{UserDB}\}$  is nonempty, so minimality holds vacuously. Thus  $\mathcal{C}' = \{\text{UserDB}\}$  is indeed an *actual cause* of  $f_2$  from  $f_1$ .

**Interpretation.** This formal analysis shows how a single misbehaving component in  $C_1$  (the database) sufficed to produce the end-user failure “FrontEnd error” in  $C_2$ , via the shared interface component “Logger.” Because **Logger** mediates all observable events between subsystems, we can decompose the global system without losing information and still identify  $\text{UserDB} = \text{dbError}$  as the minimal actual cause of  $\text{FrontEnd} = \text{error}$ .

## 6 Logical metatheory

In this section, it is established that our constructions obey a flavour of some well-established theorems that relate structural and behavioural properties of the logic. In particular, we establish two van Benthem-Bergstra-Hennessy-Milner [6] theorems. Inspired by sabotage logic [4], we use a model-changing notion of bisimulation under intervention that extends the standard back-and-forth (zig and zag) conditions (cf. [7]) to account for structural modifications induced by interventions. Further details are given in [12].

Theorem 2 establishes that two bisimulation equivalent interface-admitting system models are logically equivalent with respect to  $\mathcal{L}(\langle \theta \rangle, *)$ . Theorem 3 establishes the converse under specific restrictions on the language and for the subclass of interface-admitting system models with finitely many components and behaviours.

**Theorem 2.** *If two interface-admitting pointed system models,  $(\mathcal{M}_1, f_1)$  and  $(\mathcal{M}_2, f_2)$  are bisimilar under intervention then for all formulae  $\varphi \in L(\langle\theta\rangle, *)$ , if  $(\mathcal{M}_1, f_1) \models \varphi$  then  $(\mathcal{M}_2, f_2) \models \varphi$ .  $\square$*

**Theorem 3.** *Given a formula  $\varphi \in \mathcal{L}(\langle\theta\rangle)$ , for any two interface-admitting pointed system models  $(\mathcal{M}_1, f_1)$  and  $(\mathcal{M}_2, f_2)$  with finitely many components and behaviours, if  $(\mathcal{M}_1, f_1) \models \varphi$  implies  $(\mathcal{M}_2, f_1) \models \varphi$  then there exists a bisimulation under intervention relating  $(\mathcal{M}_1, f_1)$  and  $(\mathcal{M}_2, f_2)$ .  $\square$*

Proofs of Theorems 2 and 3 are provided in [12].

## 7 Discussion

One key distinguishing feature of our modelling approach, compared to traditional Structural Causal Models (SCMs), is the use of interventions as explicit mechanisms to enact *rule changes*, rather than merely altering variable assignments or fixed structural equations. Standard SCM approaches typically assume stable causal mechanisms represented by structural equations that remain constant throughout analysis. In contrast, our intervention modality directly modifies influence rules governing component behaviour, offering greater flexibility in modelling scenarios involving dynamic system evolution, adaptation, or deliberate operational changes.

It provides a means for decomposing and modularly reasoning about system configurations and their causal interactions, making it particularly advantageous for forensic and audit scenarios as exemplified by the microservice-based architectures example. Beyond microservice deployments, the combination of rule-based influence modelling and substructural logic applies naturally to several high-stakes domains that demand post-incident accountability. In industrial control systems, programmable-logic controllers, sensors, and safety interlocks already expose explicit control rules; atomic overrides map conceptually to our intervention operator, while separating conjunction models isolated mixing subsystems that interact only through shared pressure signals.

In large-scale payment and trading platforms gateways, atomic update to the codebase can be treated as interventions (in some suitable localised context) letting auditors trace a settlement outage to the precise validation rule that triggered it. Similarly in the context of blockchain-based Decentralised Finance Lending, root cause of flash-crashes (a sudden drop in the price of the underlying asset) can be traced back.

Finally, smart-grid demand-response systems feature distributed energy resources coordinated by tariff rules and load-shedding commands that also conceptually map to interventions. A common feature across these settings is that components exhibit explicit behavioural boundaries, interventions are applied as discrete, auditable actions, and attributing causal accountability is required.

However, these strengths come with certain limitations. For instance, the abstraction level chosen deliberately omits detailed timing or continuous-time dynamics, potentially restricting the granularity of analyses in cyber-physical

contexts. Furthermore, empirical validation through direct mappings from real-world events or operational logs to formal configurations remains a challenge.

Some directions for future work may include, an integration with explicit probabilistic reasoning or uncertainty quantification to enhance applicability to real-world scenarios, and extension of the framework toward game-theoretic analyses, explicitly incorporating strategic decision-making and equilibrium concepts. Such extensions would significantly broaden the practical utility and theoretical robustness of our framework.

Furthermore the present work deliberately leaves the question of substructurality, how resource constraints, local perspectives and non-duplicable assumptions shape causal relations, for future study. Our longer-term goal is to refine the counterfactual semantics so that an intervention is admissible only when permitted by a subsystem’s resource frame. Such a substructural extension will align naturally with our separating conjunction (capturing locality of influence) and will let us reason about responsibility and control in settings where data, authority or physical access cannot be copied, discarded or globally modified.

Finally, we remark that it would evidently be both interesting and challenging to explore the ideas presented here in the context of learning-enabled AI systems (and by extension cyber-physical systems), where questions of causality, correctness, and security are of increasing prominence: this would be a substantial programme of research in exploring a substructural approach to causal-strategic modelling.

**Acknowledgements** Chakraborty is supported by UKRI through the Centre for Doctoral Training in Cybersecurity at UCL. Caulfield and Pym acknowledge the partial support of UKRI Research Grants EP/R006865/1 and EP/S013008/1.

## References

1. Amazon Web Services: Implementing Microservices on AWS (2023), <https://docs.aws.amazon.com/pdfs/whitepapers/latest/microservices-on-aws/microservices-on-aws.pdf>, Accessed 9 June 2025
2. Anderson, G., McCusker, G., Pym, D.: A logic for the compliance budget. In: Zhu, Q., Alpcan, T., Panaousis, E., Tambe, M., Casey, W. (eds.) *Decision and Game Theory for Security [GameSec 2016]*. pp. 370–381. Springer (2016)
3. Anderson, G., Pym, D.: A calculus and logic of bunched resources and processes. *TCS* **614**, 63–96 (2016). <https://doi.org/10.1016/j.tcs.2015.11.035>
4. Aucher, G., van Benthem, J., Grossi, D.: Modal logics of sabotage revisited. *J. Log. Comput.* **28**(2), 269–303 (2017). <https://doi.org/10.1093/logcom/exx034>
5. Baier, Christel et al. : From verification to causality-based explications. In: *LIPICs 198: 48th Int. Colloq. Automata, Languages, and Programming (ICALP 2021)*. pp. 1:1–1:20 (2021). <https://doi.org/10.4230/LIPICs.ICALP.2021.1>
6. van Benthem, J., Bergstra, J.: Logic of transition systems. *J. Log. Lang. Inf.* **3**(4), 247–283 (1994). <https://doi.org/10.1007/bf01160018>
7. Blackburn, P., de Rijke, M., Venema, Y.: *Modal logic*. CUP (2001)
8. Brookes, S.: A semantics for concurrent separation logic. In: Gardner, P., Yoshida, N. (eds.) *CONCUR 2004*. pp. 16–34. Springer (2004)

9. Bujorianu, M., Caulfield, T., Ilau, M.C., Pym, D.: Interfaces in ecosystems: Concepts, form, and implementation. In: *Simulation Tools and Techniques*. pp. 27–47. Springer (2025)
10. Caulfield, T., Ilau, M.C., Pym, D.: Engineering Ecosystem Models: Semantics and Pragmatics. In: *Simulation Tools and Techniques*. pp. 236–258. Springer (2022)
11. Caulfield, T., Pym, D.: Modelling and Simulating Systems Security Policy. EAI Endorsed Trans. Sec. Safety (Proc. SIMUtools 2016, Prague) **3**(8), e3–e3 (2016)
12. Chakraborty, P., Caulfield, T., Pym, D.: Logic for decision-making about causality and influence in behavioural models of ecosystems of systems: with applications to safety and security (Accessed 5 June 2025), <http://www.cs.ucl.ac.uk/staff/D.Pym/CCP2025full.pdf>
13. Collinson, M., Monahan, B., Pym, D.: *A Discipline of Mathematical Systems Modelling*. College Publications (2012)
14. Demers, Alan et al.: Epidemic algorithms for replicated database maintenance. In: *Proc. 6th Ann. ACM Symp. on Principles of Distributed Computing*. pp. 1–12. ACM (1987). <https://doi.org/10.1145/41840.41841>
15. Financial Conduct Authority: General insurance pricing practices market study: Consultation on handbook changes, consultation Paper CP20/19\*\*\*. September 2020 (Updated December 2020)
16. Galmiche, D., Lang, T., Méry, D., Pym, D.: Bifurcation Logic: Separation Through Ordering, To appear, *Proc. TARK XX. EPTCS 2025*, Manuscript: <https://www.cantab.net/users/david.pym/current.html>
17. Galmiche, D., Lang, T., Pym, D.: Minimalistic system modelling: Behaviours, interfaces, and local reasoning, in press, *Proc 16th EAI International Conference on Simulation Tools and Techniques (SIMUtools 2024)*, Springer, 2024. Manuscript: <https://doi.org/10.48550/arXiv.2401.16109>, accessed 23/03/2025
18. Halpern, J.Y.: A modification of the halpern-pearl definition of causality. In: *Proc. 24th Int. Conf. on Artif. Intel. (IJCAI '15)*. pp. 3022–3033. AAAI Press (2015)
19. Halpern, J.Y.: *Actual Causality*. The MIT Press (2016). <https://doi.org/10.7551/mitpress/10809.001.0001>
20. Halpern, J.Y., Pearl, J.: Causes and Explanations: A Structural-Model Approach. Part I: Causes. *Brit. J. Phil. Sci.* **56**(4), 843–887 (2005)
21. Hitchcock, C.: The intransitivity of causation revealed in equations and graphs. *Journal of Philosophy* **98**(6), 273 (2001). <https://doi.org/10.2307/2678432>
22. Ikram, Azam et al.: Root cause analysis of failures in microservices through causal discovery. In: *Adv. in Neural Inf. Proc. Sys.* vol. 35, pp. 31158–31170 (2022)
23. Ishtiaq, S.S., O’Hearn, P.W.: BI as an assertion language for mutable data structures. In: *Proc. 28th ACM Symp. on Principles of Programming Languages*. pp. 14–26. ACM (2001). <https://doi.org/10.1145/360204.375719>
24. Jain, M., Pita, J., Tambe, M., Ordóñez, F., Paruchuri, P., Kraus, S.: Bayesian stackelberg games and their application for security at los angeles international airport. *SIGecom Exch.* **7**(2) (Jun 2008). <https://doi.org/10.1145/1399589.1399599>
25. Koller, D., Milch, B.: Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior* **45**(1), 181–221 (2003). [https://doi.org/doi.org/10.1016/S0899-8256\(02\)00544-4](https://doi.org/doi.org/10.1016/S0899-8256(02)00544-4)
26. Lewis, D.: Causation. *Journal of Philosophy* **70**(17), 556–567 (1973). <https://doi.org/10.2307/2025310>
27. Lin, J., Chen, P., Zheng, Z.: Microscope: Pinpoint performance issues with causal graphs in micro-service environments. In: *Service-Oriented Computing: 16th Int. Conf., ICSOC 2018, Hangzhou, China, November 12–15, 2018, Proceedings*. pp. 3–20. Springer-Verlag (2018). [https://doi.org/10.1007/978-3-030-03596-9\\_1](https://doi.org/10.1007/978-3-030-03596-9_1)

28. O’Hearn, P.W., Pym, D.J.: The Logic of Bunched Implications. *Bulletin of Symbolic Logic* **5**(2), 215–244 (1999). <https://doi.org/10.2307/421090>
29. Pardon, G., Pautasso, C., Zimmermann, O.: Consistent Disaster Recovery for Microservices: the BAC Theorem. *IEEE Cloud Computing* **5**(1), 49–59 (2018)
30. Pearl, J.: *Causality: Models, Reasoning and Inference*. Cambridge University Press, USA, 2nd edn. (2009)
31. Pearl, J.: The Causal Foundations of Structural Equation Modeling. *Handbook of Structural Equation Modeling* (12 2010)
32. Pham, Luan et al.: Root cause analysis for microservice system based on causal inference: How far are we? In: *Proc. 39th IEEE/ACM Int. Conf. Autom. Soft. Eng.* pp. 706—715. ACM, New York, NY, USA (2024)
33. van de Poel, I.: The Relation Between Forward-Looking and Backward-Looking Responsibility, pp. 37–52. Springer Netherlands, Dordrecht (2011). [https://doi.org/10.1007/978-94-007-1878-4\\_3](https://doi.org/10.1007/978-94-007-1878-4_3)
34. Read, S.: *Relevant Logic*. Blackwell (1988)
35. Shoham, Y., Leyton-Brown, K.: *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press (2008)
36. Simon, H.A.: *The Sciences of the Artificial* (3rd ed.). MIT Press (1996)
37. Simon, H.A., Barnard, C.I.: *Administrative behavior: a study of decision-making processes in administrative organization*. Macmillan Co., New York (1947)
38. Spirtes, P., Glymour, C., N., S., Richard: *Causation, Prediction, and Search*. MIT Press (1993)
39. Stirling, C.: *Modal and Temporal properties of Processes*. Springer (2001)
40. Sulis, W.: Mathematics of a process algebra inspired by whitehead’s process and reality: A review. *Mathematics* **12** (2024), <https://doi.org/10.3390/math12131988>
41. Tambe, M.e.a.: Building dynamic agent organizations in cyberspace. *IEEE Internet Computing* **4**(2), 65–73 (2000). <https://doi.org/10.1109/4236.832948>
42. Wang, Hanzhang et al.: Groot. In: *Proc. 36th IEEE/ACM Int. Conf. Autom. Soft. Eng.* pp. 419–429. IEEE (2022). <https://doi.org/10.1109/ASE51524.2021.9678708>
43. Winskel, G.: Events, causality and symmetry. *The Computer Journal* **54**(1), 42–57 (06 2009). <https://doi.org/10.1093/comjnl/bxp052>
44. Yao, Zhenhe et al.: Chain-of-event: Interpretable root cause analysis for microservices through automatically learning weighted event causal graph. In: *Compan. Proc. 32nd ACM Int. Conf. Found. Soft. Eng.* pp. 50–61. ACM (2024). <https://doi.org/10.1145/3663529.3663827>
45. Yousif, M.: Microservices. *IEEE Cloud Computing* **3**(5), 4–5 (2016). <https://doi.org/10.1109/MCC.2016.101>

## 8 Appendix

First, the proofs of the results required to establish the operational–logical equivalence — van Benthem-Bergstra-Hennessy-Milner — properties are given. Then, the results are extended to causal notions over systems (such as actual causality) and the corresponding metatheoretical framework. Halpern’s three criteria for characterizing actual causal relationship are also mentioned.

### 8.1 Equivalence

We first present the full definition of our model-changing notion of bisimulation. Recall that a pointed system model is a pair  $(\mathcal{M}, f)$ , where  $\mathcal{M}$  is a system model and  $f$  is a configuration in  $\mathcal{M}$ . Moreover as mentioned in remark 3,  $R_\Theta$  denotes a relation on the class of pointed system models that relates two such models when one is derived from the other through the application of an intervention operation  $\theta$ .

**Definition 13.** *Two interface-admitting pointed system models,*

$$(\mathcal{M}_1, f_1) = (F_1, \Delta_{\mathcal{I}_1}, \Gamma, f_1) \text{ and } (\mathcal{M}_2, f_2) = (F_2, \Delta_{\mathcal{I}_2}, \Gamma_2, f_2),$$

*are bisimilar under intervention if the following conditions are satisfied:*

1. **(Atom):** *For any atomic proposition  $p$ ,*

$$(\mathcal{M}_1, f_1) \models p \text{ if and only if } (\mathcal{M}_2, f_2) \models p$$

2. **(Zig):** *If  $f_1 \Delta_{\mathcal{I}_1} f'_1$ , then there exists  $f'_2 \in F_2$  such that  $f_2 \Delta_{\mathcal{I}_2} f'_2$  and  $(\mathcal{M}_1, f'_1)$  and  $(\mathcal{M}_2, f'_2)$  are bisimilar.*
3. **(Zag):** *If  $f_2 \Delta_{\mathcal{I}_2} f'_2$ , then there exists  $f'_1 \in F_1$  such that  $f_1 \Delta_{\mathcal{I}_1} f'_1$  and  $(\mathcal{M}_2, f'_2)$  and  $(\mathcal{M}_1, f'_1)$  are bisimilar.*
4. **(Zig $_\Theta$ ):** *If  $(\mathcal{M}_1, f_1) R_\Theta (\mathcal{M}'_1, f_1)$  then there exists  $(\mathcal{M}'_2, f_2)$  such that*

$$(\mathcal{M}_2, f_2) R_\Theta (\mathcal{M}'_2, f_2) \text{ and } (\mathcal{M}'_1, f_1) \text{ and } (\mathcal{M}'_2, f_2)$$

*are bisimilar under intervention.*

5. **(Zag $_\Theta$ ):** *If  $(\mathcal{M}_2, f_2) R_\Theta (\mathcal{M}'_2, f_2)$ , then there exists  $(\mathcal{M}'_1, f_1)$  such that*

$$(\mathcal{M}_1, f_1) R_\Theta (\mathcal{M}'_1, f_1) \text{ and } (\mathcal{M}'_2, f_2) \text{ and } (\mathcal{M}'_1, f_1)$$

*are bisimilar under intervention.* □

The two van Benthem-Bergstra-Hennessy-Milner completeness theorems mentioned in 6 are proved below. We provide a proof sketch omitting tedious details.

**Theorem 4.** *If two interface-admitting pointed system models,  $(\mathcal{M}_1, f_1)$  and  $(\mathcal{M}_2, f_2)$  are bisimilar under intervention, then for all formulae  $\varphi \in L(\langle\theta\rangle, *)$ , if  $(\mathcal{M}_1, f_1) \models \varphi$ , then  $(\mathcal{M}_2, f_2) \models \varphi$ .*

*Proof.* Let  $\varphi \in \mathcal{L}(\langle\theta\rangle, *)$ . The proof is by induction on the syntax of  $\varphi$ . First suppose that  $\varphi$  contains no connectives. The atom clause in definition of bisimulation under intervention covers the atomic propositions. Our induction hypothesis is that the implication holds for all formulae containing at most  $n$  ( $n \geq 0$ ) boolean connectives and modal operators. We must now show that the implication holds for all formulae  $\varphi$  containing  $n+1$  connectives and operators. Consider the case when  $\varphi$  contains no modal operators. If  $\varphi$  is of the form  $\neg\psi$  then by the induction

hypothesis the implication is immediate. If  $\varphi = \psi_1 \wedge \psi_2$ , then, by the induction hypothesis, both  $(\mathcal{M}_2, f_2) \models \psi_1$  and  $(\mathcal{M}_2, f_2) \models \psi_2$ , and thereby  $(\mathcal{M}_2, f_2) \models \varphi$ .

Consider the case when  $\varphi = \Diamond\psi$ . Assume that  $(\mathcal{M}_1, f_1) \models \Diamond\psi$ . By the semantics of  $\Diamond$ , there exists  $f_1 \Delta_{\mathcal{I}_1} f'_1$  such that  $(\mathcal{M}_1, f'_1) \models \psi$ . By clause **Zig** $_{\Diamond}$  in the definition of bisimulation under intervention, it follows that there exists  $f'_2$  such that  $f_2 \Delta_{\mathcal{I}_2} f'_2$ , and  $(\mathcal{M}_1, f'_1)$  and  $(\mathcal{M}_2, f'_2)$  are bisimilar under intervention. By the induction hypothesis, we conclude that  $(\mathcal{M}_2, f'_2) \models \psi$ , and consequently  $(\mathcal{M}_2, f_2) \models \varphi$ . Similarly, from  $(\mathcal{M}_2, f_2) \models \Diamond\psi$  we conclude  $(\mathcal{M}_1, f_1) \models \Diamond\psi$  by the **Zag** $_{\Diamond}$  clause. The case when  $\varphi = \Box\psi$  is similar.

Now, consider  $\varphi = \psi_1 * \psi_2$  where  $\psi_1$  and  $\psi_2$  are star-free formulae. Assume  $(\mathcal{M}_1, f_1) \models \varphi$ . Then there exists a pair of pointed partial models  $(\mathcal{M}_1^a, f_1^a)$  and  $(\mathcal{M}_2^b, f_1^b)$  s.t.  $(\mathcal{M}_1^a, f_1^a) \models \psi_1$  and  $(\mathcal{M}_1^b, f_1^b) \models \psi_2$ . By the premiss of this theorem and since the models admit of interface, we have that there exist two bisimulations under intervention such that the pairs  $(\mathcal{M}_1^a, f_1^a)$ ,  $(\mathcal{M}_2^a, f_2^a)$  and  $(\mathcal{M}_1^b, f_1^b)$ ,  $(\mathcal{M}_2^b, f_2^b)$  each are bisimilar, and the pair  $\{(\mathcal{M}_2^a, f_2^a), (\mathcal{M}_2^b, f_2^b)\}$  is a conjugate decomposition of  $(\mathcal{M}_2, f_2)$ . Since  $\psi_1$  and  $\psi_2$  are star-free, it follows that  $(\mathcal{M}_2^a, f_2^a) \models \psi_1$  and  $(\mathcal{M}_2^b, f_2^b) \models \psi_2$ , and consequently  $(\mathcal{M}_2, f_2) \models \varphi$ .

Consider the case when  $\varphi = \langle\theta\rangle\psi$ . Assume  $(\mathcal{M}_1, f_1) \models \varphi$ . By the semantics of intervention operator there exists  $(\mathcal{M}'_1, f_1)$  such that  $(\mathcal{M}'_1, f_1) \models \psi$ , and  $(\mathcal{M}_1, f_1)R_{\theta}(\mathcal{M}'_1, f_1)$  for some intervention  $\theta$ . From the **Zig** $_{\theta}$  clause, it follows that there exists  $\mathcal{M}'_2$  such that  $(\mathcal{M}_2, f_2)R_{\theta}(\mathcal{M}'_2, f_2)$ , and  $(\mathcal{M}'_1, f_1)$  and  $(\mathcal{M}'_2, f_2)$  are bisimilar under intervention. By the induction hypothesis, we conclude that  $(\mathcal{M}'_2, f_2) \models \psi$ , and thence  $(\mathcal{M}_2, f_2) \models \varphi$ . The converse follows similarly via **Zag** $_{\theta}$  clause.  $\square$

The other theorem establishes the converse under specific restrictions on the language and for the subclass of interface-admitting system models with finitely many components and behaviours.

**Theorem 5.** *Given a formula  $\varphi \in \mathcal{L}(\langle\theta\rangle)$ , for any two interface-admitting pointed system models  $(\mathcal{M}_1, f_1)$  and  $(\mathcal{M}_2, f_2)$  with finitely many components and behaviours, if  $(\mathcal{M}_1, f_1) \models \varphi$  implies  $(\mathcal{M}_2, f_1) \models \varphi$ , then there exists a bisimulation under intervention relating  $(\mathcal{M}_1, f_1)$  and  $(\mathcal{M}_2, f_2)$ .*

*Proof.* Let  $(\mathcal{M}_1, f_1) = (F_1, \Delta_{\mathcal{I}_1}, \Gamma, f_1)$  and  $(\mathcal{M}_2, f_2) = (F_2, \Delta_{\mathcal{I}_2}, \Gamma_2, f_2)$  be the two pointed models over a set of components  $\mathcal{C}$ . Since  $|\mathcal{C}|$  is finite, there are only finitely many  $f'_1$  and  $f'_2$  such that  $f_1 \Delta_{\mathcal{I}_1} f'_1$ , and  $f_2 \Delta_{\mathcal{I}_2} f'_2$ . Similarly, there are only finitely many interventions  $\theta$  possible over the subsets of  $\mathcal{C}$ . Therefore given a pointed model  $(\mathcal{M}, f)$ , there will be only finitely many  $(\mathcal{M}', f)$  such that  $(\mathcal{M}, f)R_{\theta}(\mathcal{M}', f)$  for some intervention  $\theta$ .

For all  $\varphi \in \mathcal{L}(\langle\theta\rangle)$ ,  $(\mathcal{M}_1, f_1) \models \varphi$  iff  $(\mathcal{M}_2, f_2) \models \varphi$  (by assumption), and in particular, for all atomic propositions  $p$ ,  $(\mathcal{M}_1, f_1) \models p$  iff  $(\mathcal{M}_2, f_2) \models p$ .

We show the following contradiction: Let  $(\mathcal{M}_1, f_1)R_{\theta}(\mathcal{M}'_1, f_1)$ . Assume that there is no  $\mathcal{M}'_2$  such that for all  $\varphi$ ,  $(\mathcal{M}'_1, f_1) \models \varphi$  iff  $(\mathcal{M}'_2, f_2) \models \varphi$ . Let  $S = \{\mathcal{N}_2 \mid (\mathcal{M}_2, f_2)R_{\theta}(\mathcal{N}_2, f_2)\}$ .  $S$  is neither non-empty nor infinite (since there are only finitely many interventions possible). Thus  $S = \{\mathcal{N}_2^1, \dots, \mathcal{N}_2^n\}$ . By assumption, for every  $\mathcal{N}_2^i \in S$ , there exists a formula  $\psi^i$  such that  $(\mathcal{M}'_1, f_1) \models \psi^i$

and  $(\mathcal{N}_2^i, f_2) \not\models \psi^i$ . But then there is a formula  $\psi^i$  for each  $\mathcal{N}_2^i$  such that  $(\mathcal{M}_1, f_1) \models \bigwedge_{i \in \{1, \dots, n\}} \langle \theta \rangle \psi^i$ , but  $(\mathcal{M}_2, f_2) \not\models \bigwedge_{i \in \{1, \dots, n\}} \langle \theta \rangle \psi^i$  which is a contradiction.

Similarly, the converse clause can be shown. Moreover, two similar contradictions with regards to the corresponding  $\Delta$  relations, establish the standard zig and zag clauses (which follows from the fact there are only finitely many  $f'_1$  and  $f'_2$  such that  $f_1 \Delta_{\mathcal{I}_1} f'_1$  and  $f_2 \Delta_{\mathcal{I}_2} f'_2$ ). Thus if for all  $\varphi \in \mathcal{L}(\langle \theta \rangle)$ ,  $(\mathcal{M}_1, f_1) \models \varphi$  iff  $(\mathcal{M}_2, f_2) \models \varphi$ , then  $(\mathcal{M}_1, f_1)$  and  $(\mathcal{M}_2, f_2)$  are bisimilar under intervention.  $\square$

## 8.2 Actual cause

In the sequel, we provide the definition of actual cause as found in Chapter 2 of Halpern's Actual Causality [19]. As mentioned previously, we used the **AC2**( $a^m$ ) clause.

**Definition 14.** *Given a causal setting  $(M, \vec{u})$ ,  $\vec{X} = \vec{x}$  is an actual cause of  $\varphi$  if the following three conditions hold:*

**AC1.**  $(M, \vec{u}) \models (\vec{X} = \vec{x})$  and  $(M, \vec{u}) \models \varphi$ .

**AC2**( $a^m$ ). *There is a set  $\vec{W}$  of variables in  $V$  and a setting  $\vec{x}$  of the variables in  $\vec{X}$  such that if  $(M, \vec{u}) \models \vec{W} = \vec{w}^*$ , then  $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}, \vec{W} \leftarrow \vec{w}^*] \neg \varphi$*

**AC3.**  $\vec{X}$  is minimal; there is no strict subset  $\vec{X}'$  of  $\vec{X}$  such that  $\vec{X}' = \vec{x}'$  satisfies conditions AC1 and AC2, where  $\vec{x}'$  is the restriction of  $\vec{x}$  to the variables in  $\vec{X}'$ .  $\square$

We now give the proof of Lemma 1 which relates interventions in Causal System Models:

**Lemma 2 (Characterization of Interventions in Causal System Models).** *Let  $\mathcal{M} = (F, \Delta_{\mathcal{I}}, \Gamma)$  be a causal system model with causal projection  $\mathcal{M}^c = (F^c, \Delta_{\mathcal{I}}^c, \Gamma^c)$ . Let  $\theta = (C', I'_{C'})$  be an intervention yielding the intervened system  $\mathcal{M}_\theta = (F, \Delta_{\mathcal{I}_\theta}, \Gamma)$ . The following holds:*

1. **Causal Preservation Criterion:** *If a causal chain  $(f_1, \dots, f_n) \in \text{Chain}(\mathcal{M})$  does not involve any component in  $C'$  as part of the cause for any transition, then this chain is preserved under intervention  $\theta$ . Formally, we have that  $\forall i (1 \leq i < n), \quad C' \cap \text{Cause}(f_i, f_{i+1}) = \emptyset \Rightarrow (f_1, \dots, f_n) \in \text{Chain}(\mathcal{M}_\theta)$ .*
2. **Causal Disruption Criterion:** *If a causal chain  $(f_1, \dots, f_n) \in \text{Chain}(\mathcal{M})$  contains a configuration  $f_i$  whose cause involves components in  $C'$ , and the intervention  $\theta$  modifies the influence rules such that the causal transition to  $f_{i+1}$  is invalidated, then the chain is disrupted. Formally, we have that  $\exists i (1 \leq i < n)$  such that  $C' \cap \text{Cause}(f_i, f_{i+1}) \neq \emptyset$  and  $\langle \theta \rangle \neg (f_i \Delta_{\mathcal{I}_\theta} f_{i+1})$ .*

*Proof.* Let  $\mathcal{M} = (F, \Delta_{\mathcal{I}}, \Gamma)$  be a causal system with causal projection  $\mathcal{M}^c = (F^c, \Delta_{\mathcal{I}}^c, \Gamma^c)$ , and let  $\theta = (C', I'_{C'})$  be an intervention yielding the intervened system  $\mathcal{M}_\theta = (F, \Delta_{\mathcal{I}_\theta}, \Gamma)$ .

Let  $(f_1, \dots, f_n) \in \text{Chain}(\mathcal{M})$  be an arbitrary causal chain in the original system. Assume that none of the configurations in the chain  $(f_1, \dots, f_n)$  involve

any component from  $C'$  as part of their cause. By the definition of intervention, if  $C'$  is disjoint from the cause set of every transition in the chain, the influence rules governing those transitions remain unchanged. Consequently, the transition relation  $\Delta_{\mathcal{I}_\theta}$  remains identical to  $\Delta_{\mathcal{I}}$  for these configurations. Therefore, the transitions  $(f_i, f_{i+1})$  are preserved, meaning the entire chain  $(f_1, \dots, f_n)$  persists in  $\mathcal{M}_\theta$ . Thus, the causal chain is preserved under the intervention.

Now assume that the chain  $(f_1, \dots, f_n)$  involves a configuration  $f_i$  whose cause depends on components in  $C'$ . Suppose the intervention  $\theta$  modifies the influence rules such that the cause of  $f_{i+1}$  is invalidated. By the definition of intervention, the updated influence rule  $I'_{C'}$  modifies how components in  $C'$  contribute to transitions. If the intervention disrupts the causal condition required for  $f_i$  to transition to  $f_{i+1}$ , then the transition  $(f_i, f_{i+1})$  no longer exists in  $\Delta_{\mathcal{I}_\theta}$ . Formally, this is captured by  $\langle \theta \rangle \neg (f_i \Delta_{\mathcal{I}_\theta} f_{i+1})$ . Consequently, the entire causal chain  $(f_1, \dots, f_n)$  is disrupted, as the sequence of causally linked configurations is broken. Thus, the causal disruption criterion holds.  $\square$

We now prove Theorem 1. As before, we omit the tedious details.

**Theorem 6.** *Let  $\mathcal{M} = (F, \Delta_{\mathcal{I}}, \Gamma)$  be a causal system model over a finite component set  $\mathcal{C}$ . Construct a corresponding HP causal model  $M = \langle U, V, F \rangle$ , where  $V = \{V_c \mid c \in \mathcal{C}\}$  with  $\text{dom}(V_c) = \mathbb{B}(c)$  and the structural equations in  $F$  are induced by the influence rules  $\mathcal{I}$  of  $\mathcal{M}$ . For any configuration  $f_2 \in F$ , define the effect formula  $\varphi_{f_2} = \bigwedge_{c \in \mathcal{C}} (V_c = f_2(c))$ . Then, if there exists a causal chain in  $\mathcal{M}$  from an initial configuration  $f_1$  to  $f_2$ , one can extract a candidate cause, that is, a subset  $\vec{X} \subseteq V$  and an assignment  $\vec{x}$  (with a corresponding context  $\vec{u}$ ) such that the assignment  $\vec{X} = \vec{x}$  satisfies the HP criteria (AC1–AC3) for being an actual cause of  $\varphi_{f_2}$  in  $(M, \vec{u})$ . In other words, the existence of a causal chain from  $f_1$  to  $f_2$  in  $\mathcal{M}$  implies that there is a corresponding actual cause in the HP model.*

*Proof.* Let  $\mathcal{M} = (F, \Delta_{\mathcal{I}}, \Gamma)$  be a causal system model over a finite component set  $\mathcal{C}$  and assume there exists a causal chain  $(f_1, f_2, \dots, f_n)$  in  $\mathcal{M}$  with  $f_1$  as the initial configuration and  $f_n = f_2$  as the outcome. By definition of a causal chain, each transition  $f_i \Delta_{\mathcal{I}} f_{i+1}$  is justified by the fact that some subset  $\mathcal{C}_i \subseteq \mathcal{C}$  acts as a cause for the change from  $f_i$  to  $f_{i+1}$ , while remaining unchanged, and such that altering that subset prevents the transition (counterfactual dependence), with minimality ensured by discarding any superfluous components.

We now construct a corresponding HP model  $M = \langle U, V, F \rangle$ , where the set of endogenous variables is  $V = \{V_c \mid c \in \mathcal{C}\}$ , with  $\text{dom}(V_c) = \mathbb{B}(c)$ , and the structural equations in  $F$  are induced directly by the influence rules  $\mathcal{I}$  (i.e., for each  $c$ , the equation for  $V_c$  is given by a function  $f_c$  reflecting  $\mathcal{I}_c$  and depending on the values of the variables corresponding to the influence context  $\text{Inf}(c)$ ). Choose  $U$  so that a fixed initial assignment  $\vec{u}$  yields the starting configuration  $f_1$ .

By the premiss of this theorem, we define the effect formula  $\varphi_{f_2} = \bigwedge_{c \in \mathcal{C}} (V_c = f_2(c))$ . Since the causal chain in  $\mathcal{M}$  guarantees that  $f_2$  is reached from  $f_1$  via a series of transitions that satisfy the regularity, counterfactual, and minimality

conditions, we can extract a candidate cause. In particular, there exists some subset  $\vec{X} \subseteq V$ , corresponding to the union of the causal subsets  $\mathcal{C}_i$  from each transition (or a minimal such subset), and an assignment  $\vec{x}$  such that:

1. In the causal setting  $(M, \vec{u})$ , the assignment  $\vec{X} = \vec{x}$  holds, and  $M$  under  $\vec{u}$  satisfies  $\varphi_{f_2}$ .
2. If we intervene to alter the values of  $\vec{X}$  (while keeping the values for a suitable witness set fixed), then the structural equations in  $F$  imply that the effect  $\varphi_{f_2}$  would not obtain (i.e., for every configuration reachable under the intervention,  $\varphi_{f_2}$  fails). This follows directly from the counterfactual condition in the causal chain.
3. By the minimality condition in the causal chain, no strict subset of  $\vec{X}$  would suffice to guarantee the effect under the counterfactual analysis.

Therefore, the candidate cause  $(\vec{X} = \vec{x})$  extracted from the causal chain in  $\mathcal{M}$  satisfies the HP conditions (AC1–AC3) for being an actual cause of  $\varphi_{f_2}$  in the HP model  $(M, \vec{u})$ . This completes the proof that the existence of a causal chain from  $f_1$  to  $f_2$  in  $\mathcal{M}$  implies the existence of a corresponding actual cause in  $M$ .  $\square$