Causality for Cyber-Physical Systems

Hugo Araujo HUGO.ARAUJO@KCL.AC.UK
Hana Chockler HANA.CHOCKLER@KCL.AC.UK
Mohammad Reza Mousavi MOHAMMAD.MOUSAVI@KCL.AC.UK
King's College London

Gustavo Carvalho GHPC@CIN.UFPE.BR
Augusto Sampaio ACAS@CIN.UFPE.BR
Universidade Federal de Pernambuco

Abstract

We present a formal theory for analysing causality in cyber-physical systems. To this end, we extend the theory of actual causality by Halpern and Pearl to cope with the continuous nature of cyber-physical systems. Based on our theory, we develop an analysis technique that is used to uncover the causes for examples of failures resulting from verification, which are represented as continuous trajectories. We develop a search-based technique to efficiently produce such causes and provide an implementation for such a technique. Moreover, we apply our solution to case studies (a suspension system and a connected platoon) and benchmark systems to evaluate its effectiveness; in the experiment, we show that we were able to detect causes for inserted faults.

1. Introduction

Cyber-physical systems (CPSs) are systems that integrate computation with physical processes, in contexts where communication networks and human interaction may be present. Embedded computers and networked embedded components in CPSs monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa [46]. In CPSs, components often operate in both spatial and temporal dimensions, where there is an intense link between physical and computational elements.

The importance of reliability in safety-critical CPSs warrants further research into their verification. Particularly, there is a need to devise verification techniques based on mathematical relations, e.g., notions of conformance [59], that decide whether the system behaves as expected. Conformance notions are employed by conformance testing processes to evaluate whether a system behaves as required by its specification. Conformance testing and falsification approaches [63, 3, 64, 65, 59, 7, 24, 68, 50] have the benefit of offering mathematical assurances about the correctness of the system.

Although they are undoubtedly useful, verification techniques can be of limited use, particularly in the context of complex systems, without a systematic way to trace the failures back to their original causes. Testing a system without a method for locating the source of the discovered faults can lead to an incredible amount of resources spent on manual system inspections.

In this context, causal analysis is an essential ingredient of such rigorous verification techniques, providing effective means to isolate and eventually remove the faults causing hazards and failures observed in the verification process. There have been many attempts to

define and analyse causality. In a philosophical context, causality is defined the relationship between events where the the cause contributes to the realisation of a different event (the effect), where the cause is (partially) responsible for the effect, which is (partially) dependent on its cause [66]. In a more practical context, Halpern and Pearl have made use of structural equation models to provide a mathematical definition of actual causality [39, 40]. Chockler builds upon this work by integrating the notion of actual cause into a rigorous verification process [22].

However, most definitions of actual causality only cater for discrete systems and do not cover the complexity of dynamic physical phenomena. Due to the continuous and quantitative nature of the physical parts of CPSs, the notion of causality is bound to have a quantitative nature and accommodate physical dynamics.

1.1 Problem definition and contributions

Our research goal is to propose a formal theory and a practical analysis technique for checking causality that can cope with the complexity of CPSs. The main contributions of this work are the following:

- We extend the theory of actual causality [39] to cope with CPSs. The new theory takes time and continuous dynamics into consideration.
- We develop a process for checking causality and mechanise the associated algorithms in MATLAB and integrate them with our tool, HyConf [8], for conformance testing of CPSs.
- We propose and discuss practical applications of causality in the verification of CPSs.
- We apply the developed algorithms and tools to a range of case studies and evaluate the effectiveness of our strategy.

Applying a systematic and formal methodology to determine the cause of a failure in a CPS requires structured notions of faults and causes using mathematical notations. We work in a mathematical framework where faults are represented using first-order formulae, in which atomic predicates are expressed over signal values (akin to safety subset of Signal Temporal Logic [48]). Causes are expressed using intervals of trajectories, which we call *trajectory slices*, of the system variables leading to the falsification of such properties.

Causes are determined using a causal model, which is a representation of the system dynamics using structural equations. In this model, we separate variables into endogenous and exogenous sets; the former represents the set of variables that are chosen to undergo the causal analysis and can be seen as potential causes. The latter are variables that affect the system but are not seen as potential causes. In a practical setting, having too many endogenous variables greatly increases the costs of the causal analysis in CPSs. Hence, the choice of endogenous and exogenous variables is left to the user and has an impact on the variables that can appear in the cause and the efficiency of the causal analysis.

Furthermore, for a notion of causality for CPSs, one needs to take time into consideration. For instance, faults may have been caused by trajectory slices occurring at specific time intervals. That is, if these trajectory slices had occurred in different intervals, the fault

may not have happened. Thus, one cannot ignore the history of the system execution. This way, we aim to identify not only to which components the fault should be attributed, but also provide the intervals of time when the causal behaviour has occurred in the respective components.

Finally, one must consider the continuous and infinite nature of physical phenomena. The way system variables affect each other can determine whether a cause comprises one or multiple interacting variables. It is possible that multiple alternative causes can be found for a single failure.

We mechanised our causal analysis using the Matlab/Simulink framework, which is a commonly used environment for modelling and analysis of control systems, thus increasing the accessibility of our strategy to the CPSs community. We developed a search-based algorithm to find the causes and integrated the mechanisation of our theory of actual causality into our pre-existing tool for testing cyber-physical systems [8]. We applied the developed algorithms and tools to a range of case studies to demonstrate the effectiveness of our strategy, where we pinpoint causes and present explanations for failures that have been detected in the verification process.

1.2 Structure of the paper

This paper is organised as follows. In Section 2, we present the related work. In Section 3, we discuss the preliminaries used to develop our framework. In Section 4, we present our extension to the theory of causality that considers time and continuous dynamics and, in Section 5, we develop the mechanisation of this extended theory. In Section 6, we design and conduct experiments to show how our strategy can be applied to complex systems. Finally, in Section 7, we draw some conclusions and point out the directions of our future research.

2. Related work

Our context is the theory of actual causality [38] where, in a given scenario leading to an outcome, the events are analysed in order to find causes. This is also called token-level causality, which concerns causal relations regarding particular events and settings. It is in contrast with type-level causality [41] where general causal rules governing a system are sought.

There are a few variations of the definition of actual causality. The original one is by Halpern and Pearl [38], which is followed by two variants (updated and modified) by Halpern and Pearl, and Halpern alone, respectively [39, 36]; our study is based on the definition by Halpern and Pearl in 2015 [36], which is called the *modified* definition. In the most recent version of the theory [36], Halpern simplifies the impact of contingencies, which can be seen as pre-conditions for a cause but not part of the cause itself. More specifically, the key difference between the *modified* definition and the other definitions is the requirement that the contingencies in the modified version should be set to their initial values, whereas in the updated definition, contingencies can be set to different values.

Halpern and Pearl also introduced the notion of time-indexed endogenous variables [39]; this treatment of time in the original theory of Halpern is discrete in nature and does not cater for continuous time and dynamics that are necessary to model cyber-physical

systems. In order to use this definition of Halpern and Pearl, one needs to define a fixed discretisation of continuous variables and come up with a causal model explaining their relationships. Our theory, however, provides an abstraction layer that works directly with the continuous specification of the dynamics and builds the necessary sampling as a part of the analysis. In a recent extension of their work, Peters and Halpern propose the generalised structural equations models (GSEM) [58] where a signature comprises a set of interventions and the equations map an intervention to a set of outcomes (e.g., when variable X is set to x, then variable Y is equal to y and variable Z is equal to z). It brings about the possibility of having an infinite set of valuations for variables. Furthermore, Halpern shows that this can be adapted to ordinary differential equations; in his treatment, instead of having a set of endogenous variables, the signature comprises the value of a variable at any moment in time, as defined by differential equations. This recent extension may be a theoretical alternative to our proposed framework; however, to apply any practical causal analysis on this infinite set of variables, further abstractions and algorithmic procedures need to be developed. In principle, our work can be adapted to deal with the many variants of Halpern and Pearl's theory of actual causality.

Apart from the variants developed by Halpern and Pearl, there are many other definitions of actual causality by others. In the remainder of this section, we review some of the most relevant variants or applications of actual causality developed by other researchers. Subsequently, we also mention some alternative theories, to actual causality, that can be used and extended for finding root causes in CPSs.

Several works employ the definitions of causality by Halpern and Pearl to formal verification; often as a reasoning tool for explaining counterexamples in the discrete domain. The most relevant to our work are discussed below.

Baier et al. [10] have conducted a survey on published approaches that utilise Halpern-Pearls's notion of causality. More precisely, they look into formal approaches to probabilistic causation that can explain observable behaviour in reactive systems.

Within the context of cyber-physical systems, Deng et al. [23] have proposed a temporal logic for analysing causality called causal temporal logic. Once the prospective causes and effects are expressed in this temporal logic, they assess causality by generating traces that can satisfy/violate the causal formula and, thus, calculating the degree of sufficiency and necessity. There are key differences between our work and theirs. Firstly, we propose a conservative extension of Halpern-Pearls theory, in which we aim to keep the nuances of their theory (such as the notions of contingency and causal path, which are key factors to consider when determining actual causality) and not just the notions of sufficiency and necessity. Furthermore, their work assumes access to the correct behaviour and prospective causes. In our work, we do not need access to the correct behaviour; we search for causes by making use of meta-heuristics and looking for behaviours that can violate/satisfy the effect (formula).

Leitner-Fischer and Leue [47] define a theory of causality that considers the temporal order as well as the non-occurrence of events. They also provide a search-based on-the-fly causality assessment that does not require the counterexamples to be generated in advance. There, even though the order of events is important, no concrete notion of time is introduced. In our work, however, real time plays an important role and trajectories are represented in the continuous time domain.

Caltais, Mousavi, and Singh [18] define a theory of actual causality for labelled transition systems. Their formalisation is inspired by the definition of Halpern and Pearl. Their main result is a theory to explain counterexamples in model checking with respect to properties in Hennesy-Milner Logic [42]. They mechanise their theory in a prototype tool, which interacts with the mCRL2 model-checker [34] in order to check the various conditions in their definition of actual causality. Our work shares a similar nature of employing causality to interpret traces leading to failures. However, we employ models and logic that consider quantitative aspects, such as trajectories that are solutions to systems of differential equations.

Ibrahim et al. [43] provide a process to convert attack trees, fault trees, and timed failure propagation graphs for CPSs into Halpern-Pearl causal models. They illustrate their approach using an Unmanned Aerial Vehicle case study. Even though their work, like ours, focuses on CPS-related aspects, it does not handle continuous aspects of such systems.

In the context of hardware verification, Chockler, Grumberg, and Yadgar [21] employ a notion of responsibility (degree of causality) [22] to improve the quality of abstraction refinement by producing mode efficient counterexamples. Besides the continuous aspects, our approach incorporates the modelling of platform (hardware), controllers (software) and environment into a single model that considers a high-level abstraction of the system. We do not consider such a notion of responsibility, however; this is one of the directions for our future work.

There are quantitative extensions of the theory of actual causality. Pearl studies the effect of causality in probabilistic systems [54, 55], providing the underlying theory for causal inference [61, 57], which provides the mathematical tools and the language for articulating probabilities of causation. His work employs structural equations to cope with counterfactuals and randomisation and has applications in AI [56]. Baier et al. [12, 10] introduce and formalise cause-effect relation in Markov decision processes using the probability-raising principle. They provide algorithms for checking cause-effect relationships and the existence of probability-raising causes for given effect scenarios. To our knowledge their work does not use counter-factual reasoning and is not formally related to our formal theory of actual causality. With respect to the use of causal analysis in the formal verification process itself (and not just as a way to explain the results), Baier et al. [11] have presented a temporal logic characterisation for the notions of sufficiency and necessity, which are based on the concepts found in causal reasoning. They propose an optimisation algorithm for the computation of causes based on these degrees of necessity and sufficiency. Unlike our work, however, they do not cater for the continuous properties of physical systems but they do however consider stochastic aspects. We consider extending our results to quantitative and probabilistic notions as a worthwhile future direction.

Zhang et al. [67] propose a method for the online monitoring of Signal Temporal Logic (STL). Instead of computing the distance of the system's output against the specification (called robustness value [29], also in our approach), they compute whether an "instant" is relevant to a violation and how far the instant is from a violation. This bears some resemblence to our notion of actual cause. However, their work, unlike ours, does not formally relate to a theory of actual causality and does not pinpoint a cause by tying specific variables to specific time intervals.

Beer et al. [14] use actual causality to explain counterexamples in hardware verification for Linear Temporal Logic properties. The proposed algorithm is implemented in the IBM RyleBase PE tool, where causality is applied to traces but ignores the system model from which the traces originated. Our assessment of causality considers both the setting and the model, which in our case also encompasses continuous and discrete aspects. This richer setting leads to a more computationally-intensive analysis but allows us to find causes in more complex systems and with more precision (steered by the model).

Dubslaff et al. [25] use counterfactual reasoning to identify causes in configurable systems. This is done by identifying the features and interactions that are the reason for emerging functional and non-functional properties. They call this concept feature causality. These notions are in clear contrast with the notion employed in our work where we start with a concrete scenario leading to the effect. Our choice is justified by our context, where we employ causal analysis as a step in the testing and verification process, where finding an error-trace or counterexample initiates the causal analysis process.

There are other theories of causality that are not concerned with a particular scenario. For example, Granger's causality [33] is a statistical concept that checks the possibility of a time series predicting another. It employs notions of trends, seasonal patterns and noise in time series forecasting and can be applied to machine learning, finance and weather forecast [60].

Table 1 summarises the key difference between the above-surveyed techniques and our proposed framework by showing the supported features for each technique.

Table 1: A comparison between different methods for causal analysis.

	Formal Models	**Counterexamples**	**Hybrid Systems**	**Mechanisation**	**Search-based**	**Stochastic**
Ours	✓	✓	✓	✓	✓	
Leitner-Fischer [47]				✓	✓	
Caltais [18]	✓	✓		✓		
Chockler [21]		✓		✓		
Beer [14]	✓	✓		✓		
Pearl [56]	✓	✓				✓
Granger [33]	✓			✓		
Deng [23]	✓		✓	✓	✓	
Zhang[67]	✓		✓	✓		
Peters [58]	✓	✓	✓			
Baier [12, 10]	✓	✓		✓		✓

There are fault localisation techniques that are not causality-based. For instance, fault-tree analysis [27] is an established method to investigate faults in safety critical systems. In fault trees, a graphical representation captures the logical connections between faults and their origins. Similar to our causal analysis, fault tree analysis starts from a failure event, which is represented at the top of the tree and then it is worked backwards to determine the root causes. In order to cope with dynamic systems, fault trees have been extended to include quantitative dependability analysis. Several approaches have been proposed and used such as dynamic fault trees [26], state-event fault trees [45], spectrum-based fault localisation [4], and Stochastic Hybrid Fault Tree Automaton [20]. Temporal Fault Trees [53] are one such extension, which, to our knowledge, can only handle discrete time. There are other approaches that define continuous semantics for Dynamic Fault Trees in terms of Timed Markov Chains [17, 44]. Our approach has a number of general advantages over these

approaches: (i) we provide a rigorous definition of causality and base our fault localisation on this definition, (ii) we directly employ data (in the form of trajectories) from the actual systems instead of building an intermediate representation, which, unless proven, may not be consistent with the actual system, (iii) our work is based on a generic semantic framework and can be instantiated for many different semantics for faults and conformance. We expect that the ideas developed in this paper can be applied to those non-causal theories of fault localisation; however, this may require a reformulation of the basic concepts.

3. Preliminaries

In this section, we first present the theory of actual causality in the context of discrete systems, originally proposed by Halpern and Pearl [39]. Then, we provide a brief overview of cyber-physical systems and their models, present our running example and the motivation behind this work.

3.1 Causal theory for discrete systems

Consider the following discrete example (based on the classical Billy and Suzy example [37]): suppose that two autonomous vehicles A and B are driving on a straight road, one behind the other, in an extremely foggy weather condition. At the end of the road, a pedestrian is situated; the heavy fog prevents the vehicles' cameras from detecting the pedestrian. Furthermore, a junction on the road is situated before the pedestrian, which allows the vehicles to turn right. Here, we assume that the pedestrian cannot escape the imminent collision: if one of the vehicles does not turn right at the junction, then the pedestrian will be hit. In the case that neither vehicle turns right, vehicle A will hit the pedestrian but vehicle B will not. If vehicle A turns right, and vehicle B does not, then vehicle B will hit the pedestrian. If both vehicles turn right, the collision is avoided altogether. This scenario is depicted in Figure 1.

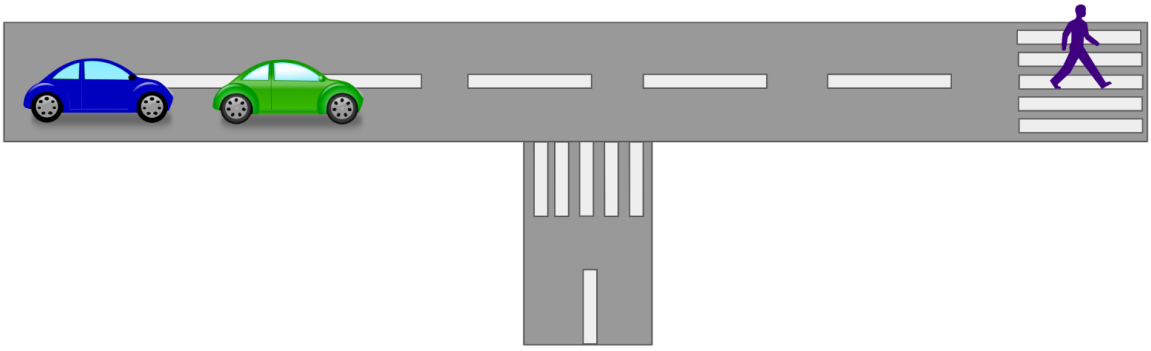


Figure 1: Illustration of the running example.

In this example, one can see that if neither of the two vehicles turn right, then only vehicle A can be the cause of the collision. Similarly, if vehicle A turns right, and B does not, then only vehicle B can be the cause.

Mathematical assessments of causality require formal modelling. As a precondition to a model, the signature provides the set of variables and their admissible valuations. Most of the formal definitions in this section are taken from those by Halpern and Pearl [38].

Definition 1 (Signature)**.** *A signature is a tuple*

$$\mathcal{S} = (\mathcal{U}, \mathcal{V}, \mathcal{R}),$$

where $\mathcal{U}$ is a finite set of exogenous variables, $\mathcal{V}$ is a finite set of endogenous variables, and $\mathcal{R}$ associates with every variable $Y \in \mathcal{U} \cup \mathcal{V}$ a finite and non-empty set $\mathcal{R}(Y)$ of possible values for Y.

Exogenous variables are determined by factors outside of the model while endogenous variables are affected by exogenous ones and also by other endogenous variables. For instance, going back to the autonomous vehicle example, vehicle A and B turns can be seen as *endogenous* variables, but the gravity and road friction that allow for the vehicles to steer can be seen as *exogenous* variables.

Definition 2 (Causal Model)**.** *A causal model over a signature $\mathcal{S}$ is a tuple*

$$M = (\mathcal{S}, \mathcal{F}),$$

where $\mathcal{F}$ associates with each variable $X \in \mathcal{V}$ a function denoted by F_X, such that:

$$F_X : (\times_{U \in \mathcal{U}} \mathcal{R}(U)) \times (\times_{Y \in \mathcal{V} \setminus \{X\}} \mathcal{R}(Y)) \to \mathcal{R}(X)$$

F_X describes how the value of the endogenous variable X is determined by the values of all other variables in $\mathcal{U} \cup \mathcal{V}$. The indexed cartesian products $\times_{U \in \mathcal{U}} \mathcal{R}(U)$ and $\times_{Y \in \mathcal{V} \setminus \{X\}} \mathcal{R}(Y)$ consider each possible values of the variables in $\mathcal{U}$ and $\mathcal{V} \setminus \{X\}$, respectively. Considering our discrete example, the causal model would have the following endogenous variables:

- AT for (vehicle) A Turns: 1 if it turns right, and 0 if it does not.
- BT for (vehicle) B Turns: 1 if it turns right, and 0 if it does not.
- AH for A Hits: 1 if it hits the pedestrian, and 0 if it does not.
- BH for B Hits: 1 if it hits the pedestrian, and 0 if it does not.
- PH for Pedestrian Hit: 1 if the pedestrian is hit, and 0 if it is not.

The set $\mathcal{U}$ of exogenous variables comprises all information we need to assume so as to render all relationships deterministic (such as the presence of oxygen, gravity and the route the vehicles follow). We denote by $\vec{u}$ (i.e., a set of valuations in $\mathcal{R}(\mathcal{U})$) as the context of a cause. That is, the context is a mapping of exogenous variables to their values, which are used to induce the value of the endogenous variables. In our example, $\vec{u}$ can be seen as the context that makes the vehicle moving possible.

Furthermore, the types of causal models to which Halpern and Pearl restrict their definitions are called *strongly recursive*. In essence, a causal model is strongly recursive, which for each endogenous variable, a context $\vec{u} \in \mathcal{R}(\mathcal{U})$ plays a role in defining its value. More

specifically, the context helps defining the value of a subset of the endogenous variables, which, in turn, will be used in conjunction with the functions in $\mathcal{F}$ to determine the value of the remaining endogenous variables.

In our discrete example, the context (which encompasses the route that each vehicle is following) comprises the variables $u_A \in \mathcal{U}$ and $u_B \in \mathcal{U}$. They represent the routes that vehicle A and B are following, respectively. They assume the value 1 if the respective vehicle is following a route that takes a right turn at the junction and 0 otherwise. In such a case, we can define the functions in $\mathcal{F}$ as follows.

- $F_{AT}(\vec{u}, BT, AH, BH, PH) = u_A$
- $F_{BT}(\vec{u}, AT, AH, BH, PH) = u_B$
- $F_{AH}(\vec{u}, AT, BT, BH, PH) = \begin{cases} 0, & AT = 1 \\ 1, & AT = 0 \end{cases}$
- $F_{BH}(\vec{u}, AT, BT, AH, PH) = \begin{cases} 0, & AT = 0 \\ 1, & AT = 1 \wedge BT = 0 \end{cases}$
- $F_{PH}(\vec{u}, AT, BT, AH, BH) = \begin{cases} 0, & AH = 0 \wedge BH = 0 \\ 1, & AH = 1 \vee BH = 1 \end{cases}$

In summary, the context considers the particular route that the cars are following, which dictates whether they will turn right or not (i.e., AT and BT). Then, these variables affect AH and BH, and those, in turn, affect PH, as defined in the functions ($\mathcal{F}$) above.

In Figure 2, we display the causal graph of this example, in which the nodes (representing variables) that have a direct impact on each other are connected by an edge.

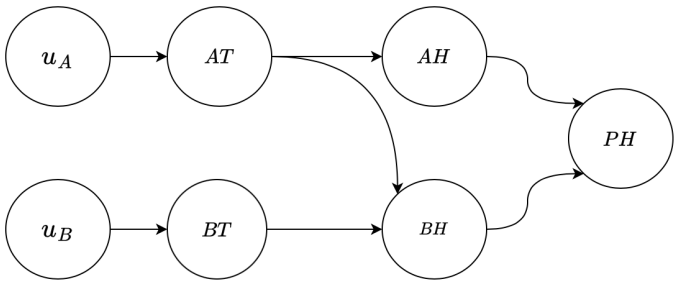


Figure 2: Causal graph of the AT/BT example.

The graphical representation of causal networks such as these are not used in the underlying theory nor in the implementation (see Section 5); however, they provide a visual aid to understand the examples.

Finally, to make the definition of cause precise, we first need a syntax for causal events. Given a signature $\mathcal{S} = (\mathcal{U}, \mathcal{V}, \mathcal{R})$, a formula of the form $X = x$, for $X \in \mathcal{V}$ and $x \in \mathcal{R}(X)$, is called a primitive event.

Definition 3 (Causal Formula)**.** *A causal formula is of the form*

$$[X_1 \leftarrow x_1, ..., X_k \leftarrow x_k]\Phi, where$$

- $X_1, ..., X_k$ *are distinct variables in* $\mathcal{V}$*.*
- $x_i \in \mathcal{R}(X_i)$*. And,*
- Φ *is a Boolean combination of primitive events.*

The formula $[X_1 \leftarrow x_1, ..., X_k \leftarrow x_k]\Phi$ states that Φ holds in a system where X_i is set to x_i for $i = 1, ..., k$. Such a formula can be abbreviated as $[\vec{X} \leftarrow \vec{x}]\Phi$.

Definition 4 (Intervention)**.** *Given a causal model* $M = (\mathcal{S}, \mathcal{F} = \{\mathcal{F}_{X_1}, \mathcal{F}_{X_2}..., \mathcal{F}_{X_k}\})$*, and a set of assignments* $[X_1 \leftarrow x_1, ..., X_k \leftarrow x_k]$*, an intervention on the causal model* M*, denoted by* $M_{\vec{X} \leftarrow \vec{x}} = (\mathcal{S}, \mathcal{F}_{\vec{X} \leftarrow \vec{x}})$*, is a modification to the structural equations in the causal model* M *such that* $\forall i \in \{1, ..., k\}, \mathcal{F}_{X_i} = x_i$*.*

An intervention of the type $X \leftarrow x$ can be interpreted as an update in $\mathcal{F}$ where the function for X is set just to x. We define a satisfaction relation between causal model and causal formulae next.

Definition 5 (Satisfaction Relation)**.** *Given a causal model* $M = (\mathcal{S} = (\mathcal{U} = \{U_1, ..., U_m\}, \mathcal{V}, \mathcal{R}), \mathcal{F})$*, a context* $\vec{u} = \{u_1, ..., u_m\}$*, and a primitive event* $(Y = y)$*, the satisfaction relation between the causal model, the context and the event, denoted by* $(M, \vec{u}) \models (Y = y)$*, holds if, and only if,* $(\mathcal{F}_Y \in \mathcal{F} \wedge U_1 = u_1, ..., U_m = u_m) \implies \mathcal{F}_Y = y$*.*

Furthermore, given a causal formula $[\vec{X} \leftarrow \vec{x}](Y = y)$*, the satisfaction relation between the causal model, the context and the causal formula, denoted by* $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}](Y = y)$*, holds if, and only if, in the causal model resulting by the intervention* $M_{\vec{X} \leftarrow \vec{x}} = (\mathcal{S}, \mathcal{F}_{\vec{X} \leftarrow \vec{x}})$*, we have that* $(\mathcal{F}_Y \in \mathcal{F}_{\vec{X} \leftarrow \vec{x}} \wedge U_1 = u_1, U_2 = u_2, ..., U_i = u_i) \implies \mathcal{F}_Y = y$*.*

Thus, given a context $\vec{u} \in \mathcal{R}(\mathcal{U})$, we write $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}](Y = y)$ if the variable $Y \in \mathcal{V}$ has the value y in a causal model M where X_i is set to x_i for $i = 1, ..., k$. The notation can also be used in the presence of a Boolean combination of primitive events: $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}]\Phi$. Note that $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}]\Phi \iff (M_{\vec{X} \leftarrow \vec{x}}, \vec{u}) \models \Phi$.

Furthermore, in the special case where no assignments are performed (i.e., $k = 0$), we write $(M, \vec{u}) \models (Y = y)$, if and only if the variable $Y \in \mathcal{V}$ has the value y given the context $\vec{u} \in \mathcal{R}(\mathcal{U})$ and the causal model M. This notation can also be used in the presence of a Boolean combination of primitive events: $(M, \vec{u}) \models \Phi$.

The types of events that are allowed as causes are of the form $(X_1 = x_1 \wedge ... \wedge X_k = x_k)$, that is, a conjunction of primitive events that can be abbreviated as $\vec{X} = \vec{x}$. Then, cause is formally defined as follows.

Definition 6 (Cause)**.** *We say that* $\vec{X} = \vec{x}$ *is a cause of* Φ *in* $(M, \vec{u})$ *if the following conditions hold:*

- *AC1.* $(M, \vec{u}) \models (\vec{X} = \vec{x}) \wedge \Phi$

- *AC2. There is a set $\vec{W}$ of variables in $\mathcal{V}$ and a setting $\vec{x}'$ of the variables in $\vec{X}$ such that if $(M, \vec{u}) \models (\vec{W} = \vec{w})$, then:*

$$(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}', \vec{W} \leftarrow \vec{w}]\neg\Phi.$$

- *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 $\vec{X}'$.*

In this definition, the variables in the set $\vec{W}$ allow for the cause to be tested under certain circumstances where the variables in $\vec{W}$ (which can be empty) are kept to their original values $\vec{w}$ even if they were supposed to be modified by the intervention $\vec{X} \leftarrow \vec{x}'$.

Consider the scenario in our autonomous vehicle example where $AT = 0, BT = 0, AH = 1, BH = 0$, $PH = 1, u_A = 0$, and $u_B = 0$. In our signature, we identify AT, BT, AH, BH and PH as endogenous variables ($\mathcal{V}$) and u_A and u_B as exogenous variables ($\mathcal{U}$); we would like to assess whether $AT = 0$ is the cause of the pedestrian being hit ($PH = 1$).

AC1 states that $(\vec{X} = \vec{x})$ cannot be a cause of Φ, unless both the primitive causal events $(\vec{X} = \vec{x})$ and the effect Φ are true in the causal model M, given the context $\vec{u}$. That is, it states that for $AT = 0$ to be the cause of $PH = 1$, then both need to be true in $(M, \vec{u})$; thus, in this scenario, AC1 holds. Conversely, if we were trying to assess whether $BT = 1$ is the cause of $PH = 1$ then, AC1 could not be satisfied, as $BT = 1$ is not true in $(M, \vec{u})$ and, therefore, it could not be considered a cause.

AC2 states that for $(\vec{X} = \vec{x})$ to be a cause of Φ, there must exist alternative values for the set $\vec{X}$ that leads to Φ not holding. This is under the assumption that there might exist a set of variables $\vec{W}$ that must be kept in its original value. To understand this concept, let us start with $\vec{W} = \{\}$ and assume we apply the intervention $AT \leftarrow 1$. This leads to vehicle A turning which results in vehicle B hitting the pedestrian ($BH = 1$) and, thus, the pedestrian is still hit ($PH = 1$); clearly, AC2 does not hold in this case. However, if we set $\vec{W} = \{BH\}$ and, hence, we maintain the value of variable BH even under the intervention $(AT \leftarrow 1)$ we have that neither car hits the vehicle $(AH, BH = 0)$ and the pedestrian is not hit $(PH = 0)$. Consequentially, we have that $(M, \vec{u}) \models [AT \leftarrow 1, BH \leftarrow 0]\neg(PH = 1)$ holds. This example shows the need for the set W.

Finally, AC3 asserts that the identified cause is minimal. In our scenario, it prevents $(AT = 0 \wedge BT = 0)$ from being a cause, since $(AT = 0)$ suffices to satisfy AC2. Thus, AC3 also holds and we can say that, in $(M, \vec{u})$, $(AT = 0)$ is a cause of $(PH = 1)$.

3.2 Cyber-physical systems

Cyber-Physical Systems (CPSs) integrate computational systems into their physical environments; examples of modern CPSs include vehicles and robotic systems [51]. A typical CPS is a system where sensors feed input signals to a digital controller (discrete component) attached to physical actuators (continuous component) in a feedback loop.

In order to model the continuous and discrete dynamics in CPSs, many formalisms have been used [5]. In this work, we make use of hybrid automata [6] to model the design of a CPS; it is a well-established formalism, with an intuitive semantics, besides being equipped with tools supporting different analyses [7, 19, 30, 31].

We first consider a running example. Then we introduce a motivation for our causal analysis of cyber-physical systems. Despite our choice of hybrid automata and Simulink in

our mechanisation, our approach imposes no constraints on the formalism that our theory can be applied to.

3.2.1 Running example: an autonomous vehicle

An autonomous electric vehicle is driving at constant speed of 10 m/s on a straight road, towards a stationary pedestrian situated on a crossroad. The braking distance (d) is a function that depends on the vehicle speed (*speed*), gravity (g) and the braking coefficient (*brakes*), which indicates the quality of the braking system (such as braking pads, tire quality, and tire pressure). Reasonable values for the latter are between 0.2 and 0.8.

$$d = \frac{speed^2}{2 * brakes * g}$$

Furthermore, the car is equipped with a lidar (a laser imaging, detection, and ranging system) that has 2 modes. A default long range mode detects objects within a 20 meters radius and a shorter range mode that halves the range. Whenever the battery enters a critical state, i.e., less than 5% of the total charge, the lidar switches to short range mode to reduce power consumption.

Consider a scenario that, when t = 0, the car is 80 meters away from the crossroad where the pedestrian is stationed. The battery is at 10% capacity and its consumption rate is constant at 1% of the total capacity per second (i.e., it will decrease to 9% after the 1 second, then to 8% after 2 seconds, and so on). Table 2 describes the initial valuation for the system variables.

Table 2: Running example variables.

Description	Type	Name	(Initial) Value
Lidar range	Variable	*lidarRange*	20
Critical battery threshold	Constant	*critical*	5 %
Braking coefficient	Constant	*brakes*	0.2
Car acceleration	Variable	*acceleration*	0 m/s^2
Car speed	Variable	*speed*	10 m/s
Car position	Variable	*carPosition*	0
Pedestrian position	Constant	*pedestrianPosition*	80
Battery decay rate	Constant	*decay*	1% / s
Battery charge left	Variable	*battery*	10%
Gravity	Constant	g	9.8 m/s^2
Critical check	Variable	*belowCritical*	False

If one simulates the system, when $t \approx 8.5s$, the car collides with the pedestrian. We would like to know to which parts of the system this design flaw can be attributed; this challenge is akin to a verification problem. Regardless of whether the undesired behaviour is due to an actual system failure or an oversight in its specification, we aim to find the causes for it and correct the behaviour.

We would like to note that the need for the distinction between constants and variables will be clear later on when interventions for cps are defined. In summary, a constant is a

special variable that does not change value over time and, in this case, interventions can only change their value throughout the entire system execution.

4. Causality for CPSs

In this section, we first present the motivation behind this work and the formal definitions for the underlying theory that supports our strategy. The definitions presented in this section are based on the theory of actual causality discussed in Section 3.1 but they are lifted to continuous systems.

4.1 Analysis of cyber-physical systems

Search-based test case generation techniques, such as the ones applied in our tool, HyConf [8], can find faults in a system by searching for inputs that exercise extreme conditions on the System Under Test (SUT). However, the downside of such a methodology is the difficulty of tracing back the fault to a particular event or part of the system.

It has been our experience that causes for faults in CPSs are very difficult to locate and, therefore, to fix. As a solution to this problem, causality is a concept with a potential to be exploited in the continuous setting. In the case of CPSs, our ultimate goal is to be able to determine which variables can be identified as causes of a fault and find the causes that exercise the safety levels of a system.

In CPSs, one can express the system behaviour in terms of trajectories. A trajectory is the valuation of a set of variables over time. We first provide some informal intuition of trajectory and other notions; in Section 4.2 we formalise all these notions. Considering our running example, Figure 3 shows a trajectory of duration $T = 12s$ with hypothetical valuation for two of the system variables: the car acceleration and the battery charge.

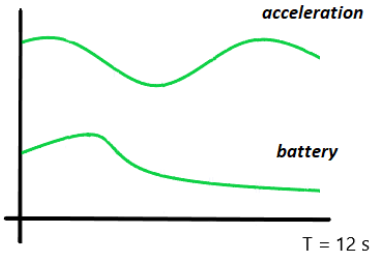
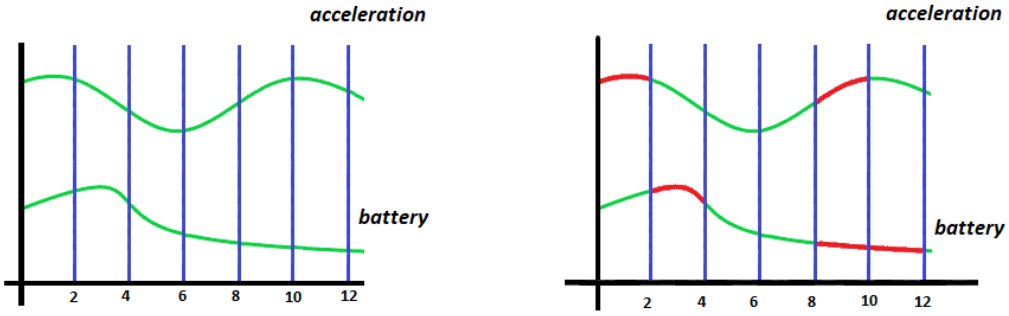


Figure 3: System trajectory.

Our strategy for causal analysis is to split a trajectory of duration T of a system into trajectory slices of equivalent size (see Figure 4a). We define trajectory slices as the projection of a trajectory considering a particular time interval. When building causal models, each trajectory slice is considered as a variable and, during the causal analysis, we use a set of slices to determine the cause of an events, such as the collision between the car and the pedestrian. For instance, in Figure 4b, the highlighted slices are the ones that could have been identified as the cause for the collision.



(a) Trajectory split into time intervals (slices). (b) Highlighted cause and effect.

Figure 4: Trajectory and trajectory projection.

When dealing with causal analysis for continuous systems, several aspects must be considered for a sound, yet realistic, theory and implementation. One of such aspects is the fact that determining cause is a costly computation [37]. Two important factors associated with the costs are (a) the number of endogenous variables in the causal model and (b) the number of variables in the cause. Thus, one cannot simply model a typical cyber-physical system by setting all variables as endogenous, otherwise determining the cause would be prohibitively computationally-intensive. In this work, in order to address factor (a), the choice of endogenous variables should be influenced by the variables that the user suspects are involved in the cause rather than the classical concept discussed in Section 3. Note that this does require some domain knowledge; however, selecting too many endogenous variables greatly affects performance (see our benchmarks in Section 6.4). Moreover, in order to address factor (b), we consider an upper band to the number of variables in a cause and in the search; this is discussed in the mechanisation section (Section 5.1).

On this topic, Beckers and Halpern [13] provide some insights into causal model abstraction which can allow modellers to think at a high level while still being faithful to a more detailed model. They discuss the notion of $\tau-$abstraction which suggests a transformation τ from causal model M1 to M2, where M2 is the high-level model where inessential differences are ignored.

Another topic to consider is with respect to cyclic models. In most intervention-based causal theories, the causal models must not comprise loops, such as in DAGs [32] and acyclic SCMs and GSEMs [58]. This has a significant impact on what can be modelled in CPS due to prominent presence of recursions and feedback loops. We circumvent this limitation by incorporating temporal dynamics into our causal models. This way, even in the presence of loops between system variables A and B (i.e., A affects the value of B and vice-versa), we use slices to build the causal models in such a way that a slice (i.e., a variable in a causal model) can only be affected by prior slices.

In what follows, we define the supporting theory and present the mechanisation steps to achieve such goals.

4.2 Trajectories and overriding

We start by defining valuation. Valuations serve as the basis for trajectories, which, in turn, define the semantic domain for models of CPSs.

Definition 7 (Valuation)**.** *Given a set of variables $\vec{V} = \{X_1, \ldots, X_n\}$, we denote by $Val(\vec{V}) = \vec{V} \to D$ the set of all total functions from $\vec{V}$ to the common domain D.*

In the remainder of this paper, we take D to be the set of real numbers $\mathbb{R}$. In cyber-physical systems, variables often have continuous valuation over time. This can be represented using trajectories, which are collections of variable valuations within a time interval. A discrete Boolean variable, for instance, can be modelled as a special case where the value taken by the variable is only 0 or 1.

Definition 8 (Trajectory)**.** *Given a set of variables $\vec{V}$, the set of trajectories over $\vec{V}$, denoted by $Trajs(\vec{V}) = \{x_1, \ldots, x_m\}$, is the set of all partial mappings $T \nrightarrow Val(\vec{V})$, where T is the time domain.*

We take T to be a convex subset of non-negative real numbers $\mathbb{R}_+$. We consider partial mappings over T, since trajectory slices (formally defined in the sequel), which may not be defined over the whole T, are also members of $Trajs(\tilde{V})$. Below we define an auxiliary function that retrieves the set of variables in a trajectory, which is useful in later definitions.

Definition 9 (Variables of a Trajectory)**.** *We denote by $var(x)$, the set of variables $\vec{V}$ over which the trajectory x operates.*

A trajectory that considers only a subset of the system variables can be obtained by projecting it over these variables.

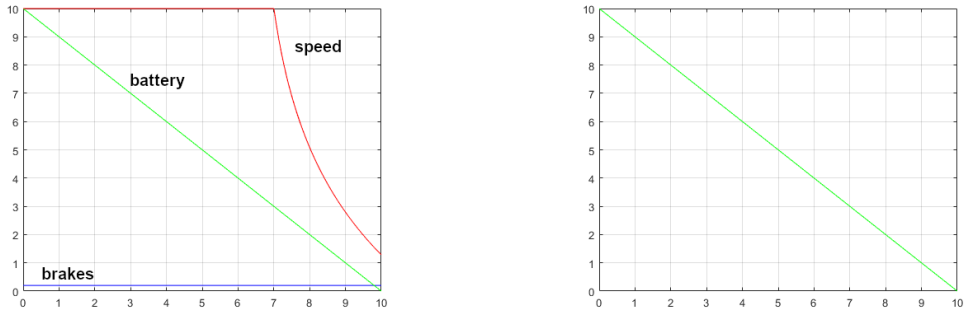
Definition 10 (Trajectory projection)**.** *Given a set of variables $\vec{V}$, the projection of a valuation $val \in Val(\vec{V})$ to $\vec{V}' \subset \vec{V}$, denoted by $val \downarrow_{\vec{V}'} \in Val(\vec{V}')$, is defined such that $\forall X \in \vec{V}'$, $(val \downarrow_{\vec{V}'})\ (X) = val(X)$. Furthermore, the projection of a trajectory $x : T \nrightarrow Val(\vec{V})$ to $\vec{V}' \subset \vec{V}$ is a trajectory $T \nrightarrow Val(\vec{V}')$, denoted by $x \downarrow_{\vec{V}'}$, such that $\forall t \in dom(x), (x \downarrow_{\vec{V}'})(t) = x(t) \downarrow_{\vec{V}'}$.*

Example 1 (trajectory and projection)**.** *Consider the running example discussed in Section 3.2.1. Figure 5a shows a trajectory $x \in Trajs(\{battery, brakes, speed\})$. Figure 5b shows the trajectory projection $x \downarrow_{\{battery\}}$.*

Note that, since the braking coefficient is constant, the valuation of *brakes* over time is a constant function. To discretise time in order to perform causal analysis and identify when the cause occurred, we first need to define the notion of time interval.

Definition 11 (Time interval)**.** *A left-closed right-open interval $[i, j)$, where $i < j$, is defined as a convex subset of $\mathbb{R}_+$, such that, $\forall x \in \mathbb{R}_+, x \in [i, j) \iff i \leq x < j$.*

In this work, a cause is defined using sub-trajectories of variables X over one or more time intervals, as shown in Figure 4. To formalise that, we define trajectory slices.



(a) A trajectory x.

(b) Trajectory projection $x \downarrow_{\{battery\}}$.

Figure 5: Trajectory and trajectory projection.

Definition 12 (Trajectory slice)**.** *Given a set of variables $\vec{V}$, a trajectory $x \in Trajs(\vec{V})$, and an interval $[i, j)$ such that $[i, j) \subseteq \mathrm{dom}(x)$, a trajectory slice $x_{[i,j)} \in Trajs(\vec{V})$ is defined as a function $x_{[i,j)} : [i, j) \to Val(\vec{V})$, such that $\forall t \in [i, j), x_{[i,j)}(t) = x(t)$.*

Example 2 (trajectory slice)**.** *Consider the trajectory x from Example 1, Figure 6 shows the trajectory slice $x_{[4,6)}$ projected over the variable battery, denoted as $x_{[4,6)} \downarrow_{\{battery\}}$.*

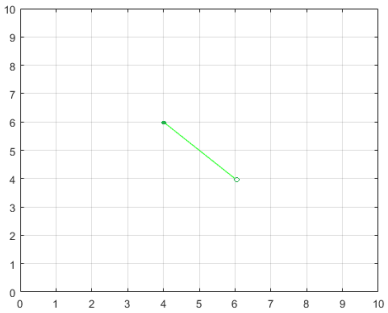


Figure 6: Trajectory slice projected over the variable *battery*.

4.3 Causal models for CPSs

As in the discrete case, a causal model is defined with respect to a signature.

Definition 13 (Signature for cyber-physical systems)**.** *Given a set of variables $\vec{V}$, a signature for a cyber-physical system C, is a tuple*

$$\mathcal{S}_C = \langle \mathcal{U}, \mathcal{V}, \mathcal{R} \rangle,$$

where:

- *$\mathcal{U} \subseteq \vec{V}$ is a finite set of exogenous variables;*
- *$\mathcal{V} \subseteq \vec{V}$ is a finite non-empty set of endogenous variables;*

- $\mathcal{R} \subseteq \mathit{Trajs}(\mathcal{U} \cup \mathcal{V})$.

In the discrete context, the set $\mathcal{R}$ contains the acceptable values for the variables in $\mathcal{U} \cup \mathcal{V}$. Analogously, in this extension, $\mathcal{R}$ is the set of selected trajectories that will be part of the causal analysis. In other words, we search for counterfactuals only within the set of trajectories $\mathcal{R}$.

Moreover, in our work, we assume that two properties always hold: the system must be deterministic and the causal model must be acyclic. We only consider deterministic systems since causal models are not achievable for non-deterministic systems; it would not be possible to define functions in $\mathcal{F}$ if variables could assume different values given the exact same parameter values. Furthermore, the dependency among endogenous variables in a causal model cannot be cyclic. That is, if the value of a variable X affects a variable Y, then the opposite must not be true. Otherwise, when applying interventions to an endogenous variable X, that would effect changes in Y and this, in turn, would result in further changes to X, which would conflict with the specific intervention that is being applied. This way, a cause could not be determined (as formalised in the sequel in Definition 18). This would result in some restrictions in systems that are inherently cyclic (e.g., with feedback loops) as only some specific combinations of endogenous variables are allowed. To mitigate this issue, we build causal models by splitting the system variables (e.g., *speed*) into slices and each slice corresponds to a causal variable (e.g., $speed_1$, $speed_2$,...,$speed_n$). Then, causes and effect are determined with respect to these slice variables. Figures 7 and 8 depict what would be causal model for the running example with and without using slices. Note how the cyyclic dependency between the variables *acceleration*, *speed*, and *carPosition* is removed; with this causal mode, variables in the nth slice only affect the variables in the $n\text{th} + 1$ slice. This results in the causal model being an abstraction of the real system, but for the purposes of verification, this suffices to determine actual causality given a sound implementation.

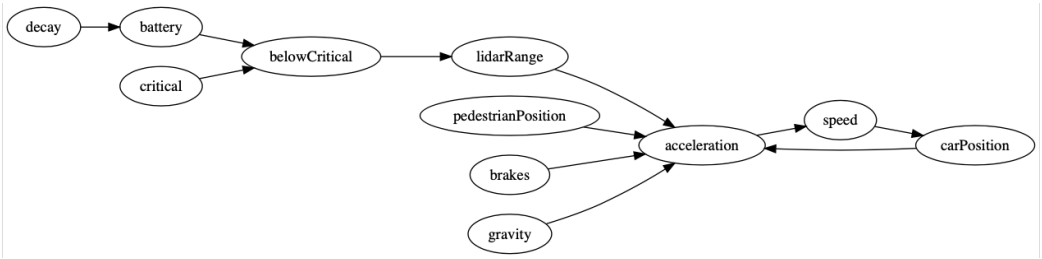


Figure 7: Cyclic causal graph of the running example.

Moreover, most cyber-physical systems comprise a large number of variables and, thus, defining all these variables as endogenous would render the analysis very costly in practice. Hence, in this work, the distinction between endogenous and exogenous variables depends on the feasibility and willingness to modify their values and assess causality. Hence, the best way to utilise this strategy require some knowledge about the system. Examples will be given throughout the paper to demonstrate this.

Example 3 (Endogenous and exogenous variables)**.** *Consider the autonomous vehicle example given in Section 3.2.1 and the hazard faced by the pedestrian and that the system is*

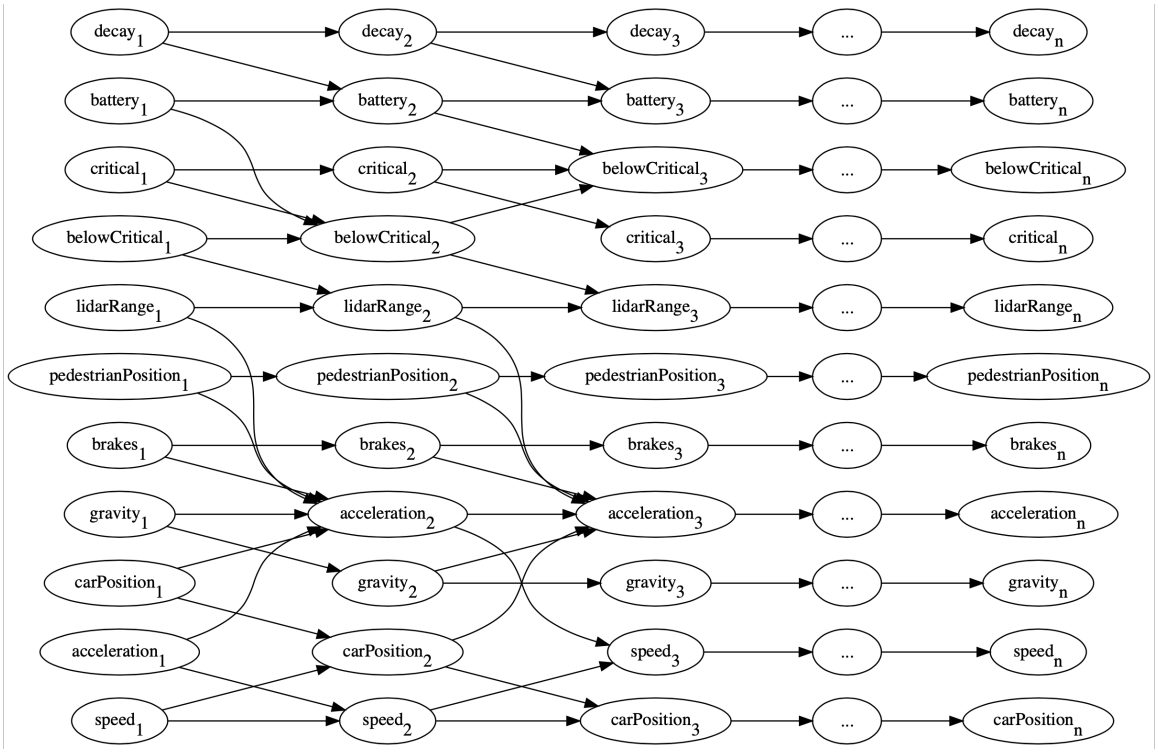


Figure 8: Acyclic causal graph of the running example.

split into two slices. A possible choice of sets for endogenous and exogenous variables can be the following:

- $\mathcal{U} = \{g_1,\ g_2,\ critical_1,\ critical_2,\ lidarRange_1,\ lidarRange_2,\ carPosition_1,\ carPosition_2,\ pedestrianPosition_1,\ pedestrianPosition_2,\ decay_1,\ decay_2,\ acceleration_1,\ acceleration_2,\ belowCritical_1,\ belowCritical_2\}$
- $\mathcal{V} = \{battery_1,\ battery_2,\ speed_1,\ speed_2,\ brakes_1,\ brakes_2\}$

The causal model takes a signature and a set of functions; each function assigns a trajectory for an endogenous variable given a trajectory for the remaining system variables.

Definition 14 (Causal model for cyber-physical systems)**.** *Given a signature $\mathcal{S}_C = \langle \mathcal{U}, \mathcal{V}, \mathcal{R} \rangle$ for a cyber-physical system C, a causal model is defined as*

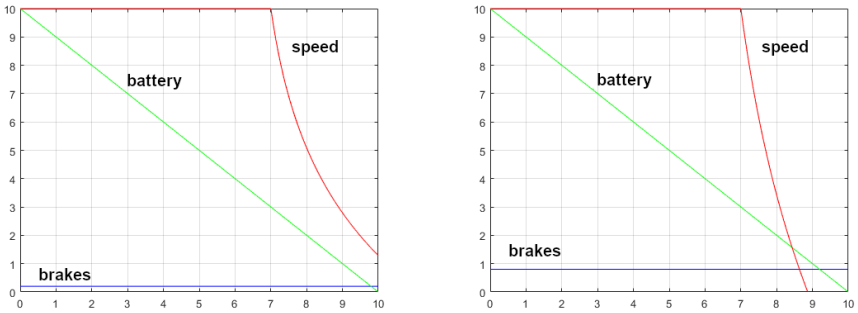
$$M = \langle \mathcal{S}_C, \mathcal{F} = \{\mathcal{F}_X \mid X \in \mathcal{V}\} \rangle,$$

where $\mathcal{F}_X : Trajs(\mathcal{U} \cup \mathcal{V} \setminus \{X\}) \nrightarrow Trajs(\{X\})$, such that $\mathrm{dom}(\mathcal{F}_X) = \{x \downarrow_{\mathcal{U}\cup\mathcal{V}\setminus\{X\}} \mid x \in \mathcal{R}\}$, and, for every trajectory $x \in \mathcal{R}$, $\mathcal{F}_X(x \downarrow_{\mathcal{U}\cup\mathcal{V}\setminus\{X\}}) = x \downarrow_{\{X\}}$.

Intuitively, a causal model associates each variable $X \in \mathcal{V}$ with a single trajectory $x \downarrow_{\{X\}} \in \mathcal{R}$, given a trajectory for each of the other variables in $\mathcal{V}$.

Example 4 (A causal model for the running example)**.** *Consider the example from Section 3.2.1, two trajectories x and y (Figure 9), and a signature $\mathcal{S}_C = \langle \mathcal{U} = \{g, critical, lidarRange, carPosition, pedestrianPosition, decay, acceleration, belowCritical\}, \mathcal{V} = \{battery, speed, brakes\}, \mathcal{R} = \{x, y\}\rangle$. Then, a causal model M can comprise the following functions:*

- $\mathcal{F}_{battery}(...) = 10 - (t * decay)$
- $\mathcal{F}_{brakes}(...) = 0.2$
- $\mathcal{F}_{speed}(...) = 10 + (t * acceleration)$



(a) Trajectory x. (b) Trajectory y.

Figure 9: Trajectories of the running example.

For simplification, we omit the variable names in the parameter field of the functions in the example above. In the next section, the formal syntax to represent causes and effects are given as Boolean expressions between trajectories. Therefore, we first define a grammar for a primitive event and then we describe its semantics informally.

Primitive Event Grammar

$$
\begin{aligned}
\langle Expression \rangle \quad ::= \; & \langle Trajectory \rangle \\
| \; & \langle Trajectory \rangle + \langle Expression \rangle \\
| \; & \langle Trajectory \rangle \text{ - } \langle Expression \rangle \\
| \; & \langle Trajectory \rangle * \langle Expression \rangle \\
| \; & \langle Trajectory \rangle \; / \; \langle Expression \rangle
\end{aligned}
$$

$$
\begin{aligned}
\langle BooleanOperator \rangle \quad ::= \; & =\langle Interval \rangle \\
| \; & <\langle Interval \rangle \\
| \; & >\langle Interval \rangle \\
| \; & \geq\langle Interval \rangle \\
| \; & \leq\langle Interval \rangle
\end{aligned}
$$

$\langle PrimitiveEvent \rangle \quad ::= \langle Trajectory \rangle \langle BooleanOperator \rangle \langle Expression \rangle$

$$\begin{aligned}\langle PrimitiveEventExpression \rangle ::= & \langle PrimitiveEvent \rangle \\ | & \neg \langle PrimitiveEvent \rangle \\ | & \langle PrimitiveEvent \rangle \wedge \langle PrimitiveEventExpression \rangle \\ | & \langle PrimitiveEvent \rangle \vee \langle PrimitiveEventExpression \rangle\end{aligned}$$

In our logic, a trajectory is assigned to a variable (e.g., $x : T \rightarrow Val(battery)$) but can also be represented by constant (e.g., $x : T \rightarrow 3.14$). An expression, however, is defined as arithmetic operations between trajectories. Given two trajectories x, y result between the addition $x + y$ is a new trajectory $s | \forall t \in dom(x) \cap dom(y) : s(t) = x(t) = y(t)$. The arithmetic operations between trajectories are defined over the addition, subtraction, multiplication and division operators.

Lastly, a primitive event is a Boolean comparison between a trajectory and an expression. Given two variables X and Y, two trajectories $x \in Trajs(\{X\})$ and $y \in Trajs(\{Y\})$, and an interval $[i, j) \subseteq dom(x) \cap dom(y)$, the primitive event of the type $x =_{[i,j)} y$ holds when for all $t \in [i, j)$, we have that $x(t) = y(t)$.

For instance, we use $x =_{[i,j)} y + z$ as syntactic sugar for $x =_{[i,j)} s \mid \forall t \in [i, j) : s(t) = y(t) + z(t)$, and, hence, we have that $\forall t \in [i, j) : x(t) = y(t) + z(t)$. Similarly, we also use real valued numbers in a primitive event (which can be seen as a new trajectory that is constant and set at that value). As an example, the primitive event $x <_{[i,j)} y * 3.14$ is syntactic sugar for $x <_{[i,j)} s \mid \forall t \in [i, j) : s(t) = y(t) * 3.14$.

In this extension, we denote Φ (i.e., the effect) as a Boolean combination of primitive events for CPSs. Given a causal model $M = \langle \langle \mathcal{U}, \mathcal{V}, \mathcal{R} \rangle, \mathcal{F} \rangle$, a trajectory $c : Trajs(\mathcal{U} \cup \mathcal{V})$ and a set of trajectory slices $\vec{x} = \{x \mid x \in Trajs(\mathcal{V})\}$, causes for CPSs are given in the format $(c \leftarrow\!\shortmid \vec{x})$. The special case where a conjunction of primitive events using the equality operator between one trajectory (c) and each trajectory slice in a set $(\vec{x} = \{x_1, x_2, \ldots, x_n\})$ is denoted by $(c \leftarrow\!\shortmid \vec{x})$. This format is an abbreviation for $\bigwedge_{i=1}^{n}(c \downarrow_{var(x_i)=_{dom(x_i)}} x_i)$. In practice, the trajectory c is obtained from a system execution and the trajectory slices $\vec{x}$ are taken from c. Hence, a cause $(c \leftarrow\!\shortmid \vec{x})$ means that the fact that the output of the system c comprises the specific trajectory slices $\vec{x}$ is a cause to an effect Φ.

In the discrete context, in order to identify causes, one needs to apply interventions and modify the value of variables. In the context of CPSs, causes are given using trajectory slices and thus the interventions need to be applied on the same slices as the prospective cause. Hence, we formally define the concept of alternative set of trajectories. Given a set of trajectories $(\vec{x})$, an alternative set of trajectories $(\vec{x}')$ is one that, for each trajectory in $\vec{x}$, there exists a corresponding trajectory in $\vec{x}'$ that ranges over the same variable(s) and the same time interval but differ in value. The opposite must also be true.

Definition 15 (Alternative set of Trajectories)**.** *Two sets of trajectories $\vec{x}$ and $\vec{x}'$ are called alternative if, and only if:*

- $\forall x \in \vec{x} : \exists! x' \in \vec{x}' : (var(x) = var(x') \wedge dom(x) = dom(x') \wedge \neg(x' =_{dom(x)} x))$
- $\forall x' \in \vec{x}' : \exists! x \in \vec{x} : (var(x) = var(x') \wedge dom(x) = dom(x') \wedge \neg(x' =_{dom(x)} x))$

We denote by $Alts(\vec{x})$ the set of all alternative set of trajectories of $\vec{x}$. Lastly, we lift the definition of the satisfaction relation (Definition 5) to work with causal models for cyber-physical systems. Similarly to the discrete version, we use (M, u) to represent a causal model and a context, which is now represented by a trajectory $u : Trajs(\mathcal{U})$ over the exogenous variables.

Definition 16 (Satisfaction relation for cyber-physical systems)**.** *Given a causal model for cyber-physical systems $M = \langle \mathcal{S}_{\mathcal{C}} = \langle \mathcal{U} = \{U_1, U_2, ..., U_m\}, \mathcal{V}, \mathcal{R} \rangle, \mathcal{F} \rangle$, a trajectory $u \in Trajs(\mathcal{U})$, and a primitive event $x =_{[i,j)} y$ where $x \in Trajs(\{X\})$ and $y \in Trajs(\{Y\})$, and $X, Y \in \mathcal{U} \cup \mathcal{V}$, the satisfaction relation between the causal model, the context, and the event, denoted by $(M, u) \models (x =_{[i,j)} y)$, holds if, for all $t \in [i, j)$, we have $(U_1 = u \downarrow_{U_1}, U_2 = u \downarrow_{U_2}, ..., U_m = u \downarrow_{U_m}) \implies (x(t) = y(t))$.*

Trivially, the satisfaction relation can be extended to a Boolean combination of primitive events, such as causes $(c \hookleftarrow \vec{x})$ or effects (Φ).

4.4 Causes for CPSs

We define below the notion of causal model update, which is akin to an intervention in the discrete case.

Definition 17 (Causal model update)**.** *Given a signature $S_C = \langle \mathcal{U}, \mathcal{V}, \mathcal{R} \rangle$, a set of variables $\vec{X} \subseteq \mathcal{V}$ and a set of trajectories $\vec{x} = \{x \mid x \in Trajs(\vec{X})\}$, an updated signature, denoted by S'_C, is defined as $\langle \mathcal{U}, \mathcal{V}, \mathcal{R}^{\vec{X} \leftarrow \vec{x}} \rangle$, where $\mathcal{R}^{\vec{X} \leftarrow \vec{x}}$ is the set of all trajectories $r \in \mathcal{R}$ such that $r \downarrow_{\{X\}} =_{dom(x)} x \downarrow_{\{X\}}$, for each $x \in \vec{x}$.*

Given a causal model $M_C = \langle S_C, \{\mathcal{F}_Y \mid Y \in \mathcal{V}\} \rangle$, an updated causal model, denoted by $M_{\vec{X} \leftarrow \vec{x}}$, is defined as $\langle S'_C, \{\mathcal{F}'_Y \mid Y \in \mathcal{V}\} \rangle$, where the updated function for $\mathcal{R}^{\vec{X} \leftarrow \vec{x}}$, denoted as $\mathcal{F}'_Y$, is defined on $\{x \downarrow_{U \cup V \setminus \{X\}} \mid x \in \mathcal{R}^{\vec{X} \leftarrow \vec{x}}\}$.

Intuitively, $M_{\vec{X} \leftarrow \vec{x}}$ is an updated causal model obtained by filtering the trajectories of the variables in $\vec{X}$ with the trajectories in $\vec{x}$. A sequence of two causal model updates is denoted by $M_{\vec{X} \leftarrow \vec{x}, \vec{Y} \leftarrow \vec{y}}$.

To determine cause, similarly to the discrete version, we use (M, u) to represent a causal model and a context, which is represented by a trajectory $u \in Trajs(\mathcal{U})$ over the exogenous variables. Now, consider a trajectory c that led to Φ where $c \downarrow_{\mathcal{U}} =_{\mathrm{dom}(u)} u$, then, a cause of Φ in (M, u), given in the format $(c \hookleftarrow \vec{x})$, is determined as follows.

Definition 18 (Cause for cyber-physical systems)**.** *Given a causal model $M = \langle \langle \mathcal{U}, \mathcal{V}, \mathcal{R} \rangle, \mathcal{F} \rangle$, a setting $u \in Trajs(\mathcal{U})$, a trajectory $c : Trajs(\mathcal{U} \cup \mathcal{V})$ such that $c \downarrow_{\mathcal{U}} =_{\mathrm{dom}(u)} u$, and a set of trajectory slices $\vec{x} = \{x \mid x \in Trajs(\mathcal{V})\}$, then $(c \hookleftarrow \vec{x})$ is a cause of Φ in (M, u) when the following three conditions hold:*

- *AC1. $(M, u) \models (c \hookleftarrow \vec{x}) \wedge \Phi$,*
- *AC2. There exists a set of variables $\vec{W} \subset \mathcal{V}$ and two sets of trajectories $\vec{x}' \in Alts(\vec{x})$, and $\vec{w} \subseteq Trajs(\vec{W}) \cap \mathcal{R}$ such that if $(M, u) \models (c \hookleftarrow \vec{w})$, then:*

$$(M_{\vec{X} \leftarrow \vec{x}', \vec{W} \leftarrow \vec{w}}, u) \models \neg\Phi$$

- *AC3. There is no strict subset of $\vec{x}$ that satisfies AC1 and AC2.*

This extension of the original definition for discrete systems (see Definition 6) fully considers continuous trajectories and time intervals, with cause being established using trajectory slices. We aimed for a conservative extension, however a few things are to be considered.

Firstly, in the original definition, Halpern uses $\vec{u}$ to describe the set of exogenous variables and valuations. In our definition, a trajectory contains multiple variables and their valuation over time, thus a trajectory $u \in \mathit{Trajs}(\mathcal{U})$ that contains all exogenous variables and their valuation over time is used.

Secondly, the minimality clause, AC3, now focus on trajectory slices rather than variables. In the original theory, AC3 states that no subset of the *variables and their specific values* in the prospective cause should satisfy AC2. In our extension, however, we employ trajectory slices, which are already defined based on both variables and values. Thus, given that a particular list of trajectory slices ($\vec{x}$) satisfies both AC1 and AC2, then no subset of $\vec{x}$ should satisfy AC1 and AC2 in order to satisfy AC3.

Thirdly, our theory does not restrict Φ to only endogenous variables. In the original work, strictly speaking, endogenous variables could not affect exogenous ones, thus they could not be included in Φ. In this work, however, this is not needed nor included in the definition as the choice of endogenous variables is a user choice in order to make the performance of the causality checks viable in practical settings (more on that in Section 6.4).

Lastly, we note how the set $\vec{w}$ is a subset of $\mathit{Trajs}(\vec{W}) \cap \mathcal{R}$. While this may seem counter-intuitive, the reason is that this enables a sound mechanisation of this theory. Suppose one is to determine whether $(c \leftarrowtail \vec{x})$ is a cause of a Φ. This requires one to evaluate whether, for all possible variables in $\vec{W}$, there exists any set of trajectory slices $\vec{w}$ that would result in a subset of $\vec{x}$ satisfying AC1 and AC2. This is intractable. Hence, instead, a mechanisation of this theory can restrict the set $\mathcal{R}$ to a countable set of trajectories (found, for instance, by a search heuristic).

We use Example 4 to explain Definition 18. Consider that the trajectory x from Figure 9a is c, that is, the scenario where a failure was observed. We define Φ as $\neg(c \downarrow_{\{speed\}} \leq_{[8.9,9)} 0)$, which holds if the variable *speed* is greater than 0 at any point in the interval $[8.9, 9)$. Essentially, we are checking the causes for the vehicle not stopping around the 9 seconds mark.

Now, we determine whether $c \leftarrowtail \{x \downarrow_{\{brakes\}}\}$ is a cause for Φ. Clearly, AC1 holds; both the cause and the effect are true in the model. Further, the cause is a singleton, which means it is minimal; thus AC3 holds. As for AC2, the set $\vec{W}$ can be empty. Consider that $\vec{X} = \{brakes\}$ and $\vec{W} = \{\}$ and consider the trajectory $\{y \downarrow_{\{brakes\}}\}$ (from Figure 9b) as our alternative set of trajectories. It is clear that AC2 is satisfied as $(M_{\vec{X} \leftarrow \vec{y}} \downarrow_{\{brakes\}}, u) \models \neg\Phi$ holds. We can see in Figure 9b that, when we increase the brakes, the car fully stops before the 9 seconds mark. We note that constants do not change over time and, thus, the value for the braking coefficient is set at the beginning of the scenario and is the same throughout. When an intervention is applied to such a variable, its value must change for the entire duration of its trajectory.

We further exemplify our theory of causality for CPSs (particularly, Definition 16) in Section 5.2. We provide our algorithm for causal assessment and use another concrete example for its explanation.

5. Practical applications of causality

In this section, we demonstrate how causal analysis can be applied in practice to CPSs. First, we detail a standalone process for causal analysis and then we exemplify, using the running example, its applicability in the verification process.

5.1 Causal analysis process

We explain the step-by-step process for causal analysis. The process incorporates the theory defined in Section 4 and has been mechanised and integrated into HyConf[1], a tool for test case generation and conformance testing of cyber-physical systems. HyConf employs multi-objective search-based heuristics to find challenging input trajectories that exercise extreme conditions of the system under test (SUT). Particularly, it checks for conformance violations based on the (τ, ϵ)-conformance notion [2]. In short, given two output trajectories (e.g., the system implementation and an idealised specification model), the tools compares the distance between the two trajectories in terms of time (τ) and space (ϵ). These parameters can be seen as margins of error, and if the two trajectories are beyond such values, the a non-conforming verdict is given (i.e., a failure was observed). The inputs and failures observed in future sections were obtained via HyConf. These are, in turn, fed as input to our causal analysis.

We consider the running example (see Section 3.2.1) and use it to illustrate our approach. Figure 10 depicts an overview of the process. The dotted lines represent manual steps or manually provided (input) artefacts whilst solid ones represent automatic steps or automatically provided artefacts, as explained in the bottom right of the figure. Our causal analysis process requires 3 inputs from the user: the system (implementation) or design (model) under verification (currently, we accept Matlab/Simulink models), the hazard or fault (Φ, written in a simplified format of Signal Temporal Logic formulae [48]) and the scenario (a trajectory c, in Matlab/Simulink output format) in which the hazard/fault occurs. What we call a scenario in this work is a finite execution of a system in the form of a trajectory that comprises all variables within the system. From this scenario, we derive the valuation of both the endogenous and the exogenous sets (including the context setting u) of variables.

The step-by-step is given below.

1. A model of a cyber-physical system, a scenario and the hazard must be given.

2. The user chooses the set of endogenous variables and the remaining variables are assumed exogenous automatically.

3. The causal model is built with a singleton $\mathcal{R}$ that (initially) only contains the trajectory c that led to the hazard.

1. https://github.com/hlsa/HyConf

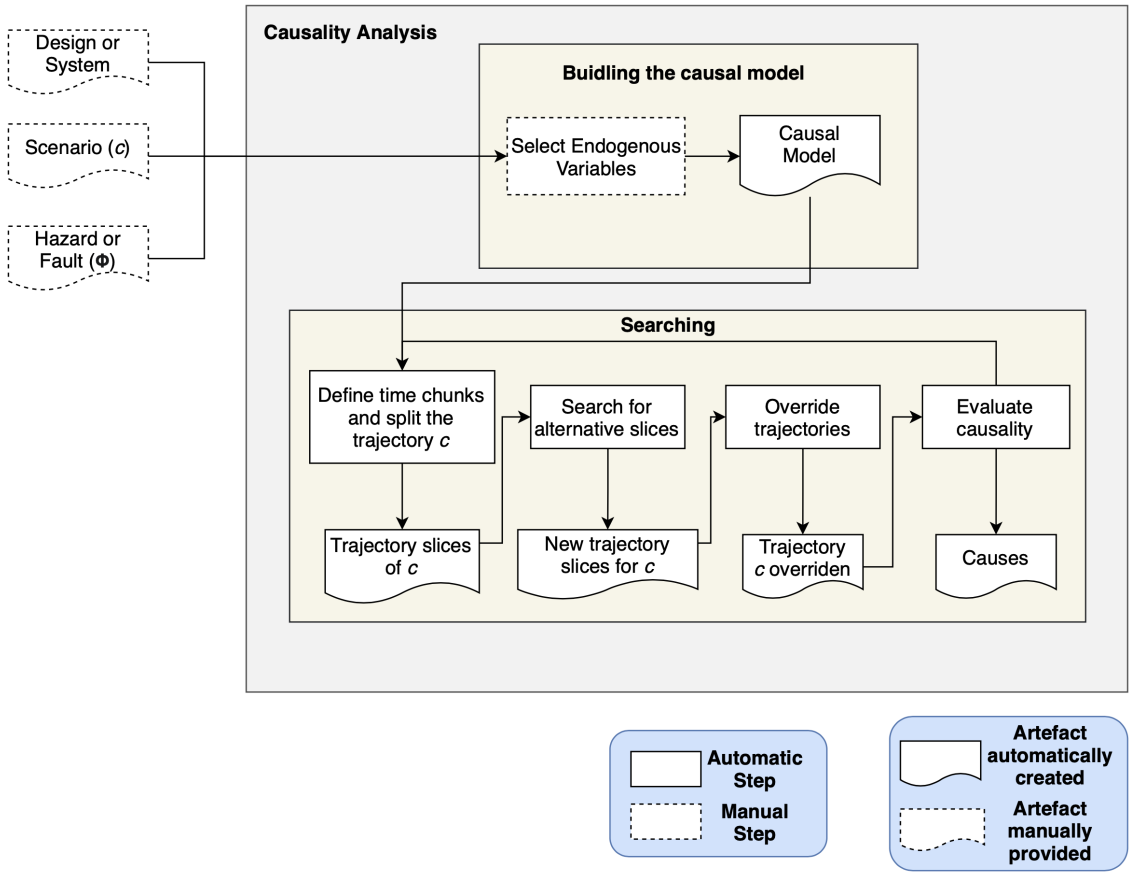


Figure 10: Causality process.

4. The search commences by automatically splitting the trajectory c (2 time intervals, initially) and therefore dividing it into smaller (and still continuous) trajectory slices.

5. The search finds new trajectory slices and then these are used to override c to look for violations of Φ. $\mathcal{R}$ is updated on-the-fly.

6. Causality assessment is performed. If a cause is found, the time intervals are reduced to increase precision.

7. If a maximum number of trajectory intervals has been explored, then the process stops. Otherwise, the search continues with a finer granularity of time intervals.

The first stage is to build a causal model. It is important to emphasise that, in a practical setting, our causal models are an approximation of the complex interactions of the physical system. Thus, there are some simplifications in the way that the causal models are built.

- In our theory, every variable that is influenced by the system should be classified as endogenous. However, in our practical causal models, having too many endogenous

variables would render the analysis very lengthy and costly. Thus, we leave the choice of endogenous variables to the user's discretion and to what they may want to analyse as cause.

- In our implementation, the set $\mathcal{R}$ is built iteratively. We start with a singleton $\mathcal{R}$ that only contains the trajectory that led to the fault (c); as the search evaluates new trajectories, by overriding c with trajectory slices in order to find a cause, they are added to $\mathcal{R}$. Once the search is concluded, multiple causes may have been found and, thus, we assess them considering the final $\mathcal{R}$ that has been built; this is the moment where we apply our theory for causality assessment of CPSs (see Section 4).

- When users choose the endogenous variables, they must set a maximum and minimum value and, if appropriate, mark the variable as constant. This is to avoid finding causes that do not respect system requirements and dynamics.

We re-iterate that the equations and variables of the causal model are provided as inputs from the user and the set $\mathcal{R}$ (state space) is built during the search. Moreover, causes are determined with respect to a particular causal model. Setting the maximum and minimum values for variables binds the set $\mathcal{R}$ of the causal model, and therefore the results are sound with respect to that particular model. Due to manual inputs (such as the boundaries for system variables), it is possible for the causal model to not reflect the actual system (from which c is taken) and, in this case, the cause might not reflect the reality of it. However, as far as cause/causal model are concerned, the results are still sound. In this work, we assume that the causal model that is built (given the manual inputs) respects the reality of the system, that is, one has to make sure that the boundaries in the causal model respect the boundaries in the physical system.

Furthermore, unlike variables, constants cannot be overridden in time intervals, only in their whole duration. This can be seen as a restriction on the set $\mathcal{R}$, such that constants can only be defined by constant trajectories. The set of functions $\mathcal{F}$ that describe how the endogenous variables are affected by other system variables must be provided by the user. Given this information, the tool checks whether the causal model is acyclic, otherwise the causality assessment becomes infeasible, as explained at the end of Section 4.3.

A difficulty associated with this strategy is the number of combinations of trajectory slices that have to be analysed to find the causes. This is particularly influenced by the chosen granularity of time intervals. That is, the number of time intervals increases the time spent in the search but conversely can identify a cause with more precision.

Furthermore, since the causes of a failure are more likely to be closer to the moment in time when the failure happened, an attenuation factor is implemented, where the search assigns higher search priority to the slices closer to the fault. In what follows, we present two algorithms employed in our strategy and we explain them using an example. Algorithm 2 shows the pseudo-code that conducts the causal analysis and Algorithm 1 shows the psuedo-code for the search that employs the causal analysis algorithm.

5.2 Application to the running example

The search algorithm (shown in Algorithm 1) receives the causal model (M_C), the hazard (Φ) and the trajectory c that leads to Φ, as input; its output is a set of causes. In what

follows, we apply the algorithm to our running example (the autonomous vehicle) to find causes for the collision hazard. First, we define the inputs to the algorithm.

5.2.1 Defining the hazard

The given cyber-physical system C is modelled in Simulink based on the example presented in Section 3.2.1. We characterise the hazard as the collision between the car and the pedestrian. The pedestrian position (*pedestrianPosition*) is set to 80 whilst the initial position of the car (*carPosition*) is set to 0. The collision occurs if the car cannot brake in time and its position exceeds that of the pedestrian (i.e., $carPosition \geq pedestrianPosition$).

Table 3: Issue observed

Effect	Φ
The car collides with the pedestrian	$c\downarrow_{\{carPosition\}} \geq_{\mathrm{dom}(c)} c\downarrow_{\{pedestrianPosition\}}$

Table 3 describes the effect with respect to which the cause is being established, along with Φ. Currently, the tool only accepts expressions using logical and arithmetic operators, logical connectors ($\wedge$, $\vee$), real numbers, and the variable names.

If we execute the system as originally defined in Section 3.2.1, the battery enters the critical state and the lidar range is reduced. Then, the car detects the pedestrian but does not brake in time and a collision occurs (i.e., $carPosition \geq 80$). Figure 11 depicts the trajectory c associated with this scenario, showing projections of c considering the variables *carPosition*, *battery*, *brakes*, and *lidarRange*.

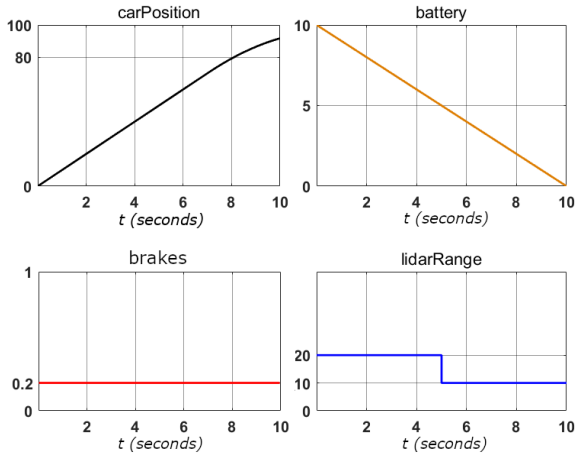


Figure 11: Projections of trajectory c that leads to Φ.

All 3 inputs for the process have now been defined: the design, the hazard and the scenario. Next, we generate the causal model.

5.2.2 Building the causal model

Our choice for elements that comprise the signature are shown below. Note that we depart from the choice made in Example 3, as to show that the set of endogenous variables can be chosen by the user. Recall that $\mathcal{U}$ is the set of exogenous variables and $\mathcal{V}$ is that of endogenous variables.

- $\mathcal{U} = \{$*acceleration, pedestrianPosition, carPosition, g, critical, speed, decay, belowCritical*$\}$
- $\mathcal{V} = \{$*brakes, battery, lidarRange*$\}$

Our choice of endogenous variables is based not only on what can reasonably be viewed as a cause but, more importantly, is based on what the user has control over and is willing to analyse and change in order to fix the system. The goal for this particular analysis is to determine which parts of the car are related to the failure and should be modified so that the car avoids colliding with the pedestrian.

Even though the pedestrian itself can be viewed as a part of the cause of the accident, we ultimately have no control over them, so we do not consider *pedestrianPosition* as an endogenous variable. Furthermore, we also classify the car acceleration and speed as exogenous variables; in our example, the car is autonomous and always travels at the road speed until it detects an obstacle, in which case, the car slows down. These variables are part of the set of exogenous variables, i.e., the external setting that exerts influence on the system.

Concerning the variables in the endogenous set, it includes the braking coefficient (*brakes*), the lidar range (*lidarRange*) and the battery level (*battery*). We aim to identify if they can be considered a cause for the collision with the pedestrian and, if applicable, the moment in time where the value of these variables can lead to the pedestrian being hit. As it happens, constants such as the braking coefficient can only be modified in its full trajectory. Variables, such as the battery, can change depending on which parts of the system is active, so it is possible to modify only a slice of it.

Initially, the set of trajectories that compose $\mathcal{R}$ is the trajectory (c) that led to the collision. As the search is conducted in the next steps, $\mathcal{R}$ will expand. The causal model is built using the sets of functions that relate the trajectories in $\mathcal{R}$ with the variables in $(\mathcal{U} \cup \mathcal{V})$. For example, considering the trajectory $\vec{u}$, we have:

- $\mathcal{F}_{brakes}(...) = 0.2$
- $\mathcal{F}_{battery}(...) = 10 - (t * decay)$
- $\mathcal{F}_{lidarRange}(...) = \begin{cases} 20, & battery \geq critical \\ 10, & battery < critical \end{cases}$

Since there are no cyclic dependencies in our set of functions, the analysis can proceed.

Algorithm 1: Pseudo-code for search algorithm.

```
input : CausalModel M;
input : Trajectory c, u;
input : BooleanPredicates Φ;
output: Set [(Trajectory, Trajectory, Set [Variable], TimeInterval)] causes;
Function Search(M,c,u,Φ) :
    Integer granularity = 2, maximumGranularity = 10;
    foreach Set [Variable] X⃗ in GetSubsetsOfSizeOne(M.signature.V) do
        foreach interval in GetIntervals(c, granularity) do
            (Set [Trajectory] x⃗′, Set [Trajectory] w⃗) = SearchHeuristic(c, X⃗,
             interval);
            Set [Variable] W⃗ = w⃗.GetVariables();
            if IsCause(M, u, Φ, x⃗, x⃗′, w⃗, X⃗, W⃗) then
                causes.Add((c, x⃗, interval));
            end
        end
    end
    if granularity < maximumGranularity then
        if causes.IsEmpty() then
            increaseGranularity(granularity);
            Search();
        end
        else
            foreach cause in causes do
                FocusSearch(cause.interval, X⃗);
            end
        end
    end
end
Function FocusSearch(interval, X⃗) :
    foreach interval in GetIntervals(c,maximumGranularity) do
        if interval.IsWithin(interval) then
            (Set [Trajectory] x⃗′, Set [Trajectory] w⃗) = SearchHeuristic(c, X⃗,
             interval);
            Set [Variable] W⃗ = w⃗.GetVariables();
            if IsCause(M, u, Φ, x⃗, x⃗′, w⃗, X⃗, W⃗) then
                causes.Add((c, x⃗));
            end
        end
    end
end
```

5.2.3 Searching for causes

We present, in this and in the next section (using the example), the algorithms both for the search (Algorithm 1) and for determining causality (Algorithm 2).

The function iterates through each endogenous variable (line 3) and intervals of trajectory c (line 4). In order to generate new trajectory slices, we apply a search heuristic (in this case, genetic algorithm [49]) to find data points that will form the alternative trajectory slices for variables in the cause and in the contingency set (line 5). For that, we use the Global Optimisation toolbox [1] for Matlab. The algorithm works on a population and attempts to find the global maxima or minima of a (fitness) function. A population is a set of points in the design space and it is initially generated randomly. The algorithm computes the next generation of the population by interacting with the fitness function and the individuals in the current generation until it finds the maxima. Our fitness function aims to generate slices that are diverse (in terms of distance and shape) from the slice that led to the fault.

Directly connecting the data points yielded by the search would result in an erratic curve. To mitigate this issue, in `SearchHeuristic`, we employ a notion of curve fitting using a fourth degree polynomial equation as a smoothing function [35]. This results in a smoother behaviour with the trade-off of the resulting curve being an approximation to the data points obtained by the search, instead of an exact match. As an alternative fit, one can use polynomial interpolation, which would result in a smooth curve that would exactly fit the points yielded by the search. However, this greatly increases computational costs since it requires functions with polynomial degree of (n-1), where n is the number of data points. The search typically yields thousands of data points, hence, for such large numbers, interpolation is impractical.

Once we have the alternative trajectories for variables in a prospective cause (sets $\vec{X}$ and $\vec{W}$), we call a function `isCause` that will check for actual causality (the details of this function can be seen in Algorithm 2). If positive, then causes are added to the outputting set (line 8). We represent a cause as a tuple composed by the original hazardous trajectory c, the variables in the cause, and the time interval.

After searching and determining causes, the algorithm attempts to increase granularity regardless of whether causes have been found or not. If a cause was found, we attempt to increase its precision (with respect to time) by searching the specific area where a cause has been found and, hence, trimming the time intervals into smaller pieces. Hence, if a cause is found with a granularity lower than the minimum, we focus the search on its specific time interval to increase precision (lines 27 to 40).

Otherwise, if no cause is found for coarser time intervals, we gradually increase granularity (from 2 to 4 and finally 10, that is, each slice should represent 50%, 25% and 10% of the trajectory (c) duration, respectively – this is performed by the function *increaseGranularity*) and resume the search (lines 13 to 16). The reason for increasing granularity even though no cause is found is due to the following two reasons:

- First, overriding only slices of a trajectory effectively turns it into a piecewise function. Since the degree of our polynomial function to smooth the curve is constant (regardless of how many data points there exist), the finer the time intervals the less data points

fall within each time interval, and, thus, the easier it is for the smooth curve to exactly match the points resulting from the search.

- Second, because our search is non-exhaustive in nature, the search can find and assess causality for more slices, if the time intervals are finer. This results in more causality assessments in overall and increases the likelihood to find causes.

There are two stopping criteria for the search. If the search does not find a cause even after splitting the trajectory into the maximum number of intervals (set to 10), then it stops. Otherwise, if it finds at least one cause, it focuses the search on those areas related to the causes in an attempt to increase precision (more subtle changes within shorter time intervals). After focusing (using the maximum number of time intervals), the search is stopped.

Once finished, another search is performed considering the interaction between variables. Algorithm 1 attempts to override one variable at a time. However, certain causes can only be found by overriding multiple variables at once. Thus, we also conduct the search considering pairwise and further considering sets of three variables. The pseudo-code for these additional searches are not shown here; in short, they lift the strategy presented in Algorithm 1 to consider multiple variables. Because the complexity of the search significantly increases, we do not attempt to search using sets with more than 3 variables.

5.2.4 Searching for a cause – part #1: lidar range

To check if the set of slices is a cause, the method `IsCause` asserts the causality clauses (as in Definition 18). The pseudo-code for this function can be seen in Algorithm 2. In this algorithm, we check the three clauses of Definition 18 (lines 3 to 5). Firstly, for AC1, the prospective cause ($\vec{x}$) is taken from c, which is built around causal model M in Algorithm 1 (and, thus, $(M, u) \models (c \leftarrow\!\!\!\shortmid \vec{x})$). Therefor, for AC1, we only need to check if the effect holds in the causal model ($(M, u) \models \Phi$). As for AC2, we check whether the slices found by the search suffice to violate Φ (lines 16 to 18). Finally, for AC3, we check whether any subset of the trajectory slices (line 22) satisfy AC1 and AC2 (lines 25 and 26). To do this, we perform a quicker version of the search (focusing on the subsets of $\vec{x}$), looking for alternative slices for $\vec{X}$ and $\vec{W}$ that satisfy both AC1 and AC2. We have previously noted that, in Definition 18, the set $\vec{w}$ is a subset of $\text{Trajs}(\vec{W}) \cap \mathcal{R}$ and hence, we do not need to check for all possible variations in $\text{Trajs}(\vec{W})$, but only the ones that are also in the set $\mathcal{R}$, which is built iteratively by the search. We discuss this algorithm in more details (including its soundness) in Appendix B.

Considering Definition 18, we have the previously defined signature (S_C), the causal model M, the trajectory c and $u = c \downarrow_{\mathcal{U}}$. The search is initiated and, in its first phase, it considers variables individually. A potential cause is identified when, after a trajectory slice is overridden, Φ does not hold.

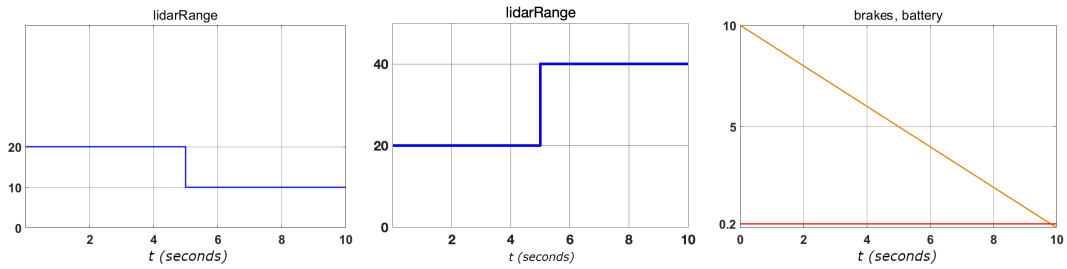
One of the causes presented is related to the range of the lidar. In the context of the theory, we have that $\vec{X} = \{lidarRange\}$. Figure 12a depicts the original *lidarRange* trajectory, i.e., $\vec{x} = \{x \downarrow_{lidarRange}\}$. Since the duration of the system simulation (i.e., the duration of trajectory c) is 10 seconds, the heuristic considers time slices of $5s$, $2.5s$, and $1s$, as explained in Section 5.1. Going back to Definition 18, our algorithm assesses whether

Algorithm 2: Pseudo-code for causal analysis algorithm.

input : CausalModel M;
input : Trajectory c, u;
input : Set [Variable] $\vec{X}$;
input : Set [Variable] $\vec{W}$;
input : Set [Trajectory] $\vec{x}$;
input : Set [Trajectory] $\vec{w}$;
input : Set [Trajectory] $\vec{x}'$;
input : BooleanPredicates Φ;
output: Boolean $isCause$;

```
Function IsCause(M, u, Φ, x⃗, x⃗′, w⃗, X⃗, W⃗) :
    isCause = False;
    if SatisfiesACOne(M, u, Φ) then
        if SatisfiesACTwo(M, u, Φ, x⃗′, w⃗, X⃗, W⃗) then
            if SatisfiesACThree(M, u, Φ, x⃗) then
                isCause = True;
            end
        end
    end
    return isCause;
end
Function SatisfiesACOne(M, u, Φ) :
    return Holds(M, u, Φ);
end
Function SatisfiesACTwo(M, u, Φ, x⃗′, w⃗, X⃗, W⃗) :
    M_C^updt = UpdateModel(M,{(X⃗, x⃗′), (W⃗, w⃗)});
    isAC2aTrue = Holds(M_C^updt, u, ¬Φ);
    return isAC2aTrue;
end
Function SatisfiesACThree(M, u, Φ, x⃗,) :
    foreach subOfX in GetSubsets(x⃗) do
        (x⃗′, w⃗′) = SearchHeuristic(c, subOfX, interval);
        X⃗′ = x⃗′.GetVariables();
        W⃗′ = w⃗′.GetVariables();
        if SatisfiesACOne(M, u, Φ) then
            if SatisfiesACTwo(M, u, Φ, x⃗′, w⃗′, X⃗′, W⃗′) then
                return False;
            end
        end
    end
    return True;
end
```

$(c \nleftarrow \vec{x})$ is a cause of Φ. For that, the 3 clauses associated with this definition must be satisfied.



(a) x. (b) x'. (c) w.

Figure 12: Trajectory slices related to the cause.

The trajectory c and the set of slices $\vec{x}$ lead to Φ, as shown in Figure 11, and, thus, $AC1$ is satisfied. Additionally, $\vec{x}$ is a singleton, which trivially satisfies $AC3$.

For $AC2$, we consider the set $\vec{W} = \{battery, brakes\}$, and the sets of trajectories $\vec{x'} = \{x' \downarrow_{lidarRange}\}$ (Figure 12b) and $\vec{w} = \{w \downarrow_{battery}, w \downarrow_{brakes}\}$ (Figure 12c). We can see that Φ does not hold for $(M_{\vec{X} \leftarrow \vec{x'}, \vec{W} \leftarrow w}, u)$ (see Figure 13). Thus, $AC2$ is satisfied.

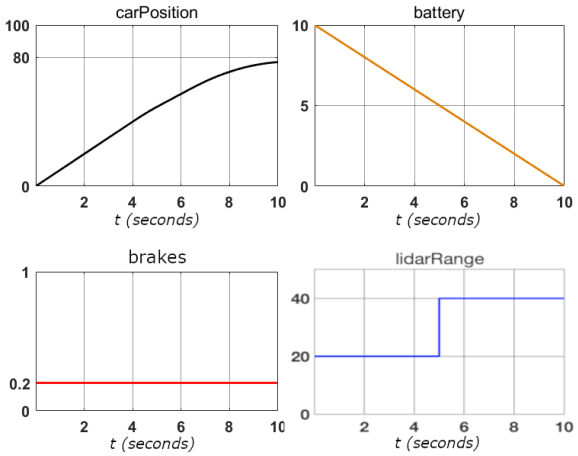


Figure 13: $\vec{X} \leftarrow \vec{x'}, \vec{W} \leftarrow \vec{w}$.

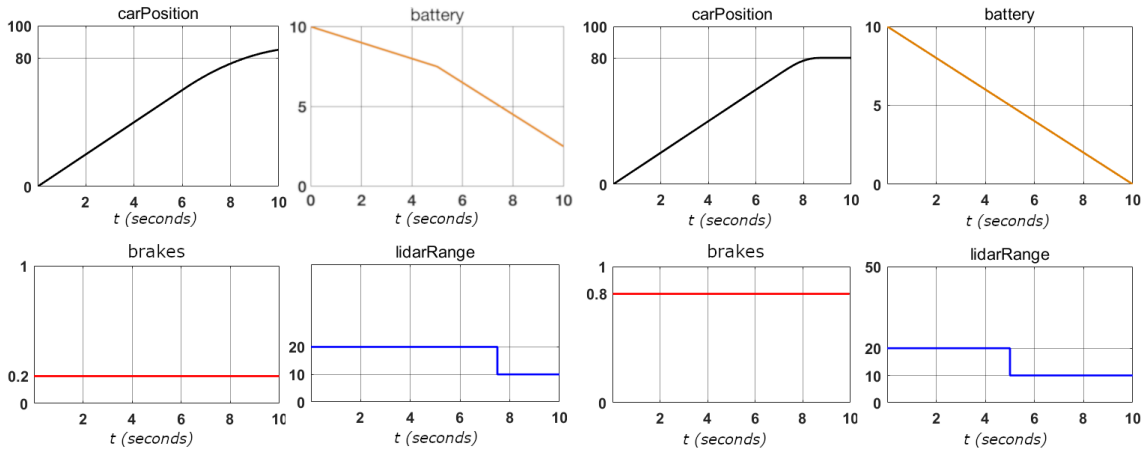
Increasing the range of the lidar alone would suffice to prevent the collision. The reason is that the car can detect the pedestrian at a greater distance. Although it is a valid cause, the search continues, now considering the remainder of the endogenous variables. Our search presents to the user every cause found. For instance, one could argue that it could be prohibitively expensive to install a much more powerful lidar in an actual car.

5.2.5 Searching for a cause – part #2: battery and brakes separately

No additional causes have been found whilst the search considered these variables individually. As an illustration, we present some of the trajectories that were considered by the search.

For instance, the battery was overridden during the time interval $[0, 5)$, depicted in Figure 14a. Due to an improved battery, the lidar works in long range mode, detects the pedestrian and starts braking sooner. However, with the default braking system the car still hits the pedestrian. Thus, the battery alone has not been identified as a cause.

Analogously, the search considered the braking system and overrode it to the highest value permissible by the design. This resulted in a much lower braking distance, as shown in Figure 14b. However, this also does not prevent the accident. The car brakes harder but too late since the lidar is in a lower range mode due to a battery that is below the critical threshold. The brakes alone have also not been identified as a cause.



(a) Overriding only the battery.

(b) Overriding only the brakes.

Figure 14: Overriding the trajectories (search – part #2).

So far, when analysing the endogenous variables individually, only the lidar range has been identified as a cause. The next step is to consider variables pairwise, that is, overriding two variables at the same time.

5.2.6 Searching for a cause – part #3: battery and brakes simultaneously

Since the lidar range has been identified as a cause by itself, the search ignores this variable when searching for causes related to multiple variables. The reason for that is the minimality clause, AC3.

A second cause was found when overriding trajectory slices for both the battery and the brakes. As shown in Figure 15, the car brakes harder and sooner, thus avoiding the accident. As a conclusion, the slices for both the battery and the brakes together can also be identified as a cause for the collision.

We emphasise that some limitations apply to the variables in the interventions. For example, its maximum and minimum values and whether the variable is a constant. This

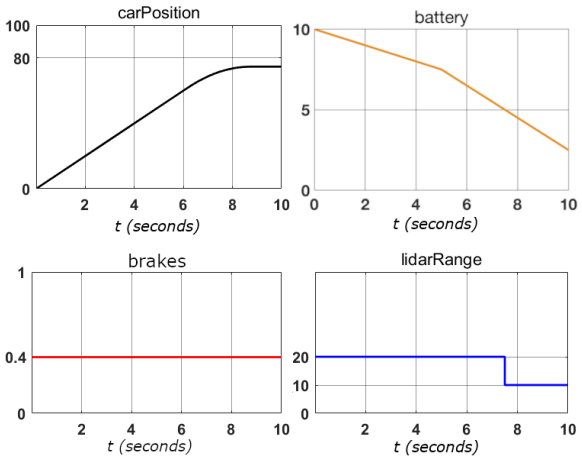


Figure 15: Overriding both battery and braking coefficient (search – part #3).

is to avoid finding causes that do not respect physics or system requirements. In this case study, reasonable values for the braking coefficient is between 0.2 and 0.8 and, thus, having a braking coefficient of, for instance, 300 would generate a cause that does not respect real world scenarios. These restrictions can be seen as applying restrictions to the possible trajectories in $\mathcal{R}$.

6. Empirical evaluation

In this section, we present two case studies and a series of benchmarks that explore the applicability of our technique for systems more complex than the running example.[2] More precisely, we describe the application of the strategy proposed in Section 5.1 to two case studies. In the first case study, we find causes for (systematically inserted) faults on a suspension system of a vehicle. In the second one, we explore an autonomous vehicle platoon to find causes behind communication issues. For the benchmarks, we explore 4 scalable systems.

6.1 Research objectives

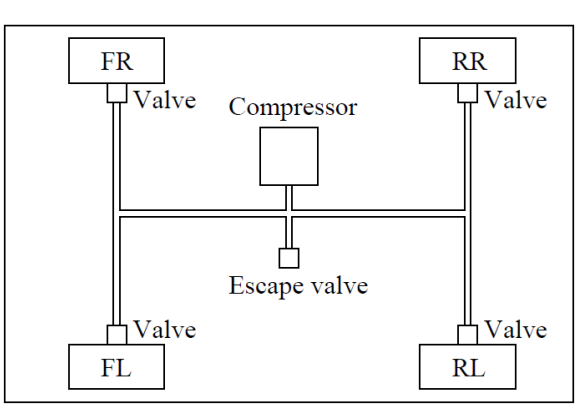
The main goal of the case studies is the evaluation of our causal analysis technique in the context of two verification problems. The first one is about whether it can be applied to identify the causes of (inserted) faults in the selected CPS. As for the second problem, we aim to assess whether it can also identify causes in hazardous situations due to design oversights. In particular, the case studies illustrate how the causality analysis can be used to identify injected faults and design oversights.

We consider case studies containing failures and hazards reported in previous works [8, 9], and apply our technique by defining the effects (Φ), building the causal models, and assessing causality.

2. A lab package for the experiments described in this section can be found on Zenodo: `http://tinyurl.com/CausalityForCPS-labpackage`

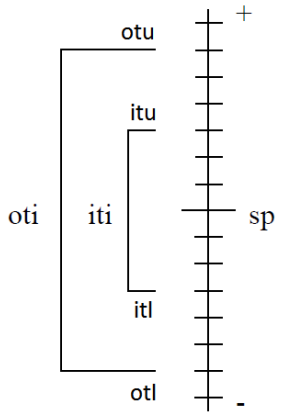
6.2 Case study #1: verifying a suspension system

In this first case study, we examine an automotive pneumatic suspension system [52]. We make use of results from a previous experiment [8], in which the faults were manually inserted in the system via a mutation process and detected using Hyconf. The purpose here is to confirm whether our analysis can detect their causes and assist with the correction of such faults. The system's goal is to increase driving comfort by adjusting the chassis level to compensate for road disturbances. This is achieved by a suspension system that connects the valves attached to each wheel to a compressor and an escape valve (see Figure 16a).



FR: front right wheel RR: rear right wheel
FL: front left wheel RL: rear left wheel

(a) Suspension system overview.



(b) Tolerance levels.

Figure 16: Suspension system.

The system aims to keep the chassis level as close as possible to a defined set point in each of the four wheels. The decision to increase or decrease the chassis level is based on the tolerance intervals defined for each wheel, as depicted in Figure 16b. We consider $[sp{-}otl, sp{+}otu]$ and $[sp{-}itl, sp{+}itu]$ as the outer and inner tolerance intervals, respectively. Here, *sp* represents the set point, which is the target value of the chassis level, and *itu, itl, otu* and *otl* represent the inner and outer tolerance thresholds along with their respective upper and lower values. Table 4 displays the variables in the system.

The system receives four inputs and it outputs the current chassis level h. The inputs are *dist*, *cp*, *ev*, and *bend*. The first one (*dist*) corresponds to the disturbance level coming from the environment, which indicates road perturbations such as small depressions or elevations. The *cp* and *ev* inputs dictate the change in the chassis level performed by the compressor and the escape valve, respectively. Finally, the variable *bend* indicates whether the vehicle is turning, which prohibits the adjustment of the pneumatic levels.

The control flow is described as follows. The system starts with the chassis set within the tolerance interval and all valves, as well as the compressor, are closed. Changes to the chassis level are constantly being monitored in order to determine the need to its increase or decrease. This is done by comparing the filtered chassis level (f) to tolerance limits

Table 4: Suspension system variables.

Name	Type	Description
dist	Input	Road disturbance
cp	Input	Compression valve
ev	Input	Escape valve
bend	Input	Car is on a bend
c	Output	Influence of valves on the chassis
h	Internal	Current chassis level
f	Internal	Filtered chassis level
e	Internal	Filtered disturbance
T	Internal	Filter constant
sp	Internal	Set point of the chassis
otu	Internal	Outer tolerance upper limit
otl	Internal	Outer tolerance lower limit
itu	Internal	Inner tolerance upper limit
itl	Internal	Inner tolerance lower limit

(*otu*, *otl*, *itu*, *itl*). The filtered chassis value is obtained by setting is first derivative to an equation that depends on the current chassis level, the previous filtered value and the filter constant ($\dot{f} = \frac{h-f}{T}$). On the other hand, the current chassis level h depends on the influence of the compressor and escape vales and on the filtered road disturbances ($\dot{h} = \dot{c} + \dot{e}$).

In what follows, we manually insert faults to the actual implementation and apply our process to determine their causes.

6.2.1 Defining the faults

We have selected two mutants from our previous experiment [8] and an overview of them can be seen in Table 5. A manual inspection guarantees that the mutants are non-equivalent. The first mutant changes the value of a system constant. The second replaces the value of a variable when the system should be lowering the chassis.

Table 5: Faults inserted and the issue observed.

Original	Mutation	Effect	Φ
$otu = 5$	$otu = 6$	Pressure increases more than allowed around 10s.	$c\downarrow_{\{f\}} >_{[8,12)} c\downarrow_{\{sp\}} +5$
$\dot{e} = dist$	$\dot{e} = 0$	Sometimes, disturbances do not affect the system.	$c\downarrow_{\{dist\}} >_{\mathrm{dom}(c)} 0 \wedge c\downarrow_{\{c\}} =_{\mathrm{dom}(c)} 0$

6.2.2 Building the causal model

Here, we define the signature by selecting the endogenous and exogenous variables, in addition to building the causal model to assess the failures. With respect to the first fault, the noticed effect is a high pressure in the tires. As for the second fault, the issue is that road disturbances are not affecting the system sometimes. To properly fix these faults, we need to identify why and when they happen.

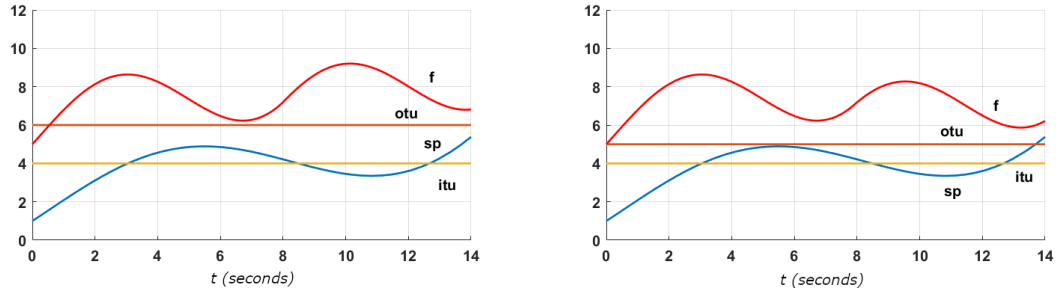
Considering we are trying to find causes for system failures, we do not want to assign inputs as causes for such faults. Inputs are external factors and cannot be controlled by the system. Thus, we select as endogenous variables everything but the inputs *dist*, *cp*, *ev* and *bend*.

- $\mathcal{V} = \{e,\ c,\ h,\ f,\ T,\ otu,\ itu,\ itl,\ otl,\ sp\}$
- $\mathcal{U} = \{dist,\ cp,\ ev,\ bend\}$

With respect to $\mathcal{R}$, we start with the trajectory that led to the issue in each case and more trajectories are added to each $\mathcal{R}$ as the search is conducted. The initial trajectories were found in a previous experiment during conformance verification [8]. The structural equations are as follows.

- $\mathcal{F}_e(...) = t * dist$
- $\mathcal{F}_c(...) = \begin{cases} 6.8 + t * ev, & f > sp + otu \\ 6.8 + t * cp, & f < sp + otl \\ 6.8 + t * 0, & otherwise \end{cases}$
- $\mathcal{F}_h(...) = t * (\dot{e} + \dot{c})$
- $\mathcal{F}_f(...) = t * ((h - f)/T)$
- $\mathcal{F}_T(...) = 10.3$
- $\mathcal{F}_{otu}(...) = 5$
- $\mathcal{F}_{itu}(...) = 4$
- $\mathcal{F}_{itl}(...) = 2$
- $\mathcal{F}_{otl}(...) = 1$
- $\mathcal{F}_{sp}(...) = dist + ((itl + itu)/2) * 1/T$

As defined in Section 4, for certain slices to be classified as cause, 3 clauses need to be satisfied. In our implementation, AC1 and AC3 are automatically satisfied as they always hold by construction. First, the search only tries to override trajectory slices that led to the fault, considering the causal model, thus AC1. Second, the search starts looking at slices individually before expanding the search to consider multiple variables. Furthermore, we do not consider the variables that have already been included in causes when considering further combinations, thus AC3. In what follows, we will focus our analysis on clause AC2.



(a) Trajectory c that led to the first failure. (b) Overriding *otu*.

Figure 17: Causal analysis for the first mutant.

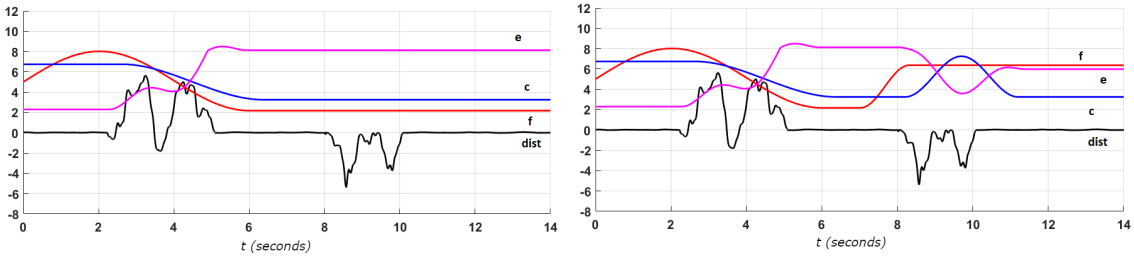
6.2.3 Determining cause – mutant #1

In this step, the search looks for trajectory slices candidates and confirms whether they are actual causes. The first failure is observed when the chassis level increases further than what is expected; the system should lower the pressure when $f > sp + 5$, which does not happen immediately in the implementation (see Figure 17a): When $t \approx 10\ s$, we have that $sp \approx 3.71$ and $f \approx 9.27$.

The cause (i.e., in c, the variable *otu* is equals to 6 during the time interval [0,14)) is trivially found by the search, when it assesses whether $(c \hookleftarrow \vec{x})$ when $\vec{x} = \{x_{otu}\}$. The failure ceases to occur when the constant *otu* is reduced (see Figure 17b), which is a clear indication that *otu* might have been set to a higher value than it was required.

6.2.4 Determining cause – mutant #2

With respect to the second fault, to find its cause it was necessary to override 2 slices of the original problematic trajectory. The trajectory displaying the issue is depicted in Figure 18a. The first disturbance affects the system and it responds accordingly: the chassis level is lowered and so is the value of f. The second disturbance, however, does not affect the system.



(a) Trajectory c that led to the second failure. (b) Overriding e and f.

Figure 18: Causal analysis for the second mutant.

The search found that no slices of e or f alone have been identified as cause. To find a situation where $\neg\Phi$, it was necessary to override two slices, as shown in Figure 18b. Firstly,

the filtered chassis level f was overridden and increased in the $[7, 8)$ time interval, and, secondly, the value of e had to be changed during the $[8, 10)$ time interval, which triggered a correction on the chassis level c. The AC2 clause was only satisfied when these two slices were overridden together.

This cause seems intuitive. Due to the fact that if the system is not affected by *dist* but is affect by e, we can assume that there is a problem in the disturbance equation. Moreover, the search increased f to the point that the system left the *down* location, which suggests that the problematic equation is associated with this particular location.

6.3 Case study #2: exploring the design of a connected platoon

Vehicular platooning is a cooperative and autonomous driving technology for linking two or more vehicles in a convoy. The goal of the convoy is to keep a close but safe distance between the vehicles using V2V (vehicle-to-vehicle) communications and automated driving technologies [15].

We have built a model of such a system [9] whose goal is that all vehicles in the platoon should keep a safe distance but within each other's communications range. In this model, the leading vehicle is driven by a human driver, while the velocity of the following vehicles is autonomously controlled; the autonomous followers should keep up with the leading vehicle's velocity.

The communication rules follow the standard defined in the ETSI EN 302 637-2 (Cooperative Awareness Basic Service - CAS) documentation [28], which, among others, describes the rules for the frequency of packet transmission. These packets comprise Cooperative Awareness Messages (CAM), which contain information about the vehicle, such as acceleration, direction and position. Rules for sending a packet are parameterised and take into consideration three factors: (i) whether a vehicles has moved a long enough distance, (ii) a certain amount of time has passed, and (iii) its speed has changed above a certain threshold. If any of the three cases holds, then the vehicle must send a packet to communicate with the others. Moreover, we use a simple controller called the Intelligent Driver Model (IDM) [62] in order to accelerate and decelerate the followers.

An overview of our model can be seen in Figure 19. In the communication architecture that we use, each vehicle receives information from the leader and from the vehicle in front of it. This allows for each follower to learn about the leader's manoeuvres soon after they happen and anticipate them before they are fully propagated through the platoon. Other design decisions regarding the communication architecture may lead to different analysis results; for instance, using direct communication with the remaining followers might increase safety but also channel congestion.

The current model only takes longitudinal movement of the vehicle into account. We assume that the platoon moves along a very long highway without any drastic changes in direction. The main input of our testing campaign is the behaviour of the human driver in the lead vehicle (i.e., its acceleration). Once this input is generated and provided to the lead vehicle, then, via V2V communication, the convoy of followers must autonomously adjust their behaviour to match the vehicle in front. The other inputs to our algorithms are the design-space parameters of the CAS protocol: by automatically searching through

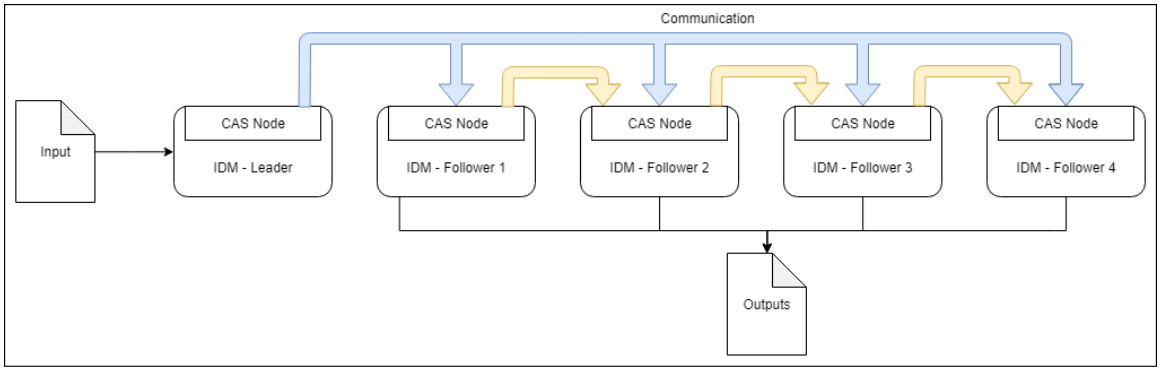


Figure 19: Platooning communication diagram [9].

the design space, we explore the effect of these parameters on the safety and the quality of the platoon behaviour.

6.3.1 Causal analysis

In a previous work [9], we used HyConf to analyse how our model fared with respect to the ETSI standard. For that, we conducted a controlled experiment to compare the default parameters against alternative values. In order to generate inputs for continuous systems, a search-based approach was used: we formulated a multi-objective search problem that maximises hazard likelihood, data age as well as coverage of the input space via diversity of test inputs. Our approach automatically generated scenarios that resulted in hazardous situations (i.e., collision) whilst abiding by the standard.

Here, we select the faulty scenarios obtained by Araujo et al. [9] and use our causal analysis to explore safety levels between network parameters and platoon speeds. The scenarios are a combination of variations in the parameters of the ETSI standard to trigger CAMs (see Table 6) as well as patterns of acceleration and deceleration from the leading vehicle generated by HyConf (see Figure 20).

Table 6: ETSI 302 637-2 CAM triggering parameters (adapted from [9]).

	T_{min}	T_{max}	d_{min}	v_{min}
Default	100 Hz	1000 Hz	4 m	0.5 m/s
Increased frequency	50 Hz	500 Hz	2 m	0.25 m/s
Decreased frequency	125 Hz	1250 Hz	5 m	0.625 m/s

Table 6 shows the default values for the parameters as well as two variations that increase or decrease the overall frequency rate. The parameters are for the minimum and maximum frequency (T_{min} and T_{max}, respectively), minimum covered distance (d_{min}) and minimum change in speed (v_{min}). Furthermore, Figure 20 depicts one of the acceleration scenarios generated as input that led to a faulty behaviour.

We conduct a causal analysis on two incidents. For each one of them, we define the corresponding Φ that describes the failure scenarios.

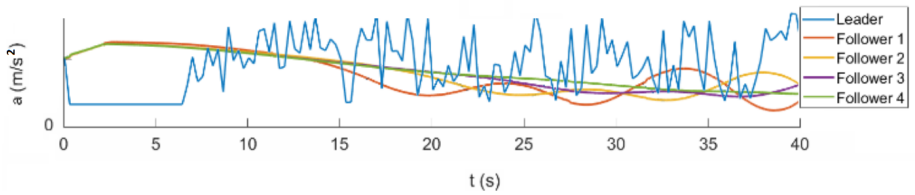


Figure 20: Example of a scenario generated by HyConf [9].

Table 7: Observed failures.

Variation	Effect	Φ
Increased frequency	Follower 4 goes out of range (60 m)	$c\downarrow_{\{f4.position\}} + 60 <_{\mathrm{dom}(c)} c\downarrow_{\{f3.position\}}$
Decreased frequency	Follower 2 collides with follower 1	$c\downarrow_{\{f2.position\}} \geq_{\mathrm{dom}(c)} c\downarrow_{\{f1.position\}}$

In this case, we would like to assess whether the vehicles kinematic rules can be interpreted as cause of the faulty behaviour instead of the network parameters. For simplicity, we classify the endogenous variables of the system as the leader and the followers, and the remaining variables are classified as exogenous. The leading vehicle has its own acceleration, speed and position (the variables are named *leader.acceleration*, *leader.speed* and *leader.position*, respectively). Furthermore, the exogenous variable u_1 captures the leader acceleration (which is a abstraction of the car pedal position, wind speed, friction, tire conditions) and serves as input to the system. This variable is then linked to *leader.acceleration*. The remaining exogenous variables are related to communication rates.

- $\mathcal{V} = \{leader,\ f1,\ f2,\ f3,\ f4\}$
- $\mathcal{U} = \{T_{min},\ T_{max},\ v_{min},\ d_{min}, u1\}$

Furthermore, the structural equations are defined as follows. For simplicity, we choose to omit the equations for the second, third and fourth follower, but they follow the same pattern as the equations for the first follower.

- $\mathcal{F}_{leader.acceleration}(...) = u_1$
- $\mathcal{F}_{leader.speed}(...) = 22.2 + leader.acceleration * t$
- $\mathcal{F}_{leader.position}(...) = 0 + (22.2 * t + 0.5 * leader.acceleration * t^2)$
- $\mathcal{F}_{f1.acceleration}(...) = a_{IDM}(leader.position, f1.speed, \mid f1.speed - v_0 \mid)$
- $\mathcal{F}_{f1.speed}(...) = 22.2 + f1.acceleration * t$
- $\mathcal{F}_{f1.position}(...) = 0 + (22.2 * t + 0.5 * f1.acceleration * t^2)$
- $\mathcal{F}_{f1.v_0}(...) = 22.2$
- $\mathcal{F}_{f1.\delta}(...) = 4$
- $\mathcal{F}_{f1.T}(...) = 1.5$

- $\mathcal{F}_{f1.s_0}(...) = 2$
- $\mathcal{F}_{f1.a}(...) = 4$
- $\mathcal{F}_{f1.b}(...) = 3$

6.3.2 Results

In both cases (increased and decreased frequency), the analysis found that the variable *leader.acceleration* was a possible cause of the undesired behaviour, more specifically, its rate (i.e., the jolt, the third derivative of position). The sharp acceleration, followed by a deceleration from the leading vehicle, triggers a high frequency of packet transmission for a long time, which causes channel congestion. A congested channel increases the rate of packet loss, which measures the number of packets that do not arrive to the destination over the total number of packets that were sent. When this happens, the vehicles do not receive information in time, which increases the likelihood of collisions, as well as going out of range. Figure 21 shows an analysis of the packet loss rate and how it grows to high levels as the simulation goes on.

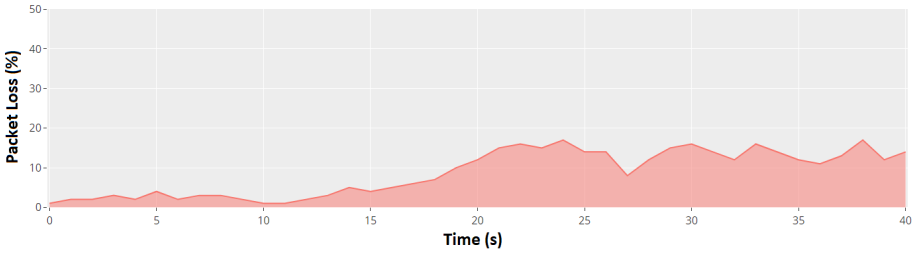


Figure 21: Packet loss analysis.

By overriding the leader acceleration, we decrease the acceleration rate, which in turn results in a smoother response from the followers, and thus prevents the hazards from occurring. Figure 22 shows the trajectory overriding in the $[20, 30)$ time interval.

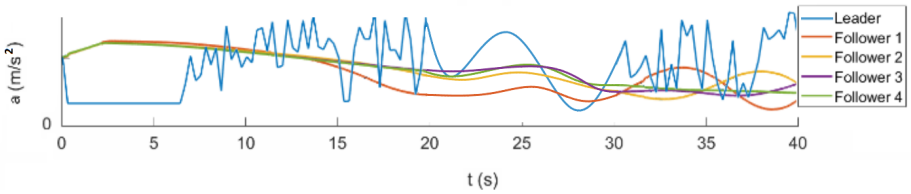


Figure 22: Overriding the leader trajectory.

Given that high channel congestion creates a hazard when vehicles are travelling at high speed, we would like to avoid that. As a way to correct our model, we have made adjustments such that whenever a high value for packet loss and channel congestion is detected, the model limits the acceleration rate until packet loss decreases to acceptable levels.

6.4 Benchmarks

Here, we discuss the result of an experiment using CPS benchmarks found in the literature[3] to assess and report on the scalability of our strategy. The benchmarks that are presented here can be used to assess different aspects of CPSs verification. All benchmarks were chosen such that they are scalable in different problem dimensions.

The benchmarks originate from a variety of domains with the purpose of testing scalability with respect to the number of variables or locations. We make use of three benchmarks and each one assess a different scalability aspect.

This collection is organised by model and complexity. To ensure comparability of results, the specifications are unambiguous and formally described in hybrid automata [6] which is a state-based modelling formalism for CPSs. One of the most constraining problem dimensions in hybrid systems verification are the number of continuous variables and the number of states (or locations) and transitions. All models are available with the HyConf tool[4]. The aim of this experiment is to evaluate the performance of the causal analysis. We measure its efficiency using 3 benchmarks systems. Moreover, each system comprises four variations that are incrementally complex (see Table 8).

Table 8: Overview of the benchmark systems.

System	Variables	Locations	Transitions
Glycemic Control I	11	6	10
Glycemic Control II	11	9	18
Glycemic Control II	11	12	32
Glycemic Control IV	11	15	56
Filtered Oscillator I	6	4	4
Filtered Oscillator II	10	4	4
Filtered Oscillator III	18	4	4
Filtered Oscillator IV	34	4	4
Two-tank system I	4	8	28
Two-tank system II	10	16	66
Two-tank system III	16	24	128
Two-tank system IV	22	34	296

The Glycemic Control example is used to evaluate the increase in the number of locations and transitions whilst the number of variables remains constant. Conversely, the filtered oscillator is used to asses the increase in the number of variables whilst the structural components (i.e., locations and transitions) remain constant. Finally, the two-tank system is used to assess the increase in both the number of variables and number of locations and transitions, simultaneously.

3. https://cps-vo.org/group/ARCH/benchmarks
4. https://github.com/hlsa/HyConf

6.4.1 Methodology

The main goal of this study is the performance evaluation of our causal analysis. This experiment aims to verify whether the duration of the analysis grows linearly with the size of the subject system and the number of endogenous variables selected. Another motivation behind this study is the fact that there are few empirical and controlled experiments to evaluate the efficiency of causal analysis in general.

- **RQ1:** Does the duration of the causal analysis grow linearly with the number of selected endogenous variables?
- **RQ2:** Does the duration of the causal analysis grow linearly with the number of variables in the subject system?
- **RQ3:** Does the duration of the causal analysis grow linearly with the number of locations and transitions in the subject system?

We would like to note that it is possible to increase the number of endogenous variables without increasing the number of overall variables in the system by moving variables from the exogenous set of variables to the endogenous one. Conversely, it is also possible to increase the number of system variables without increasing the number of endogenous ones by adding them to the exogenous set.

We analyse the efficiency of our strategies using mutation analysis. The mutation operators used in this experiment were chosen based on a study on mutation operators for critical systems [16]. We introduce five mutations for each system; all of the mutations change the value of variables in the systems. Then, we divide the executions in 3 stages. In each stage, we iteratively increment the number of endogenous variables considered in the search by a third of the total system variables. That is, in first stage, a third of the variables are considered as endogenous. In the second stage, we consider two thirds of the variables, and finally, in the third stage, every system variable is considered as endogenous. This choice is made randomly. We, then, collect the mean time to determine causes, which is followed by a statistical analysis of the results.

6.4.2 Hypotheses

In order to answer our research questions, we define the metric TDC, which represents the **t**ime to **d**etermine a **c**ause. Hypotheses A, B, and C aim to evaluate the research questions that have been explained previously, respectively. For this, null hypotheses are defined, which state that the duration of the causal analysis does not grow linearly with the increase in the complexity of the analysis. This experiment aims to refute such hypotheses. Thus, alternative hypotheses are also defined, which have a complementary role to the null hypotheses, and can be accepted in case its counterpart hypotheses are rejected. We define 6 hypotheses: A0 and A1 (null and alternative, respectively), checks whether the TDC grows linearly with the number of endogenous variables. Analogously, hypotheses B0 and B1 (null and alternative, respectively) consider the number of variables in the systems. Lastly, C0 and C1 consider the number of locations and transitions in the system.

- H_{A0}: The TDC grows linearly with the number of selected endogenous variables.

- H_{A1}: The TDC does not grow linearly with the number of selected endogenous variables.
- H_{B0}: The TDC grows linearly with the number of variables in the system.
- H_{B1}: The TDC does not grow linearly with the number of variables in the system.
- H_{C0}: The TDC grows linearly with the number of locations and transitions in the system.
- H_{C1}: The TDC does not grow linearly with the number of locations and transitions in the system.

6.4.3 Threats to validity

Here we list the threats to validity that apply to this experiment. As **Internal Validity**, the mutation operators used in this experiment were chosen based on a study on mutation operators for safety critical systems [16] and the number of inserted faults is decided manually. Concerning **External Validity**, this experiment only considers 3 systems; we cannot generalise the outcome of this experiment for a general class of CPSs. Besides, since we introduced the faults ourselves, the mutants may also not represent real world faults. As **Construct Validity**, the values of the mutants (i.e., the degree of the fault) have an impact on how difficult are for causes to be determined. Furthermore, the choice of endogenous variables has a big impact on the performance, as the cause may not be among the chosen variables.

6.4.4 Results

Table 9 shows a summary of the results. For each row of Table 9 we show the respective system along with the number of variables, the number of locations and the number of transitions respectively. For instance, "Glycemic Control I (11 - 6 - 10)" represents the Glycemic Control I model that comprises 11 variables, 6 locations and 10 transitions. Furthermore, we depict the stage along with its respective number of endogenous variables and the mean time for the causality assessment. We would like to note that the time reported is the maximum time taken to analyse the system. We do not stop the search when a cause is found, even though there is only one cause by the experiment design.

The algorithm's time complexity seems to increase not linearly with the number of selected endogenous variables but linearly with the number of variables and locations in the automaton. The time reported in Table 9 is displayed in minutes and considers both the search and simulation times. For statistical significance, each time reported is computed as an average of 5 executions. In each of these five executions, the selected endogenous variables ("End. Vars.") remain the same.

We use the Glycemic Control model to study the effects of increasing locations and transitions but keeping the total number of variables constant. For each variation (I through IV), the results indicate a linear growth in the time taken, due to the increase in operations contained within each added location. For instance, we double the number of locations from 6 to 12 (in Glycemic Control I and Glycemic Control III, respectively) but the time taken to search and assess causality does not double but increases steadily. However, when

Table 9: Benchmark results.

System	Stage I		Stage II		Stage III	
	End. Vars.	**Mean Time**	**End. Vars.**	**Mean Time**	**End. Vars.**	**Mean Time**
Glycemic Control I (11 - 6 - 10)	4	6.7 min	8	18.2 min	11	41.6 min
Glycemic Control II (11 - 9 - 18)	4	8.2 min	8	18.9 min	11	43.7 min
Glycemic Control III (11 - 12 - 32)	4	9.2 min	8	20.1 min	11	44.9 min
Glycemic Control IV (11 - 15 - 56)	4	10.4 min	8	21.6 min	11	46.3 min
Filtered Oscillator I (6 - 4 - 4)	2	3.7 min	4	4.8 min	6	8.9 min
Filtered Oscillator II (10 - 4 - 4)	3	8.9 min	7	14.8 min	10	23.5 min
Filtered Oscillator III (18 - 4 - 4)	6	24.2 min	12	50.1 min	18	107.8 min
Filtered Oscillator IV (34 - 4 - 4)	11	53.9 min	23	96.4 min	34	192.8 min
Tank system I (4 - 8 - 28)	2	4.4 min	3	10.2 min	4	22.6 min
Tank system II (10 - 16 - 66)	4	13.1 min	7	28.7 min	10	89.7 min
Tank system III (16 - 24 - 128)	5	34.2 min	11	71.3 min	16	156.4 min
Tank system IV (22 - 34 - 296)	7	58.2 min	15	132.1 min	22	253.5 min

doubling the number of selected endogenous variables (from 4 to 8), the time taken increases considerably. See, for instance, Glycemic Control I, where the mean time goes from 6.7 minutes to 18.2 minutes.

The filtered oscillator model is used to assess the growth in the number of total variables in the system whilst the number of discrete locations and transitions remain the same. Similarly to the Glycemic Control model, a growth in the number of variables (and, with it, the number of operations) increases the search time linearly. For instance, when the number of selected endogenous variables remain the same, we notice only a modest increase in the amount of time taken when our strategy is applied to Filtered Oscillator II (10 system variables) compared to Filtered Oscillator III (18 system variables).

The findings obtained by analysing the previous models, can also be observed with the tank system. This time, both the total number of variables and the number of locations and transitions are increased and the time taken increases linearly if the number of selected endogenous variables remains the same. The tank system can be used to analyse all three metrics of this benchmark.

The observations of this experiment indicate that increasing the complexity of the subject systems increases the time taken linearly. However, increasing the number of selected endogenous variables has a much greater impact on the overall efficiency of the strategy. Thus, we refute hypothesis H_{A0}, however we accept hypotheses H_{B0} and H_{C0}.

Finally, we consider how Halpern has calculated the complexity of determining causality as an NP-complete problem in terms of complexity[37]. Note that, in Halpern's calculations, the number of exogenous variables does not play a role in the complexity as they are all grouped into the context (u). What plays a role is the number of variables in the cause and the number of endogenous variables (particularly the size of the sets X, W, and Z - all subsets of the endogenous set). Thus, even though we have observed linear growth in some circumstances, this was only observed when the overall size of the system increased but the number of endogenous variables remained the same (i.e., the additional variables were all added to the set of exogenous ones). When the number of endogenous variables increases,

then the time increases non-linearly. Moreover, we emphasise the limitations regarding the choice of endogenous variables and the maximum number of variables in a cause that were applied to our strategy in order to achieve these numbers.

7. Conclusions

Causal analysis is an essential ingredient of counterexample analysis conducted after verification techniques, providing an effective technique to isolate and eventually remove hazards in the design and failures observed in the verification process.

In this work, we propose a formal theory and a practical analysis technique for assessing causality in continuous systems. We attained the following results: 1) we extended an existing theory of actual causality [39] to cope with cyber-physical systems 2) we developed a process to apply the theory of causality and developed and integrated the algorithms in our tool (HyConf [8]), and 3) we applied the developed technique to two case studies to evaluate its effectiveness.

As for future work, in the case where multiple causes are provided, we aim to rank the causes based on a weighted criteria (e.g., cost or responsibility [22]). Moreover, the notation for describing the effects (i.e, Φ) should be extended to consider the temporal quantifiers of Signal Temporal Logic [48]. Furthermore, there is an important discussion to be had with respect to the use of (good) causal models. In our strategy, causal models will determine whether an appropriate cause can be found and, thus, the choice of causal model is of high importance. An option is to obtain and using input-output relationship descriptions from hybrid system models [50], which can assist the user with building relevant causal models.

Currently, concerning the theory, we allow R to be any trajectory vector that obeys the signal types, but without considering any further constraints of the specification besides maximum and minimum values for variables. This may result in causes that do not obey specified dynamics. We aim to address this issue as future work by, during the search, limiting the derivatives to reasonable boundaries obtained by, for instance, exploring hybrid automata [6] models that can be fed as input.

Lastly, despite our measures to address performance-related issues, we believe it is possible to further improve efficiency of the tool. We aim to do this by studying the effect of different search strategies as well as analysing the impact of the length of the trajectories involved in the causal analysis (e.g., the one that contains the fault and the ones that contain the causes).

Acknowledgments

Hugo Araujo and Mohammad Reza Mousavi have been partially supported by the UKRI Trustworthy Autonomous Systems Node in Verifiability, Grant Award Reference EP/V026801/2.

References

Global optimisation toolbox. `https://uk.mathworks.com/products/global-optimization.html`, 2021.

Houssam Abbas, Bardh Hoxha, Georgios E. Fainekos, J. V. Deshmukh, James Kapinski, and Koichi Ueda. Conformance testing as falsification for cyber-physical systems. In *Proceedings of the ACM/IEEE 5th International Conference on Cyber-Physical Systems (ICCPS 2014)*, page 211. IEEE, 2014.

Houssam Abbas, Bardh Hoxha, Georgios E. Fainekos, J. V. Deshmukh, James Kapinski, and Koichi Ueda. WiP abstract: Conformance testing as falsification for cyber-physical systems. In *Proceedings of the ACM/IEEE 5th International Conference on Cyber-Physical Systems (ICCPS 2014)*, page 211. IEEE CS, 2014. Available online: http://arxiv.org/abs/1401.5200.

Rui Abreu, Peter Zoeteweij, Rob Golsteijn, and Arjan JC Van Gemund. A practical evaluation of spectrum-based fault localization. *Journal of Systems and Software*, 82(11):1780–1792, 2009.

Rajeev Alur. *Principles of cyber-physical systems*. MIT Press, 2015.

Rajeev Alur, Costas Courcoubetis, Thomas A Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid systems*, pages 209–229. Springer, 1993.

Yashwanth Annpureddy, Che Liu, Georgios Fainekos, and Sriram Sankaranarayanan. S-TaLiRo: A tool for temporal logic falsification for hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 254–257. Springer, 2011.

Hugo Araujo, Gustavo Carvalho, Mohammad Mousavi, and Augusto Sampaio. Multi-objective search for effective testing of cyber-physical systems. In *Proceedings of the 17th International Conference on Software Engineering and Formal Methods*. Springer, 2019.

Hugo Araujo, Ties Hoenselaar, Mohammad Reza Mousavi, and Alexey Vinel. Connected automated driving: A model-based approach to the analysis of basic awareness services. In *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1–7. IEEE, 2020.

Christel Baier, Clemens Dubslaff, Florian Funke, Simon Jantsch, Rupak Majumdar, Jakob Piribauer, and Robin Ziemek. From verification to causality-based explications. 2021.

Christel Baier, Clemens Dubslaff, Florian Funke, Simon Jantsch, Jakob Piribauer, and Robin Ziemek. Operational causality–necessarily sufficient and sufficiently necessary. In *A Journey from Process Algebra via Timed Automata to Model Learning: Essays Dedicated to Frits Vaandrager on the Occasion of His 60th Birthday*, pages 27–45. Springer, 2022.

Christel Baier, Florian Funke, Jakob Piribauer, and Robin Ziemek. On probability-raising causality in Markov decision processes. In *FoSSaCS*, pages 40–60, 2022.

Sander Beckers and Joseph Y Halpern. Abstracting causal models. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 2678–2685, 2019.

Ilan Beer, Shoham Ben-David, Hana Chockler, Avigail Orni, and Richard Trefler. Explaining counterexamples using causality. In *International Conference on Computer Aided Verification*, pages 94–108. Springer, 2009.

Carl Bergenhem, Steven Shladover, Erik Coelingh, Christoffer Englund, and Sadayuki Tsugawa. Overview of platooning systems. In *Proceedings of the 19th ITS World Congress, Oct 22-26, Vienna, Austria (2012)*, 2012.

Nguyen Thanh Binh et al. Mutation operators for Simulink models. In *Knowledge and Systems Engineering (KSE), 2012 Fourth International Conference on*, pages 54–59. IEEE, 2012.

Hichem Boudali, Pepijn Crouzen, and Mariëlle Stoelinga. A compositional semantics for Dynamic Fault Trees in terms of Interactive Markov Chains. In *International Symposium on Automated Technology for Verification and Analysis*, pages 441–456. Springer, 2007.

Georgiana Caltais, Mohammad Reza Mousavi, and Hargurbir Singh. Causal reasoning for safety in Hennessy-Milner logic. *Fundamenta Informaticae*, 173(2-3):217–251, 2020.

Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *International Conference on Computer Aided Verification*, pages 258–263. Springer, 2013.

Ferdinando Chiacchio, Diego D'Urso, Lucio Compagno, Marzio Pennisi, Francesco Pappalardo, and Gabriele Manno. SHyFTA, a stochastic hybrid fault tree automaton for the modelling and simulation of dynamic reliability problems. *Expert Systems with Applications*, 47:42–57, 2016.

Hana Chockler, Orna Grumberg, and Avi Yadgar. Efficient automatic STE refinement using responsibility. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 233–248. Springer, 2008.

Hana Chockler and Joseph Y Halpern. Responsibility and blame: A structural-model approach. *Journal of Artificial Intelligence Research*, 22:93–115, 2004.

Ziquan Deng, Samuel P Eshima, James Nabity, and Zhaodan Kong. Causal signal temporal logic for the environmental control and life support system's fault analysis and explanation. *IEEE Access*, 11:26471–26482, 2023.

Alexandre Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *International Conference on Computer Aided Verification*, pages 167–170. Springer, 2010.

Clemens Dubslaff, Kallistos Weis, Christel Baier, and Sven Apel. Causality in configurable software systems. *arXiv preprint arXiv:2201.07280*, 2022.

Joanne Bechta Dugan, Salvatore J Bavuso, and Mark A Boyd. Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on reliability*, 41(3):363–377, 1992.

Clifton A Ericson and Clifton Ll. Fault tree analysis. In *System Safety Conference, Orlando, Florida*, volume 1, pages 1–9, 1999.

ETSI EN 302 637- 2; Intelligent Transport Systems (ITS); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service, 2013.

Georgios E Fainekos and George J Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.

Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In *International Conference on Computer Aided Verification*, pages 379–395. Springer, 2011.

Nathan Fulton, Stefan Mitsch, Jan-David Quesel, Marcus Völp, and André Platzer. Keymaera x: An axiomatic tactical theorem prover for hybrid systems. In *International Conference on Automated Deduction*, pages 527–538. Springer, 2015.

Dan Geiger and Judea Pearl. On the logic of causal models. In *Machine intelligence and pattern recognition*, volume 9, pages 3–14. Elsevier, 1990.

Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pages 424–438, 1969.

Jan Friso Groote and Mohammad Reza Mousavi. *Modeling and analysis of communicating systems*. MIT press, 2014.

Philip George Guest and Philip George Guest. *Numerical methods of curve fitting*. Cambridge University Press, 2012.

Joseph Halpern. A modification of the Halpern-Pearl definition of causality. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Joseph Y Halpern. *Actual causality*. MiT Press, 2016.

Joseph Y Halpern and Judea Pearl. Causes and explanations: A structural-model approach - part I: Causes. In *Proc. Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI 2001)*, pages 194—-202, 2001.

Joseph Y Halpern and Judea Pearl. Causes and explanations: A structural-model approach. part I: Causes. *The British journal for the philosophy of science*, 56(4):843–887, 2005.

Joseph Y Halpern and Judea Pearl. Causes and explanations: A structural-model approach. part II: Explanations. *The British journal for the philosophy of science*, 56(4):889–911, 2005.

Daniel Murray Hausman. Causal relata: Tokens, types, or variables? *Erkenntnis*, 63(1):33–54, 2005.

Matthew Hennessy and Robin Milner. On observing nondeterminism and concurrency. In *International Colloquium on Automata, Languages, and Programming*, pages 299–309. Springer, 1980.

Amjad Ibrahim, Severin Kacianka, Alexander Pretschner, Charles Hartsell, and Gabor Karsai. Practical causal models for cyber-physical systems. In *NASA Formal Methods Symposium*, pages 211–227. Springer, 2019.

Sebastian Junges, Joost-Pieter Katoen, Mariëlle Stoelinga, and Matthias Volk. One net fits all: a unifying semantics of dynamic fault trees using GSPNs. In *Application and Theory of Petri Nets and Concurrency: 39th International Conference, PETRI NETS 2018, Bratislava, Slovakia, June 24-29, 2018, Proceedings 39*, pages 272–293. Springer, 2018.

Bernhard Kaiser, Catharina Gramlich, and Marc Förster. State/event fault trees—a safety analysis model for software-controlled systems. *Reliability Engineering & System Safety*, 92(11):1521–1537, 2007.

Edward Ashford Lee and Sanjit A Seshia. *Introduction to embedded systems: A cyber-physical systems approach*. Mit Press, 2016.

Florian Leitner-Fischer and Stefan Leue. Causality checking for complex system models. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, pages 248–267. Springer, 2013.

Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166. Springer, 2004.

Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

Stefan Mitsch and André Platzer. ModelPlex: Verified runtime validation of verified cyber-physical system models. *Formal Methods in System Design*, 49(1):33–74, 2016.

Pieter J Mosterman and Justyna Zander. Cyber-physical systems challenges: a needs analysis for collaborating embedded software systems. *Software & Systems Modeling*, 15(1):5–16, 2016.

Olaf Müller and Thomas Stauner. Modelling and verification using linear hybrid automata–a case study. *Mathematical and Computer Modelling of Dynamical Systems*, 6(1):71–89, 2000.

Girish Keshav Palshikar. Temporal fault trees. *Information and Software Technology*, 44(3):137–150, 2002.

Judea Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688, 1995.

Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

Judea Pearl. The seven tools of causal inference, with reflections on machine learning. *Communications of the ACM*, 62(3):54–60, 2019.

Judea Pearl et al. Causal inference in statistics: An overview. *Statistics surveys*, 3:96–146, 2009.

Spencer Peters and Joseph Y. Halpern. Causal modeling with infinitely many variables, 2021.

Hendrik Roehm, Jens Oehlerking, Matthias Woehrle, and Matthias Althoff. Model conformance for cyber-physical systems: A survey. *ACM Trans. Cyber-Phys. Syst.*, 3(3), aug 2019.

Ali Shojaie and Emily B Fox. Granger causality: A review and recent advances. *Annual Review of Statistics and Its Application*, 9:289–319, 2022.

Peter Spirtes. Introduction to causal inference. *Journal of Machine Learning Research*, 11(5), 2010.

Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.

Jan Tretmans. Conformance testing with labelled transition systems: Implementation relations and test generation. *Computer networks and ISDN systems*, 29(1):49–79, 1996.

Michiel Van Osch. Hybrid input-output conformance and test generation. In *Formal Approaches to Software Testing and Runtime Verification*, pages 70–84. Springer, 2006.

Martin Weiglhofer, Bernhard K Aichernig, and Franz Wotawa. Fault-based conformance testing in practice. *Int. J. Softw. Informatics*, 3(2-3):375–411, 2009.

Edward N Zalta, Uri Nodelman, Colin Allen, and John Perry. Stanford encyclopedia of philosophy, 1995.

Zhenya Zhang, Jie An, Paolo Arcaini, and Ichiro Hasuo. Online causation monitoring of signal temporal logic. *arXiv preprint arXiv:2305.17754*, 2023.

Zhenya Zhang, Gidon Ernst, Sean Sedwards, Paolo Arcaini, and Ichiro Hasuo. Two-layered falsification of hybrid systems guided by monte carlo tree search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2894–2905, 2018.

Appendix A. List of operators

X	Variable.
$\vec{X}$	List/Set of variables.
x	Trajectory.
$\vec{x}$	List/Set of trajectories.
$x \downarrow_{\vec{X}}$	Trajectory restricted to the set of variables $\vec{X}$.
$x_{[i,j)}$	Trajectory slice.
$x =_{[i,j)} y$	The trajectories x and y are equals when restricted to the set $\vec{Z}$ and within [i,j).
$c \hookleftarrow \vec{x}$	The values in the list of trajectory $\vec{x}$ are equals to the values in c.
$M_{\vec{X} \leftarrow \vec{x}}$	Filtering the trajectories of the variables in $\vec{X}$ with the trajectories in $\vec{x}$.

Appendix B. Soundness

In order to discuss the soundness of our implementation of the definition of cause, we provide a one-to-one compositional mapping between the constructs used in Algorithm 2 and the concepts introduced in Definition 18. Firstly, we provide Proposition 1 that makes the mapping easier to understand.

Proposition 1. *In our implementation,* $(M, u) \models (c \hookleftarrow \vec{x})$ *always holds by construction.*

Proof. In our implementation, the causal model M is built independently of a system execution. However, once a trajectory (c) that leads to a failure (Φ) is found, the trajectory that corresponds to the exogenous variables (u) is built by applying the projection ($c \downarrow_{\mathcal{U}}$). Thus (M, u) is always built around c. Furthermore, the list of trajectory slices in the cause ($\vec{x}_X$) is, by construction, taken from $c \downarrow_{\mathcal{V}}$. Thus, Proposition 1 follows. □

In what follows, we provide a one-to-one compositional mapping of Definition 18 into Algorithm 2 (see Table 10) and explain it below. For that, we rely on the assumption that the inputs (generated by our search heuristic in Algorithm 1) are generated appropriately from the models and that the causal model conforms with the system. In the preamble for the definition of cause, a causal model M, a trajectory c, and a conjunction of Boolean predicates Φ are given; analogously, they are given as input for our algorithm. Furthermore, the definition requires a list of variables $\vec{X} \in \mathcal{V}$ that is taken from the causal model in the search algorithm. Lastly, the theory also calls for a list of trajectory slices $\vec{x}$. In both the algorithm and the definition, to determine cause, three clauses must be satisfied (4th row of Table 10).

The mapping of clause AC1 is straightforward. By construction (in Algorithm 1), $\vec{x}$ is taken from c and, thus, it suffices for Algorithm 2 to only evaluate whether Φ holds in the causal model. As for AC2, the set $\vec{W}$ is also given as input. An alternative trajectory is the result of overriding c with the slices $\vec{x}'$ and $\vec{w}$, which is done via the model update (9th row). This is used to check that Φ does not hold in this updated model. Lastly, AC3 checks minimality by searching for causes that are subset of the prospective one. This new search adds trajectories to $\mathcal{R}$, which then confirms soundness as, according to Definition 18, we need to check every $\vec{w} \subseteq \mathcal{R} \cap \text{Trajs}(\mathcal{W})$.

Table 10: Mapping between Algorithm 2 and Definition 18.

Given a causal model $M = \langle\langle \mathcal{U}, \mathcal{V}, \mathcal{R}\rangle, \mathcal{F}\rangle$	**input :** CausalModel M;
a setting $u \in \mathit{Trajs}(\mathcal{U})$, a trajectory $c : \mathit{Trajs}(\mathcal{U} \cup \mathcal{V})$ such that $c \downarrow_{\mathcal{U}=\mathrm{dom}(u)} u$	**input :** Trajectory c, u;
a list of variables $\vec{X}$,	**input :** Set [Variable] $\vec{X}$;
and a set of trajectory slices $\vec{x} = \{x \mid x \in \mathit{Trajs}(\mathcal{V})\}$,	**input :** Set [Trajectory] $\vec{x}$;

then $(c \leftarrowtail \vec{x})$ is a cause of Φ in (M, u) when the following three conditions hold:	**Function** `IsCause`(M, u, Φ, $\vec{x}$, $\vec{x}'$, $\vec{w}$, $\vec{X}$, $\vec{W}$) **:**
	  $isCause$ = False;
	  if `SatisfiesACOne`(M, u, Φ) **then**
	  if `SatisfiesACTwo`(M, u, Φ, $\vec{x}'$, $\vec{w}$, $\vec{X}$, $\vec{W}$) **then**
	  if `SatisfiesACThree`(M, u, Φ, $\vec{x}$) **then**
	  $isCause$ = True;
	  end
	  end
	  end
	  return $isCause$;
	end

AC1. $(M, u) \models (c \leftarrowtail \vec{x}) \wedge \Phi$	**Function** `SatisfiesACOne`(M, u, Φ) **:** **return** `Holds`(M, u, Φ); **end**
AC2. There exists a set of variables $\vec{W} \subset \mathcal{V}$	**input :** Set [Variable] $\vec{W}$;
and two sets of trajectories $\vec{x}' \in \mathit{Alts}(\vec{x})$, and $\vec{w} \subseteq \mathit{Trajs}(\vec{W}) \cap \mathcal{R}$ such that if $(M, u) \models (c \leftarrowtail \vec{w})$, then:	**input :** Set [Trajectory] $\vec{x}$; **input :** Set [Trajectory] $\vec{w}$;

$(M_{\vec{X} \leftarrow \vec{x}', \vec{W} \leftarrow \vec{w}}, u) \models \neg\Phi$	**Function** `SatisfiesACTwo`(M, u, Φ, $\vec{x}'$, $\vec{w}$, $\vec{X}$, $\vec{W}$) **:**
	  M_C^{updt} = `UpdateModel`(M,{($\vec{X}$, $\vec{x}'$), ($\vec{W}$, $\vec{w}$)});
	  $isAC2aTrue$ = `Holds`(M_C^{updt}, u, $\neg\Phi$);
	  return $isAC2aTrue$;
	end

AC3. There is no strict subset of $\vec{x}$ that satisfies AC1 and AC2	**Function** `SatisfiesACThree`(M, u, Φ, $\vec{x}$,) **:**
	  foreach $subOfX$ ***in*** `GetSubsets`($\vec{x}$) **do**
	  ($\vec{x}'$, $\vec{w}'$) = `SearchHeuristic`(c, $subOfX$, $interval$);
	  $\vec{X}'$ = $\vec{x}'$.`GetVariables`();
	  $\vec{W}'$ = $\vec{w}'$.`GetVariables`();
	  if `SatisfiesACOne`(M, u, Φ) **then**
	  if `SatisfiesACTwo`(M, u, Φ, $\vec{x}'$, $\vec{w}'$, $\vec{X}'$, $\vec{W}'$) **then**
	  return False;
	  end
	  end
	  end
	  return True;
	end

The computational complexity of determining a cause is considered intractable (NP-complete for some cases [37]). And it is due to this reason that we apply some limitations to our strategy (as highlighted in Sections 5.1 and 5.2) in order to reach a verdict (e.g., the number of endogenous variables and a maximum number on interactions between variables

during the search). Hence, our algorithm only provides an approximation to the solution of the actual problem. Since we impose finite limits to the number of chosen variables, the granularity and number of slices, and the number of interactions between variables (up to three-wise) when searching, we guarantee that the program will terminate. However, due to the problem's intractability and the pseudo-random nature of the search, we do not guarantee that our approach will find a cause. Our approach is not exhaustive but we have shown via the mapping that our approach is sound with respect to the discretised version of the problem. That is, our approach finds actual causes (with respect to our theory) when one considers the necessary discretisation steps (in which we approximate a strictly continuous problem into a discrete one), which are needed for simulation of CPSs in physical machines.