Causal Liability in Autonomous Systems

Kaveh Aryan$^{1\,[0009-0003-7245-4677]}$, Hana Chockler$^{1\,[0000-0003-1219-0713]}$, and Mohammad Reza Mousavi$^{1\,[0000-0002-4869-6794]}$

King's College London, London, UK
{kaveh.aryan,hana.chockler,mohammad.mousavi}@kcl.ac.uk

Abstract. With the widespread use of autonomous systems, liability often shifts towards the manufacturers and part suppliers, especially when various system components, sourced from different suppliers, contribute to a failure. This necessitates a framework for automatic liability apportionment which can be embedded in manufacturer-supplier contracts and minimise legal disputes. To this end, we propose a formal framework based on the notion of actual causality in structural causal models and the robustness semantics of logical specifications. We prove several desirable properties of this framework. Moreover, we formalise the notion of causal non-interaction, give sufficient conditions for it to hold, and demonstrate its utility in deriving upper bounds for liability analysis. Furthermore, we relate our definition to the existing notion of harm, empirically evaluate our framework to demonstrate its efficacy, and release a software package implementing our approach. We extend our framework to handle interval specifications.

Keywords: formal methods · causality · liability

1 Introduction [1]

The widespread adoption of autonomous systems, including autonomous vehicles (AVs) and service robots, significantly reshapes the insurance landscape [2]. Traditionally, liability rested primarily with system users and owners; in the case of reckless driving, the driver is typically held legally liable and is covered by personal auto insurance that provides protection against such risks [1]. In contrast, autonomous systems aim to function safely and reliably on their own, increasingly shifting liability toward manufacturers and their suppliers of mechanical, electronic, and software components. Manufacturers, even after indemnifying suppliers externally, still need effective means to recoup costs associated with system faults.

To fairly and automatically apportion liability within complex multi-supplier ecosystems, we propose a framework based on structural causal models (SCMs) [26] and robustness semantics of logical specifications [13]. Our method systematically apportions liability among components, allowing integration into contracts between manufacturers and suppliers so that they incorporate both component

[1] We gratefully acknowledge and appreciate Prof. Özlem Gürses, Dickson Poon School of Law, King's College London for their valuable insights.

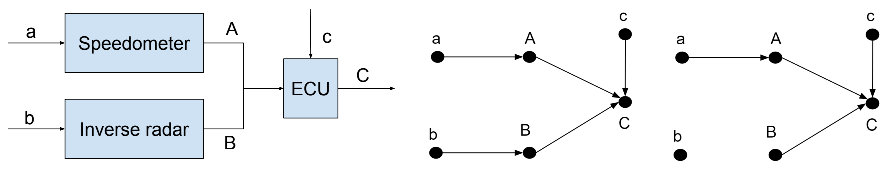

Fig. 1: Simplified AEB system in Examples 1 and 6. System (left), graph without intervention (middle), and with intervention (right). An *intervention* forces the value of a component to a fixed value, overriding its normal functional dependencies.

specifications as well as the rules for liability apportionment in cases of failure. Incorporating formal causality into contractual terms minimises legal disputes and enhances the insurability of advanced autonomous technologies [22].

A key challenge is ensuring the framework remains meaningful and tractable in complex systems with interacting faults. This issue is prominently manifest in existing approaches such as Shapley values [12]. To address this, we introduce the concept of *causal non-interaction*, provide conditions for its application, and demonstrate its utility for bounding liability calculations.

We also show that our framework relates to existing literature, specifically aligning with harm quantification by Beckers, Chockler, and Halpern (BCH) [6]. We empirically evaluate our model to underscore its practical applicability in facilitating precise and fair liability apportionment. We offer a Python toolbox implementing our liability measure (DOI: `https://doi.org/10.5281/zenodo.18175468`).

We illustrate our approach using an Automatic Emergency Braking (AEB) system example:

Example 1. Consider an AEB system comprising a speedometer, radar, and an Electronic Control Unit (ECU) calculating the braking force (Fig. 1 (left)). In the event of a failure–such as those documented in high-profile autonomous vehicle (AV) crashes[2], our framework identifies which component failures cause malfunctions, quantifies respective liabilities, and enables embedding these apportionments into manufacturer-supplier contracts.

While motivated by AV liability, our framework addresses the broader problem of attributing responsibilities in any component-based system. Our framework focuses on apportioning liability among system components conditional on a system-level failure having occurred. External or unavoidable causes, such as extreme environmental conditions or third-party actions, can be naturally modeled in our framework as exogenous variables in the causal model. If such factors alone suffice to cause the failure, no internal component forms a cause and the resulting liabilities are zero

[2] e.g., `https://www.ntsb.gov/investigations/Pages/HWY18MH010.aspx`

The paper proceeds as follows: Section 2 reviews the related literature; Section 3 reviews the preliminaries, i.e., component model, failure quantification, structural causal models, and actual causality; Section 4 defines liability formally and explores its properties and the notion of causal non-interaction; Section 5 extends our framework to handle interval specifications; Section 6 establishes the formal relation of our work to the BCH notion of harm; Section 7 presents the empirical studies; and Section 8 concludes.

2 Related Work

2.1 Actual causality

Liability inherently involves causal reasoning [21]. There are two types of causal relations: *type causality*, relating general phenomena (e.g., smoking and lung cancer), and *actual causality*, concerning specific events (e.g., a particular radar defect causing a crash) [15, 33]. Actual causality is pertinent for liability apportionment.

The simplest explication of actual causality is *but-for* causality [25]—an event without which the effect wouldn't occur. There are also more sophisticated definitions like Halpern-Pearl (HP) [16, 18] definitions, which, although influential, are computationally demanding—HP is D_2^P [3] or D_1^P based on the version of the definition [16]. We base our efficient approach on but-for causality, although our framework is general and accommodates any reasonable notion of actual causality.

2.2 Causality in liability law

Actual causality (cause in fact) underpins liability law (criminal, tort, insurance) [21]. Liability apportionment typically involves: (i) dividing damages into indivisible causal components and (ii) apportioning each component, often using subjective rules based on fault [19]. Recent principled yet abstract methods (i.e., Shapley values) are emerging [12, 14, 20]. These works are abstract because they do not address how one can measure the contribution of a component in the overall system failure. Our approach retains this two-step process, contributing a principled, concrete, and computationally efficient liability definition.

2.3 Liability in autonomous systems

The rise of autonomous vehicle functions necessitates new insurance frameworks. Two notable examples are Manufacturer Enterprise Liability [2] and the UK's Automated & Electric Vehicles Act 2018, which faces legal criticism [11]. Manufacturer Enterprise Liability exclusively burdens manufacturers, regardless of fault. Our paper provides a formal approach enabling manufacturers to distribute liability among suppliers, supporting broader insurance adoption.

2.4 Quantification of causality

Although causality is typically binary, liability apportionment requires quantified causality measures [17]. Three prominent quantified causality approaches are responsibility, harm, and Shapley values.

Responsibility and harm Responsibility [8], developed from HP causality, quantifies actual causality as inversely proportional to the number of causes in a causal set [9, 30]. Unlike our method, responsibility addresses actual implementation systems without relying on formal specifications and is computationally expensive. Harm quantifies causality by looking at the outcome differences with and without an event [6]. Our work is a generalisation of the notion of the harm where component interactions are taken into account and failure is quantified by robustness of the system wrt a general failure formula.

Game theoretic approaches Shapley values apportion payoffs in cooperative games based on fairness axioms [7]. They are applied to liability by modelling tortfeasors as game players [12, 14, 20]. Unlike abstract Shapley approaches, we propose a concrete structural causal model-based method with polynomial-time complexity, avoiding Shapley values' exponential complexity.

Remark 1. Type causality quantification significantly differs from actual causality. For example, Pearl's graphical approach employs probability distributions and interventions [26, 27], unlike actual causality focused on individual events.

2.5 Risk assessment

Reliability engineering methods, such as Failure Mode and Effects Analysis (FMEA) [29] and Fault Tree Analysis (FTA) [31], focus on risk identification and modelling *failure propagation* via logical gates. Our method, in contrast, mathematically models causal structures and assigns liability after failures, differing fundamentally in both approach and purpose.

3 Preliminaries

3.1 Component Model

As we are dealing with systems with multiple components, we define formal abstractions for components, systems, and their specifications and implementations, inspired by a similar architecture for software liability [24].

Definition 1. *A* component *is a tuple $X = (\mathcal{I}, O, \mathcal{P}, R, f)$ in which $\mathcal{I}$ is a set of variables called* input variables*, O is a single variable not in $\mathcal{I}$ called the* output variable*, $\mathcal{P}$ is a set of* types*, $R : \mathcal{I} \cup \{O\} \to \mathcal{P}$ is a function that determines the* types *of the input and output variables, and $f : \prod_{Y \in \mathcal{I}} R(Y) \to R(O)$ is the* component's function*. As a notational convenience, when it is clear from the context, we use the component's output variable and the component's name interchangeably. A component is an* implementation *of another component (labelled as* specification*) if they vary only in their functions. A* system *is a set of components $\mathcal{S} = \{(\mathcal{I}_i, O_i, \mathcal{P}_i, R_i, f_i)\}_{i=1}^{N}$ subject to the condition that the output variables of the components are distinct. An* implementation system *consists of the implementations of the components in a* specification system*.*

As evident from the definition, a component is an entity characterised by one or several input variables with specified types, an output variable with a specified type, and a mechanism to transform the inputs into the output. The decision to have a single output variable is primarily driven by notational convenience. However, this choice is not restrictive, as there is always the option to combine multiple variables into a vector.

Example 2. In the context of the AEB system of Example 1, the speedometer component is defined as $X_A = (\{a\}, A, \{\mathbb{R}\}, R_A, f_A)$ in which $R_A(a) = R_A(A) = \mathbb{R}$ and $f_A(a) = a + 10$, where a denotes the actual speed of the vehicle in meters per second and f_A reflects the sensor error in our model. The (inverse) radar component is defined as $X_B = (\{b\}, B, \{\mathbb{R}\}, R_B, f_B)$ in which $R_B(b) = R_B(B) = \mathbb{R}$ and $f_B(b) = b + 1$, where b represents actual inverse distance of the vehicle to an impending obstacle in reciprocal meter. The ECU component is defined as $X_C = (\{A, B, c\}, C, \{\mathbb{R}\}, R_C, f_C)$ in which $R_C(A) = R_C(B) = R_C(c) = R_C(C) = \mathbb{R}$ and $f_C(A, B, c) = AB + c$, where c represents a calibrating constant set at construction time. The term AB shows that the magnitude of the braking force is proportional to the speed of the speedometer and inversely proportional to the distance to the obstacle. Assuming that X_C is a component specification, there can be many different implementations thereof which differ only in their functions. The system consisting of these components is $\mathcal{T} = \{X_A, X_B, X_C\}$. As mentioned earlier, when clear, we use output variables to denote components, so a more succinct notation is $\mathcal{T} = \{A, B, C\}$.

Note that while we present algebraic definitions of specifications and implementations for the purpose of formal reasoning, the proposed framework treats components as black boxes in practice: it only requires query access to component outputs for given inputs, and does not rely on symbolic representations of their internal functions.

It is useful to define the notion of a replacement system, a system in which one or more components is replaced with others.

Definition 2. *If $\mathcal{S}$ is a system, $\mathcal{X} = \{(\mathcal{I}_i, O_i, \mathcal{P}_i, R_i, f_i)\}_{i=1}^{k} \subseteq \mathcal{S}$ is any set of components in $\mathcal{S}$, and $\mathcal{X}' = \{(\mathcal{I}_i, O_i, \mathcal{P}_i, R_i, g_i)\}_{i=1}^{k}$ is another set of components, then the* replacement system *$S_{\mathcal{X} \to \mathcal{X}'}$ is defined as $(\mathcal{S} - \mathcal{X}) \cup \mathcal{X}'$. If $\mathcal{X} = \{X\}$ and $\mathcal{X}' = \{X'\}$ are singleton sets, we conveniently denote $\mathcal{S}_{\mathcal{X} \to \mathcal{X}'} = \mathcal{S}_{\{X\} \to \{X'\}}$ by $\mathcal{S}_{X \to X'}$. If one or more implementation components are replaced with their corresponding specifications, the replacement is called a* fix.

Example 3. Consider the AEB system $\mathcal{T}$ defined in Example 2. Suppose there is a specification radar component $X'_B = (\{b\}, B, \{\mathbb{R}\}, R_B, g_B(b) = b)$. The replacement system $\mathcal{T}_{X_B \to X'_B}$ is the system $\mathcal{T}$ in which the radar component is replaced with the specification one, that is $\mathcal{T}_{X_B \to X'_B} = \{X_A, X'_B, X_C\}$.

3.2 Quantification of Failure

Two final definitions concern the characterisation and quantification of the extent of failure. Failure is characterised as a quantifier-free first-order formula over the system variables. The extent of failure is quantified by how much these variables need to change to no longer satisfy the failure formula.

Definition 3. *If P is a quantifier-free first-order formula over a set of variables $\mathcal{V}$, the* extension *of P, denoted by $ext(P)$, is the set of points in the $\mathcal{V}$-space that satisfy P. If P characterises a system's failure, it is called a* failure formula *and $\mathcal{F} = ext(P)$ is called a* failure set*.*

Example 4. Following Example 2, suppose $P(A,B,C) = B \geq 10 \wedge C \leq 1$, that is, a failure happens if the detected distance is less than 0.1 meters and the generated braking force is less than 1 units. Then the failure set is $\mathcal{F} = ext(P) = \{(A,B,C) \in \mathbb{R}^3 : B \geq 10 \wedge C \leq 1\}$.

To simplify the presentation, we usually refer to failure sets such as $\mathcal{F}$, implicitly assuming suitable quantifier-free first-order formulas, such as P, for which $\mathcal{F} = ext(P)$.

We quantify the extent of a failure in a system, aka its *robustness*, by measuring the distance of the state of the system to the boundary of the failure set, along the same lines of the robustness semantics for logical specifications [13]. Assuming the reader is familiar with the concept of a metric space [34], we recall the definition of the depth of a point in a set:

Definition 4. *Consider a metric space, $(\mathcal{G},d)$, a point in the space, $x \in \mathcal{G}$, and a subset of the space, $\mathcal{H} \subseteq \mathcal{G}$, then*

- *The* distance *from x to $\mathcal{H}$ is defined as $dist_d(x,\mathcal{H}) = \inf\{d(x,y) : y \in \bar{\mathcal{H}}\}$, where $\bar{\mathcal{H}}$ is the closure of $\mathcal{H}$, that is the intersection of all closed sets containing $\mathcal{H}$; and*
- *The* depth *of x in $\mathcal{H}$ is defined as $depth_d(x,\mathcal{H}) = dist_d(x,\mathcal{G} - \mathcal{H})$.*

The subscript d denotes the distance metric, d, in the metric space, which will be the Euclidean distance throughout the paper. Also, $dist_d$ and $depth_d$ are non-negative numbers; for all $x \in \mathcal{H}$, $dist_d(x,\mathcal{H}) = 0$; and for all $x \notin \mathcal{H}$, $depth_d(x,\mathcal{H}) = 0$.

Example 5. In Example 4, $dist_d((100,5,1),\mathcal{F}) = 5$ and $depth_d((100,5,1),\mathcal{F}) = 0$.

In practice, the failure formula (hence, the failure set) is usually a function of only a few system variables. For instance, in our running example, a reasonable failure formula could be a proposition about the impact velocity (e.g, $P : M_{\text{vehicle}} v \geq 1$) where M_{vehicle} is the vehicle's mass. In such cases, the causal models maps inputs in different dimensions to the same output space through the robustness quantity, making them comparable through the robustness of the variables in the failure formula. This is one of the benefits of our approach, which facilitates straightforward comparisons across different variables and units of measurement.

In the rest of this section we define structural causal models [17, 26] and use them to define a notion of actual causality.

3.3 Structural Causal Models

Our work is based on *structural causal models*, which serve both as a basis for the notion of causality and for our proposed apportionment of liabilities.

Definition 5. *A* structural causal model (SCM) *is a tuple* $\mathcal{M}=(\mathcal{U},\mathcal{V},\mathcal{P},R,\mathcal{E})$ *in which* $\mathcal{U}$ *and* $\mathcal{V}$ *are two disjoint sets of variables called* exogenous *and* endogenous *variables, respectively,* $\mathcal{P}$ *is a set of* types*,* $R:\mathcal{U}\cup\mathcal{V}\to\mathcal{P}$ *is a function that determines the* types *of variables in* $\mathcal{U}$ *and* $\mathcal{V}$*, and* $\mathcal{E}$ *is the set of* structural equations $\{X=f_X(Y_1,Y_2,Y_3,...,Y_n)\}_{X\in V}$ *where* $Y_i\in(\mathcal{U}\cup\mathcal{V})-\{X\}$ *for all* $X\in\mathcal{V}$*.*

Notation 1 *If* $\mathcal{M}=(\mathcal{U},\mathcal{V},\mathcal{P},R,\mathcal{E})$ *is an SCM, we denote members of each set by non-calligraphic face with an integer index, such as* U_i*. Elements in* $\mathcal{V}$*,* $\mathcal{R}$*, and* $\mathcal{E}$ *that share the same index are assumed to correspond. For example,* R_i *and* E_i *denote the type and structural equation of variable* V_i*, respectively.*

The motivation for differentiating between exogenous and endogenous variables is to distinguish between variables that are the inputs of the system and those that are influenced by the inputs, respectively. Structural *equations* (or *assignments*, more precisely) specify how each endogenous variable is influenced by the exogenous and other endogenous variables.

Example 6. In the AEB system in Example 2, each component (causally) determines the value of its output based on its inputs. The whole system corresponds to the following structural causal model: $\mathcal{M} = (\{a,b,c\},\{A,B,C\},\mathbb{R},R_A\cup R_B\cup R_C,\{A=f_A(a),B=f_B(b),C=f_C(A,B,c)\})$.

In line with our focus on particular events and actual causality, we are interested in the particular values, called *state*, that variables of the system take:

Definition 6. *If* $\mathcal{M}=(\mathcal{U},\mathcal{V},\mathcal{P},R,\mathcal{E})$ *is an SCM, a* context $\boldsymbol{u}$ *is a setting of (i.e., a vector assignment to) variables in* $\mathcal{U}$*. Similarly, a* state $\boldsymbol{v}$ *is a setting of variables in* $\mathcal{V}$*. The* state space *of* $\mathcal{M}$*, denoted by* $SS(\mathcal{M})$*, is the set of all states of* $\mathcal{M}$*. Moreover, if* $\boldsymbol{x}$ *is a setting of variables in* $\mathcal{X}\subseteq\mathcal{U}\cup\mathcal{V}$*, the value of the variable* $X\in\mathcal{X}$ *under* $\boldsymbol{x}$ *is denoted by* $\boldsymbol{x}[X]$*.*

An SCM can be visualised through a graph where each variable in $\mathcal{U}\cup\mathcal{V}$ is represented by a node, and a directed edge from X to Y exists iff the value of Y depends, in at least one state and context, on the value of X. Fig. 1 (middle) shows the graph corresponding to the SCM in Example 6. An SCM is acyclic if its corresponding graph is acyclic. If $\mathcal{M}$ is an acyclic SCM, a given context u uniquely determines the state of $\mathcal{M}$. This state is denoted by $\boldsymbol{v}=M[\boldsymbol{u}]$. For the purpose of this paper, we align with existing literature and focus on acyclic SCMs.

Example 7. In the SCM in Example 6, one possible context is $\boldsymbol{u}=(a,b,c)=(80,4,1)$. The resulting state is $\mathcal{M}[\boldsymbol{u}] = (90,5,451)$. The state space of the system is $\{(A=f_A(a),B=f_B(b),f_C(A,B,c)) : a,b,c\in\mathbb{R}\}=\{(a+10,b+1,(a+10)(b+1)+c) : a,b,c\in\mathbb{R}\}$. Also, $\boldsymbol{u}[a]=80$ and $\mathcal{M}[\boldsymbol{u}][A]=90$.

As suggested by Example 6, there is a close connection between our component model and SCMs. As formalised in the following definition, for each system there exists a corresponding SCM:

Definition 7. *Suppose* $\mathcal{S} = \{(\mathcal{I}_i, O_i, \mathcal{P}_i, R_i, f_i)\}_{i=1}^N$ *is a system with* N *components. The* corresponding structural causal model (SCM) *is defined as* $SCM(\mathcal{S}) = (\mathcal{U}, \mathcal{V}, \mathcal{P}, R, \mathcal{E})$ *in which* $\mathcal{V} = \cup_{i=1}^N \{O_i\}$ *is the set of endogenous variables,* $\mathcal{U} = (\cup_{i=1}^N \mathcal{I}_i) - \mathcal{V}$ *is the set of exogenous variables,* $\mathcal{P} = \cup_{i=1}^N \mathcal{P}_i$ *is the set of types,* $R = \cup_{i=1}^N R_i$ *is the function that determines the type of the variables, and* $\mathcal{E} = \{O_i = f_i(a_i)\}_{i=1}^N$ *is the set of structural equations where* a_i *denotes the arguments of* f_i*.*

Example 8. The corresponding SCM of the system in Example 2 is identical to the SCM presented in Example 6: $SCM(\mathcal{T}) = \mathcal{M}$.

Another important notion for defining and quantifying causal effects is that of an *intervention.* An intervention refers to deliberately changing the value of a variable to observe its subsequent effects on the other variables. By conducting interventions, we gain insights into the causal relationships and assess the impact of specific variables.

Definition 8. *If* $\mathcal{M} = (\mathcal{U}, \mathcal{V}, \mathcal{P}, R, \mathcal{E})$ *is an SCM,* $X \in \mathcal{V}$*, and* $x \in R(X)$*, then the* intervention model $\mathcal{M}[X = x]$ *is the SCM* $(\mathcal{U}, \mathcal{V}, \mathcal{P}, R, \mathcal{E}')$ *in which* $\mathcal{F} = \mathcal{F}'$ *except that the equation* $X = f_X(Y_1, Y_2, Y_3, ..., Y_n), Y_i \in (\mathcal{U} \cup \mathcal{V}) - \{X\}$*, is replaced with* $f_X(Y_1, Y_2, Y_3, ..., Y_n) = x$*.*

Example 9. If we intervene on the SCM in Example 6 by replacing the equation for B with $B = b_0$ where $b_0 \in R_B$ is a fixed inversed distance to an obstacle, the resulting intervention model will be $\mathcal{M}[B = b_0] = (\{a, b, c\}, \{A, B, C\}, \mathbb{R}, R_A \cup R_B \cup R_C, \{A = f_A(a), B = b_0, C = f_C(A, B, c)\})$. The corresponding graph is shown in Fig. 1 (right).

There is a similarity between the concept of intervention (Definition 8) and replacement (Definition 2). However, they have a significant difference as well: in a replacement, a component is replaced with an arbitrary component and function (with the same arguments, because the replacing component should be the same as the replaced one in its input and output variables). In contrast, in an intervention, the replacing function is a constant function.

Notation 2 *If* S *and* T *are specification and implementation systems, respectively, and* $\boldsymbol{u}$ *is a context, then the state of the systems under* $\boldsymbol{u}$ *is denoted by* $\boldsymbol{s}[\boldsymbol{u}]$ *and* $\boldsymbol{t}[\boldsymbol{u}]$*, or simply,* $\boldsymbol{s}$ *and* $\boldsymbol{t}$*, respectively. If* X *is a component or a set of components, the state of the replacement system where* X *is fixed to* X'*, i.e.,* $SCM(\mathcal{T}_{X \to X'})[\boldsymbol{u}]$ *is denoted by* $\boldsymbol{t} + X$*.*

3.4 Actual Causality

We conclude this section with our definition of actual causality, which is based on the notion of but-for causality. We define *a set* of implementation components as a cause of failure if replacing them with their specifications moves the state of the system out of the failure region.

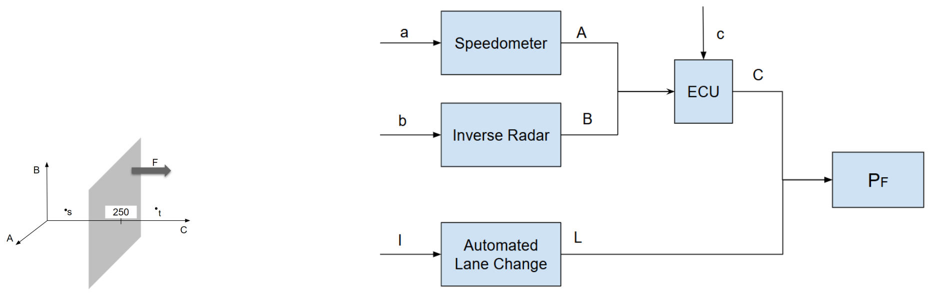

Fig. 2: Failure set in Example 10 (left) and component diagram in Example 12 (right).

Definition 9. *Suppose $\mathcal{S}$ is a specification system, $\mathcal{T}$ is an implementation of $\mathcal{S}$, $\mathcal{F}$ is a failure set, $\boldsymbol{u}$ is a context, $\mathcal{X} \subseteq \mathcal{T}$, and $\mathcal{X}'$ is the corresponding components in $\mathcal{S}$. The components in $\mathcal{X}$ are a* but-for (BF) cause *of $\mathcal{T}$ ending up in $\mathcal{F}$, denoted by $BF(\mathcal{T},\mathcal{X},\boldsymbol{u},\mathcal{F})$, iff the following conditions hold:*

- *BF1. $SCM(\mathcal{T})[\boldsymbol{u}] \in \mathcal{F}$;*
- *BF2. $SCM(\mathcal{T}_{\mathcal{X}\to\mathcal{X}'})[\boldsymbol{u}] \notin \mathcal{F}$; and*
- *BF3. $\mathcal{X}$ is minimal, in the sense that none of its subsets satisfies BF1 and BF2.*

We define the *degree of interaction* between the components of a system as the maximum size of a causal set for the system's failure. That is, every causal set has a cardinality less than or equal to the system's degree of interaction where no "higher order" interaction exists in the system.

Definition 10. *Suppose $\mathcal{S}$ is a specification system, $\mathcal{T}$ is an implementation of $\mathcal{S}$, $\mathcal{F}$ is a failure set, and $\boldsymbol{u}$ is a context. The* degree of interaction *between the components of $\mathcal{T}$ is defined as:*

$$deg_{int}(\mathcal{T},\boldsymbol{u},\mathcal{F}) = max\{|\mathcal{X}| : BF(\mathcal{T},\mathcal{X},\boldsymbol{u},\mathcal{F})\} \tag{1}$$

Example 10. Suppose $\mathcal{S}$ and $\mathcal{T}$ are a pair of specification and implementation AEB system similar to the one in Examples 1, respectively. Suppose $\mathcal{M}_\mathcal{S}$ and $\mathcal{M}_\mathcal{T}$ are the corresponding SCMs characterised by $\mathcal{U} = \{a,b,c\}$, $\mathcal{V} = \{A,B,C\}$, and the equations shown in Table 1 (left). Suppose $\boldsymbol{u} = (a,b,c) = (10,10,10)$, resulting into specification and implementation states, $\boldsymbol{s} = (A,B,C) = (10,10,110)$ and $\boldsymbol{t} = (A,B,C) = (20,20,420)$, respectively. Suppose $\mathcal{F} = \{(A,B,C) \in \mathbb{R}^3 : C \geq 250\}$, representing braking forces beyond the tolerance of the vehicle's actuators. Fig. 2 (left) depicts this situation.

Under $\boldsymbol{u}$, the states of the specification, implementation, and fixed systems are shown in Table 1 (right). It can be seen that $\{A\}$ and $\{B\}$ are a cause of $\mathcal{T}$ ending up in $\mathcal{F}$, while $\{C\}$ is not. The degree of interaction in this system is 1 for the specified context.

As stated below, if a component is not faulty, it is not a member of a cause of the failure:

Theorem 1. *Suppose $\mathcal{S}$ is a specification system, $\mathcal{T}$ is an implementation of $\mathcal{S}$, $\mathcal{F}$ is the failure set, u is a context, $X \in \mathcal{T}$, and X' is the corresponding component in $\mathcal{S}$. If $X = X'$, there is no set $\mathcal{X} \subseteq \mathcal{T}$ where $X \in \mathcal{X}$ and $BF(\mathcal{T},\mathcal{X},\boldsymbol{u},\mathcal{F})$.*

Proof by contradiction. For space considerations, we provide missing proofs in [4].

4 The Liability Framework

In this section, we explain our method of apportionment of liabilities. In line with the legal procedures for determining liabilities (Section 2), we propose a two-step procedure: (i) identification of the causal components, and (ii) apportionment of liabilities. Step (i) is done in accordance with Definition 9. Step (ii) is carried out by measuring the effect of fixing each component X. To this end, we start by identifying all causal sets that include the component X. Within each causal set, we measure the effect of X by fixing all components within the set including X, and then again, without fixing X. We calculate the difference to measure the effect. We then average these differences across all causal sets that include X, to determine the average effect of fixing X. This procedure is repeated for all other components, and the resulting values are normalised to sum to one.

To keep the liability computation tractable, we restrict our causal sets to a fixed maximum cardinality, k. This is justified from both a theoretical and an empirical perspective: theoretically, we show how the assumption about the degree of interaction can be efficiently checked, in Section 4.2. Additionally, there is empirical evidence showing that higher degrees of interaction (beyond 3) rarely occur in practice [10, 28, 32].

Definition 11. *Suppose $\mathcal{S}$ is a specification system, $\mathcal{T}$ is an implementation of $\mathcal{S}$, $\mathcal{F}$ is the failure set, $\boldsymbol{u}$ is a context, $\boldsymbol{s}$ and $\boldsymbol{t}$ are specification and implementation states under $\boldsymbol{u}$, respectively, and $k \leq |\mathcal{T}|$ is a fixed number, then the* k-leg liability *of X, denoted by ϕ^k_X, is defined as:*

$$\phi^k_X(\mathcal{T},\mathcal{S},\boldsymbol{u},\mathcal{F},k,X) = \omega \frac{1}{|\mathcal{B}_X|} \sum_{\mathcal{K} \in \mathcal{B}_X} max\big(0, depth_d(\boldsymbol{t}+\mathcal{K},\mathcal{F})\big) \tag{2}$$

Table 1: System equations (left) and states (right) in Example 10.

Component	**Spec.**	**Impl.**
A	a	$a+10$
B	b	$b+10$
C	$c+AB$	$c+AB+10$

State name	**State value**	$\boldsymbol{BF(\mathcal{T},\{X\},u,\mathcal{F})}$
$\boldsymbol{s}$	(10,10,110)	
$\boldsymbol{t}$	(20,20,420)	
$\boldsymbol{t}+A$	(10,20,220)	*True*
$\boldsymbol{t}+B$	(20,10,220)	*True*
$\boldsymbol{t}+C$	(20,20,410)	*False*

in which $\mathcal{B}_X = \{\mathcal{K} \subseteq \mathcal{T} : BF(\mathcal{T}, \mathcal{K} \cup \{X\}, \boldsymbol{u}, \mathcal{F}) \wedge |\mathcal{K}| < k\}$*,* $\boldsymbol{t} + \mathcal{K}$ *is defined as in Notation 2, and:*

$$\omega = \left[\sum_{X \in \mathcal{T}} \sum_{\mathcal{K} \in \mathcal{B}_X} \frac{1}{|\mathcal{B}_X|} max\big(0, depth_d(\boldsymbol{t} + \mathcal{K}, \mathcal{F})\big) \right]^{-1} \tag{3}$$

The normalizing coefficient ω *is chosen so that* $\sum_X \phi_X^k = 1$.

According to this definition, to measure the liability of the implementation component X, we first create the set $\mathcal{B}$ of all component sets of maximum size k, that include X and are causal for $\mathcal{T}$ ending up in $\mathcal{F}$. For each causal set, we measure the effect of X, as the difference in system robustness between fixing the entire causal set and fixing the causal set without X. We then average these differences across all such causal sets. The following example shows several desirable properties of our definition which are formalised and proved afterwards.

Example 11. Consider the AEB system shown in Table 2. In this example, A and B are interpreted as before, C represents a braking coefficient, and D is the final braking force. Assuming $\boldsymbol{u} = (10,10,10,10)$, state of the specification and implementation systems are $\boldsymbol{s} = (A,B,C,D) = (10,10,10,110)$ and $\boldsymbol{t} = (A,B,C,D) = (20,20,20,420)$, respectively. Suppose $\mathcal{F} = \{(A,B,C,D) \in \mathbb{R}^4 : D \geq 250\}$ represents the braking forces beyond the tolerance of the vehicle's actuators.

The states of the specification, implementation, and replacement systems are shown in Table 3. It is evident from the table that rows containing D are not a cause for $\mathcal{T}$ ending up in $\mathcal{F}$, because for these rows $D = 410 \geq 250$. Fixing D, even in combination with other components, does not fix the problem. Therefore, intuitively, D should not be held liable. Moreover, the error—i.e., the distance between

Table 2: System equations in Example 11.

Cm.	**Spec.**	**Impl.**
A	a	$a+10$
B	b	$b+10$
C	c	$c+8$
D^{sp}	$d+\max(AB,AC,BC)$	
$D^{im.}$	$d+\max(AB,AC,BC)+10$	

Table 3: System states in Example 11.

St.	$\boldsymbol{A}$	$\boldsymbol{B}$	$\boldsymbol{C}$	$\boldsymbol{D}$	**Value**	**St.**	$\boldsymbol{A}$	$\boldsymbol{B}$	$\boldsymbol{C}$	$\boldsymbol{D}$	**Value**
$\boldsymbol{s}$					(10,10,10,110)	$t+AB$	*	*			(10,10,20,220)
$\boldsymbol{t}$					(20,20,20,420)	$t+AC$	*		*		(10,20,10,220)
$t+A$	*				(10,20,18,380)	$t+AD$	*			*	(10,20,20,410)
$t+B$		*			(20,10,18,380)	$t+BC$		*	*		(20,10,10,220)
$t+C$			*		(20,20,10,420)	$t+BD$		*		*	(20,10,20,410)
$t+D$				*	(20,20,18,410)	$t+CD$			*	*	(20,20,10,410)

specification and implementation in this context—in C is 8, which is less than the error in A and B which is 10. As these errors symmetrically contribute to the failure as mediated by D, C should be apportioned less liability followed by A and B, and A and B should be held equally liable. The 2-leg liability follows these intuitions:

$$\begin{aligned}\phi_A^2 &= \omega\frac{1}{2}((depth_d(\boldsymbol{t}+B,\mathcal{F})-depth_d(\boldsymbol{t}+AB,\mathcal{F}))+(depth_d(\boldsymbol{t}+C,\mathcal{F})-depth_d(\boldsymbol{t}+AC,\mathcal{F})))\\ &= \omega\frac{1}{2}((130-0)+(170-0))=\omega 150=0.348\end{aligned}$$

Similarly, $\phi_B^2=0.348$, $\phi_C^2=0.302$, and $\phi_D^2=0$. In this example the degree of interaction is at least $k=2$, because any single variable's effect on the output could be masked by other variables. However, two variables together are able to affect the output.

4.1 Properties of Liability

Now, we formalise several desirable properties of k-leg liability, which serve as an axiomatic justification of our definition.

Consistency Consistency states that k-leg liability remains constant beyond the system's degree of interaction (see Definition 10):

Theorem 2. *Suppose $\mathcal{S}$ is a specification system, $\mathcal{T}$ is an implementation of $\mathcal{S}$, $\mathcal{F}$ is the failure set, and $\boldsymbol{u}$ is a context. If $k=deg_{int}(\mathcal{T},\boldsymbol{u},\mathcal{F})$, then for all component $X\in\mathcal{T}$, $\phi_X^k=\phi_X^{k+1}$.*

This justifies the relevance of a fixed k in k-leg liability. In the next section, we show that if the system can be partitioned into non-interacting subsets, the degree of interaction is bounded accordingly. We also prove sufficient conditions for such decomposability.

Dummy component This property posits that components not causal for the failure receive no liability:

Theorem 3. *Suppose $\mathcal{S}$ is a specification system, $\mathcal{T}$ is an implementation of $\mathcal{S}$, $\mathcal{F}$ is the failure set, u is a context, and $X\in\mathcal{T}$ is a component. If no subset of $\mathcal{T}$ that includes X is a BF cause of $\mathcal{T}$ ending up in $\mathcal{F}$, then $\phi_X^k=0$ for $k=1,2,3,\ldots,|\mathcal{T}|$.*

In particular, non-faulty components receive no liability, because, by Theorem 1, if a component is not faulty, it cannot be a member of a but-for cause of the failure. Therefore, its k-leg liability is zero.

Polynomial Complexity

Proposition 1. *Suppose $\mathcal{S}$ is a specification system, $\mathcal{T}$ is an implementation of $\mathcal{S}$, $\mathcal{F}$ is the failure set, and $\boldsymbol{u}$ is a context. Calculating k-leg liability of each component $X\in\mathcal{T}$ is $O(2^k|\mathcal{T}|^{k+1})$. As k is a fixed number, it is $O(|\mathcal{T}|^{k+1})$.*

4.2 Causal Non-Interaction

In this section, we examine conditions for bounding the parameter k in k-leg liability. For this purpose, we start with a characterisation of the notion of *non-interaction.*

Definition 12. *Suppose $\mathcal{M} = (\mathcal{U}, \mathcal{V}, \mathcal{P}, R, \mathcal{E})$ is an SCM; $A, B,$ and $H \in \mathcal{V}$; $h \in R(H)$; $\boldsymbol{u}$ is a context; and $\boldsymbol{v}$ is the corresponding state. Two variables A and B are* causally non-interacting *wrt the event $H = h$ iff there is no set $\mathcal{C} \subseteq \mathcal{V}$ for which $C = \boldsymbol{v}$ is a cause of $H = h$ and $\{A, B\} \subseteq \mathcal{C}$. Two sets of variables $\mathcal{A} \subseteq V$ and $\mathcal{B} \subseteq V$ are* causally non-interacting *iff for all $A \in \mathcal{A}$ and $B \in \mathcal{B}$, A and B are causally non-interacting. These definitions are generalised to more than two variables and sets in a straightforward manner.*

The following theorem shows a sufficient condition for causal non-interaction:

Theorem 4. *Suppose $\mathcal{M} = (\mathcal{U}, \mathcal{V}, \mathcal{P}, R, \mathcal{E})$ is an SCM. If P is a failure formula in the form $P(\mathcal{A}) = \bigwedge_i^n f_i(\mathcal{A}_i)$, where $\mathcal{A}_i \subseteq \mathcal{A}$, for $i = 1, \ldots, n$, and A_is are disjoint, then under any context $\boldsymbol{u}$ for which P, $\mathcal{A}_i$s are causally non-interacting wrt P.*

This condition mirrors a common way the output of different subsystems contribute to a failure: when both subsystems fail [35]. Now, if we know that system variables are causally non-interacting, we can use the following corollary to bound k in k-leg liability.

Corollary 1. *Suppose $\mathcal{M} = (\mathcal{U}, \mathcal{V}, \mathcal{P}, R, \mathcal{E})$ is an SCM and P is a failure formula. If $\mathcal{V}$ is partitioned into sets $\mathcal{A}_1, \ldots, \mathcal{A}_n$ that are causally non-interacting wrt P under a context $\boldsymbol{u}$, then* $\deg_{int}(\mathcal{M}, \boldsymbol{u}, \mathcal{F}) \leq \max\{|\mathcal{A}_i|\}$.

The utility of these results is illustrated in the following example.

Example 12. Consider the AV, shown in Fig. 2 (right), that includes two subsystems: one is an AEB system described in Example 10, and the other is an Automated Lane Change (ALC) subsystem consisting of three more components, that is $|ALC| = 3$. If an obstacle is detected, the vehicle will crash if neither the AEB is able to brake nor the ALC is able to change lanes to avoid the obstacle. In this case, P_F is a conjunction of the failure condition of the AEB and the failure condition of the ALC. Therefore, according to Theorem 4, AEB and ALC are causally non-interacting with respect to P_F. By Corollary 1, the maximum k of k-leg liability is given by $\max(|AEB|, |ALC|) = 3$. Thus, we need to search for causal sets of size at most 3, not 6 (the total number of components). Furthermore, this search is conducted only within each separate subsystem, without considering potential causal sets that have elements from both subsystems.

5 Extension to Interval Models

Our liability framework can be extended to handle various real-world complexities beyond the basic deterministic point-value models. In this section, we present one key extension in detail. In many real-world systems, specifications define acceptable ranges rather than exact values. For instance, a temperature sensor might be specified to operate within ś2řC of the true temperature. We extend our framework to handle such interval specifications.

Definition 13. *An* interval component *is a component $X=(\mathcal{I},O,\mathcal{P},R,f)$ where the specification function $f'_X:\prod_{Y\in\mathcal{I}}R(Y)\to\mathcal{P}([l,u])$ returns an interval $[l,u]\subseteq\mathbb{R}$ rather than a point value. That is, for inputs $(y_1,...,y_n)$ where $y_i\in R(Y_i)$ for each $Y_i\in\mathcal{I}$, we have $f'_X(y_1,\dots,y_n)=[l(y_1,\dots,y_n),u(y_1,\dots,y_n)]$. The implementation function f_X returns a point value that may or may not lie within this interval.*

For interval models, we need to redefine the notion of fixing a component:

Definition 14. *When fixing a component X to its interval specification X', the resulting system projects the implementation value onto the specification interval. For a system $\mathcal{T}$ with interval specification $\mathcal{S}$ and context $\boldsymbol{u}$, the state of the fixed system $SCM(\mathcal{T}_{X\to X'})[\boldsymbol{u}]$ has the value of X computed as:*

$$X=\begin{cases}l(\boldsymbol{y}) & \text{if } f_X(\boldsymbol{y})<l(\boldsymbol{y})\\ f_X(\boldsymbol{y}) & \text{if } f_X(\boldsymbol{y})\in[l(\boldsymbol{y}),u(\boldsymbol{y})]\\ u(\boldsymbol{y}) & \text{if } f_X(\boldsymbol{y})>u(\boldsymbol{y})\end{cases}$$

where $\boldsymbol{y}=(y_1,...,y_n)$ are the values of X's input variables in state $SCM(\mathcal{T})[\boldsymbol{u}]$, and $[l(\boldsymbol{y}),u(\boldsymbol{y})]=f'_X(\boldsymbol{y})$ is the specification interval.

The next example illustrates the interval extension of our liability framework.

Example 13. Consider an AEB system where:

- Speedometer spec: $f'_A(a)=[a-1,a+1]$ (ś1 m/s tolerance)
- Speedometer impl: $f_A(a)=a+5$ (5 m/s bias error)
- Radar spec: $f'_B(b)=[b-0.5,b+0.5]$ (ś0.5 1/m tolerance)
- Radar impl: $f_B(b)=b+2$ (2 1/m bias error)
- ECU spec: $f'_C(A,B,c)=[AB+c-1,AB+c+1]$ (ś1 unit tolerance)
- ECU impl: $f_C(A,B,c)=AB+c+10$ (10 unit bias error)

Given context $\boldsymbol{u}=(14,14,12)$ and failure set $\mathcal{F}=\{(A,B,C):C\geq 250\}$, the implementation state is $\boldsymbol{t}=(19,16,326)$ where $A=f_A(14)=19$, $B=f_B(14)=16$, $C=f_C(19,16,12)=326$. Since $C=326\geq 250$, the system is in failure.

To calculate 2-leg liability, we examine how fixing components affects the depth in failure:

- $\boldsymbol{t}+A$: Projects $A=19$ onto $[13,15]$ yielding $A=15$, state $(15,16,262)$, $depth=12$
- $\boldsymbol{t}+B$: Projects $B=16$ onto $[13.5,14.5]$ yielding $B=14.5$, state $(19,14.5,297.5)$, $depth=46.5$
- $\boldsymbol{t}+C$: Projects $C=324$ onto $[315,317]$ yielding $C=317$, state $(19,16,317)$, $depth=67$
- $\boldsymbol{t}+\{A,B\}$: $A=15$ (projected), $B=14.5$ (projected), $C=f_C(15,14.5,12)=15\cdot 14.5+12+10=239.5$, state $(15,14.5,239.5)$, $depth=0$ (since $238.5<250$)

– $\boldsymbol{t}+\{A,C\}$: $A=15$ (projected), $C=f_C(15,16,12)=15\cdot16+10+12=261$, projection of C onto $[251,253]$ since $f'_C(15,16,12)=[15\cdot16+12-1,15\cdot16+12+1]=[251,253]$, yielding $C=253$, state (15,16,253), $depth=3$
– $\boldsymbol{t}+\{B,C\}$: $B=14.5$ (projected), $C=f_C(19,14.5,12)=19\cdot14.5+12+10=297.5$, projection of C onto $[286.5,288.75]$, yielding $C=288.5$, state (19,14.5,288.5), $depth=38.5$

The only causal set is $\{A,B\}$ since fixing both A and B brings the system out of failure. The 2-leg liability calculation proceeds as follows:

– For component A: $\phi^2_A=\omega\cdot\frac{1}{1}\cdot\max(0,depth_d(\boldsymbol{t}+B,\mathcal{F})-depth_d(\boldsymbol{t}+\{A,B\},\mathcal{F}))=\omega\cdot(46.5-0)=46.5\omega$
– Similarly, for component B: $\phi^2_B=12\omega$
– For components C: $\phi^2_C=0$ (not part of any causal set)

With normalization $\omega=(46.5+12)^{-1}=1/58.5$, the final liabilities are: $\phi^2_A=0.795$, $\phi^2_B=0.205$, $\phi^2_C=0$, which is expected because of the symmetric roles of A and B in causing failure, while A having larger deviation from its specification results in higher liability.

This demonstrates how interval specifications naturally handle tolerance-based engineering requirements. The following theorem confirms that the key properties of our framework are preserved under this extension.

Theorem 5. *For systems with interval specifications, the k-leg liability framework preserves all key properties of **Consistency** (Theorem 2), **Dummy component** (Theorem 3), and **Polynomial complexity** (Proposition 1).*

Proof. The projection operation during fixing, only changes the values of fixed components, not the causal dependencies. Therefore, the proofs of the original theorems apply directly, as they rely on the structure of causal sets and distance calculations, which remain unaffected by interval specifications.

6 Comparison with Harm

Beckers, Chockler, and Halpern (BCH) proposed a qualitative and quantitative definition of harm [5, 6]. Any definition of harm involves an event *causing* a loss for an agent. The causal mechanism is characterised by an SCM that includes a designated outcome variable. This outcome variable is directly linked to the utility of the agent. Quantitative harm is defined as the following:

Definition 15. *If $\mathcal{M}=(\mathcal{U},\mathcal{V},\mathcal{P},R,\mathcal{E})$ is an SCM, $O\in\mathcal{V}$ is a certain endogenous variable called the* outcome variable, *$ut: R(O)\to[0,1]$ is a function called the utility function, and $d\in[0,1]$ is a number called* default utility*, then $\mathcal{C}=(\mathcal{M},ut,d)$ is called causal utility model.*

Definition 16. *If $\mathcal{M} = (\mathcal{U}, \mathcal{V}, \mathcal{P}, R, \mathcal{E})$ is an SCM and $\mathcal{C} = (\mathcal{M}, ut, d)$ is a causal utility model, and $X = x$ instead of $X = x'$ causes $O = o$ rather than $O = o'$ in $(\mathcal{C}, \boldsymbol{u})$, then the (quantitative) harm to agent ag relative to $(X = x, X = x', O = o, O = o')$ is defined as $QH(\mathcal{C}, \boldsymbol{u}, X = x, x', o, o') = max(0, min(d, ut(o')) - ut(o))$. The quantitative harm to agent ag caused by $X = x$ in $(\mathcal{C}, \boldsymbol{u})$ is defined as $QH(\mathcal{C}, \boldsymbol{u}, X = x) = max_{x', o'} QH(\mathcal{C}, \boldsymbol{u}, X = x, x', o, o')$ if there is some x' and o' such that $X = x$ instead of $X = x'$ causes $O = o$ instead of $O = o'$; if there is no such x' and o', then the quantitative harm is taken to be 0.*

Both harm and the 1-leg liability (i.e., k-leg liability with $k = 1$) measure the difference between the utility of an actual and an expected outcome. In 1-leg liability, the utility of a state is the depth of the state within the set of failure states $\mathcal{F}$. The following theorem formalises this relationship:

Theorem 6. *Suppose $\mathcal{S}$ is a specification system, $\mathcal{T}$ is an implementation of $\mathcal{S}$, $\mathcal{M} = SCM(\mathcal{S})$, $\mathcal{N} = SCM(\mathcal{T})$, $\mathcal{F}$ is the failure set, u is a context, $X \in \mathcal{T}$, X' is the corresponding component in $\mathcal{S}$, and $\{X\}$ is a BF cause of $\mathcal{T}$ ending up in $\mathcal{F}$. Suppose $\mathcal{W} = \mathcal{T}_{X \to X'}$ and $\mathcal{P} = SCM(\mathcal{W})$. Define $x = \mathcal{N}[\boldsymbol{u}][X']$ and $x' = \mathcal{P}[\boldsymbol{u}][X]$. Also define the utility function $ut(s) = depth_d(s, \mathcal{F})$, loosening its normalisation condition. Define $\mathcal{M}'$ as an SCM identical with $\mathcal{M}$ with the additional (vector) endogenous variable $\boldsymbol{O} = \mathcal{V}(\mathcal{M})$ where $\mathcal{V}(\mathcal{M})$ denotes the set of endogenous variables of $\mathcal{M}$. Then we have:*

$$\phi^1_X = \alpha QH(\mathcal{M}', ut, \infty, X = x, x', \mathcal{P}[\boldsymbol{u}], \mathcal{N}[\boldsymbol{u}])$$

where α is constant.

7 Implementation and Empirical Evaluation

In this section, we aim to assess the effectiveness and efficiency of our proposed k-leg approach for liability apportionment compared to the widely accepted Shapley value method. We use the Shapley values as the baseline because it is considered the standard apportionment approach [7, 12, 14, 20]. Our research questions are:

- **RQ1 (Accuracy)**. Is there a meaningful difference between liabilities calculated via the k-leg approach and Shapley values?
- **RQ2 (Efficiency)**. Is there a meaningful difference between computational time of the k-leg approach and Shapley values?

7.1 Methodology

In our experiments, each subject consists of a random linear system of M components representing the specification system, re-randomisation of the specification to act as the implementation, a random failure quantifier-free first-order formula, and a context: (i) To randomly generate a specification system, we first generate a random DAG; for each DAG node (corresponding to a system components) the output is defined as a random linear function of its direct children. The linear function coefficients are real numbers uniformly sampled from $[-100,+100)$. (ii) The first-order failure formula is of the form $\bigwedge(x_i \leq | \geq a_i)$ where at most $\lfloor|\mathcal{V}|/3\rfloor$ of endogenous variables are randomly selected for x_is and a_is are real numbers uniformly sampled from $[-90,+90)$. (ii) Generation of a suitable experimental context is intricate, because we are interested in the contexts leading to a failure in implementation, but not a failure in the specification. To this end, we use Z3 SMT solver [23] to find a suitable context. If no context satisfies the constraints, we discard the generated systems and the failure and try again.

To answer RQ1, we compute the liability of the components in a subject with both Shapley and k-leg methods ($k = 1,2,3$). This gives us two vectors $\boldsymbol{s}$ and $\boldsymbol{l}$, each summing to one. We get the sum of the absolute value of the difference of the vectors (i.e., $\sum_i |s_i - l_i|$), which is a number between 0 and 2. We repeat this process $N = 1000$ times for each pair of $M = 5-14$ and $k = 1,2,3$. Similarly, for RQ2, we find the difference between Shapley and k-leg computation times (in seconds), for $N = 1000$ random subjects, and repeat this for each pair of $M = 5-14$ and $k = 1,2,3$. We developed a Python package for quantitative liability analysis within causal models which will be made available after acceptance.

7.2 Results

RQ1. For each (M,k) pair, we generated $N = 1000$ data points representing the difference between Shapley and k-leg values. Figure 3 (left) shows these differences as box plots. The 1st, 2nd, and 3rd quartiles are always 0, indicating the methods typically agree. To assess significance, we computed the percentage of cases where the L_1 norm difference exceeds 0.4 (maximum possible difference: 2). The resulting p-values are reported in Table 4. For $k = 3$, all p-values are below 0.1, showing no significant difference for systems with up to 9 components and supporting the k-leg method's efficacy.

RQ2. For each pair of (M,k) we have $N = 1000$ data points that represent the time difference between Shapley and k-leg values. The results are plotted on Fig. 3 (right). The exponential time difference is in line with the complexity results stated earlier.

Table 4: p-values for the difference between k-leg and Shapley liabilities for $M = 5-14$ and $k = 1,2,3$.

M	**k=1**	**k=2**	**k=3**
5	**0.021**	**0.004**	**0.000**
6	**0.028**	**0.018**	**0.000**
7	**0.027**	**0.034**	**0.002**
8	**0.033**	0.121	**0.039**
9	**0.028**	0.101	**0.041**

M	**k=1**	**k=2**	**k=3**
10	**0.007**	0.183	0.062
11	**0.037**	0.094	0.050
12	**0.036**	0.093	0.061
13	**0.029**	0.085	0.073
14	**0.040**	0.079	0.071

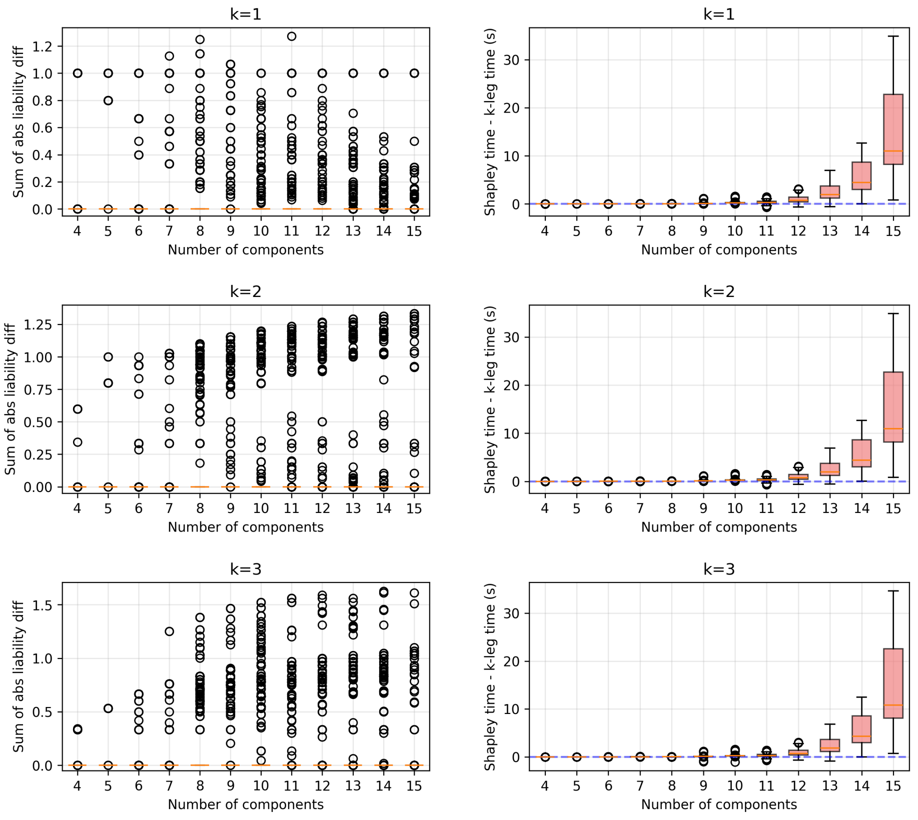

Fig. 3: Comparison of Shapley and k-leg methods in terms of liability difference (left) and computational time (right).

7.3 Discussion

(RQ1) The experimental results indicate that the k-leg approach provides liability apportionments that are closely aligned with the Shapley values, particularly for smaller k values such as $k = 3$, with p-values < 0.05, suggesting no significant difference between the methods for systems with up to nine components. This implies that the k-leg method effectively approximates the Shapley value, maintaining fairness while simplifying computations. **(RQ2)** Additionally, the computational efficiency of the k-leg approach is evident from the significant reduction in execution time compared to the Shapley method, especially as the number of components M increases (see Fig. 3 (right)). Therefore, the k-leg method offers a practical and efficient alternative for liability assessment in complex systems without compromising accuracy.

8 Conclusion and Future Work

We proposed a formal framework for automatically apportioning liabilities, illustrated its desirable features through examples, and established its formal properties. We introduced the concept of causal non-interaction and identified sufficient conditions for its validity. Furthermore, we illustrate that our framework handles real-world complexities such as interval specifications.

Future research directions include applying the k-leg liability concept directly to definitions of actual causality, extending the framework to incorporate manufacturer design flaws, and formally connecting k-leg liability with Shapley-based apportionment methods, including fairness axioms. Additionally, exploring formal semantic definitions of component interactions and identifying syntactic conditions that entail them present promising avenues for further research.

Acknowledgments

Mohammad Reza Mousavi has been partially supported by the UKRI Trustworthy Autonomous Systems Node in Verifiability, Grant Award Reference EP/V026801/2, EPSRC project on Verified Simulation for Large Quantum Systems (VSL-Q), grant reference EP/Y005244/1 and the EPSRC project on Robust and Reliable Quantum Computing (RoaRQ), Investigation 009 Model-based monitoring and calibration of quantum computations (ModeMCQ), grant reference EP/W032635/1, ITEA/InnovateUK projects GENIUS, grant reference 600642, and ITEA/InnovateUK GreenCode, grant reference 600643.

References

1. Reducing driver error accidents – orsa.org.uk, `https://www.orsa.org.uk/reducing-occupational-road-risk/reducing-driver-error-accidents/`
2. Abraham, K.S., Rabin, R.L.: Automated vehicles and manufacturer responsibility for accidents: a new legal regime for a new era. Virginia Law Review **105**(1), 127–171 (2018)
3. Aleksandrowicz, G., Chockler, H., Halpern, J.Y., Ivrii, A.: The computational complexity of structure-based causality. Journal of Artificial Intelligence Research **58**, 431–451 (2017)
4. Aryan, K., Chockler, H., Mousavi, M.R.: Artifact for the paper "causal liability in autonomous systems" (Jan 2026). `https://doi.org/10.5281/zenodo.18393980`
5. Beckers, S., Chockler, H., Halpern, J.Y.: A causal analysis of harm. In: Advances in Neural Information Processing Systems. vol. 35 (2022)
6. Beckers, S., Chockler, H., Halpern, J.Y.: Quantifying harm. In: Proceedings of Thirty-Second International Joint Conference on Artificial Intelligence (2023)
7. Chakravarty, S.R., Mitra, M., Sarkar, P.: A course on cooperative game theory. Cambridge University Press (2015)
8. Chockler, H., Halpern, J.Y.: Responsibility and blame: a structural-model approach. Journal of Artificial Intelligence Research **22**, 93–115 (2004)
9. Chockler, H., Kroening, D., Sun, Y.: Explanations for occluded images. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021)

10. Damasceno, C.D.N., Mousavi, M.R., Simao, A.d.S.: Learning by sampling: learning behavioral family models from software product lines. Empirical Software Engineering **26**(1), 4 (2021)
11. Davey, J.: By insurers, for insurers: the UK's liability regime for autonomous vehicles. Journal of Tort Law **13**(2), 163–188 (2020)
12. Dehez, P., Ferey, S.: How to share joint liability: a cooperative game approach. Mathematical Social Sciences **66**(1), 44–50 (2013)
13. Fainekos, G.E.: Robustness of Temporal Logic Specifications. Ph.D. thesis, University of Pennsylvania (2008)
14. Ferey, S., Dehez, P.: Multiple causation, apportionment, and the shapley value. The Journal of Legal Studies **45**(1), 143–171 (2016)
15. Gallow, J.D.: The metaphysics of causation. In: Zalta, E.N., Nodelman, U. (eds.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, fall 2022 edn. (2022)
16. Halpern, J.Y.: A modification of the Halpern-Pearl definition of causality. In: Proceedings of the 24th International Conference on Artificial Intelligence. pp. 3022–3033. IJCAI'15, AAAI Press (Jul 2015)
17. Halpern, J.Y.: Actual causality. MIT Press (2016)
18. Halpern, J.Y., Pearl, J.: Causes and explanations: a structural-model approach. part i: causes. The British Journal for the Philosophy of Science **56**(4), 843–887 (2005)
19. Institute, A.L.: Restatement of the law third, torts: apportionment of liability, as adopted and promulgated. American Law Institute Publishers (2000)
20. Kim, J.Y., Lee, S.: Joint liability and stochastic shapley value. International Review of Law and Economics **60** (2019)
21. Moore, M.: Causation in the law. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, winter 2019 edn. (2019)
22. Morgan, P.: Tort and Autonomous Vehicle Accidents - The Automated and Electric Vehicles Act 2018 and the Insurance Solution? In: Soyer, B., Gurses, O. (eds.) Insurability of Emerging Risks: Law, Theory and Practice, p. Chapter 10. Bloomsbury Publishing, Hart Publishing, 1st edn. (2025)
23. de Moura, L., Bj∅rner, N.: Z3: an efficient smt solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. Springer (2008)
24. Métayer, D., Maarek, M., Mazza, E., Potet, M.L., Frenot, S., Viet Triem Tong, V., Craipeau, N., Hardouin, R., Alleaune, C., Benabou, V.L., Beras, D., Bidan, C., Goessler, G., Clainche, J., Mé, L., Steer, S.: Liability in software engineering overview of the LISE approach and illustration on a case study. ACM/IEEE 32nd International Conf. on Software Engineering (ICSE 2010) **1** (May 2010)
25. Norton, D.F., Norton, M.J.: David Hume: A Treatise of Human Nature: Volume 1: Texts. OUP Oxford (Jan 2011)
26. Pearl, J.: Causality: models, reasoning, and inference. Cambridge University Press (Sep 2009)
27. Pearl, J., Glymour, M., Jewell, N.P.: Causal inference in statistics: a primer. John Wiley & Sons (2016)
28. Petke, J., Yoo, S., Cohen, M.B., Harman, M.: Efficiency and early fault detection with lower and higher strength combinatorial interaction testing. In: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. pp. 26–36. Association for Computing Machinery (2013)
29. Rausand, M., Barros, A., Hoyland, A.: System Reliability Theory: Models, Statistical Methods, and Applications. John Wiley & Sons (2020)

30. Sarda Gou, M., Lakatos, G., Holthaus, P., Robins, B., Moros, S., Jai Wood, L., Da Silva Araujo, H., deGraft Hanson, C., Mousavi, M., Amirabdollahian, F.: Kaspar explains: the effect of causal explanations on visual perspective taking skills in children with autism spectrum disorder. In: Proceedings of the 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN 2023). IEEE (2023)
31. Signoret, J.P., Leroy, A.: Reliability assessment of safety and production systems: analysis, modelling, calculations and case studies. Springer series in reliability engineering, Springer (2021)
32. Steffens, M., Oster, S., Lochau, M., Fogdal, T.: Industrial evaluation of pairwise SPL testing with MoSo-PoLiTe. In: Proceedings of the 6th International Workshop on Variability Modeling of Software-Intensive Systems. pp. 55–62. VaMoS '12, Association for Computing Machinery, New York, NY, USA (Jan 2012)
33. Wetzel, L.: Types and tokens. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, fall 2018 edn. (2018)
34. Willard, S.: General topology. Courier Corporation (2012)
35. Xiang, J., Ogata, K.: Formal Fault Tree Analysis of State Transition Systems. In: Fifth International Conference on Quality Software (QSIC'05). IEEE Computer Society (2005)