ECAI 2020
G.D. Giacomo et al. (Eds.)
© 2020 The authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA200472

Actual Causality Canvas: A General Framework for Explanation-Based Socio-Technical Constructs

Amjad Ibrahim and **Tobias Klesel** and **Ehsan Zibaei** and **Severin Kacianka** and **Alexander Pretschner**[1]

Abstract. The rapid deployment of digital systems into all aspects of daily life requires embedding social constructs into the digital world. Because of the complexity of these systems, there is a need for technical support to understand their actions. Social concepts, such as explainability, accountability, and responsibility rely on a notion of actual causality. Encapsulated in the Halpern and Pearl's (HP) definition, actual causality conveniently integrates into the socio-technical world if operationalized in concrete applications. To the best of our knowledge, theories of actual causality such as the HP definition are either applied in correspondence with domain-specific concepts (e.g., a lineage of a database query) or demonstrated using straightforward philosophical examples. On the other hand, there is a lack of explicit automated actual causality theories and operationalizations for helping understand the actions of systems. Therefore, this paper proposes a unifying framework and an interactive platform (Actual Causality Canvas) to address the problem of operationalizing actual causality for different domains and purposes. We apply this framework in such areas as aircraft accidents, unmanned aerial vehicles, and artificial intelligence (AI) systems for purposes of forensic investigation, fault diagnosis, and explainable AI. We show that with minimal effort, using our general-purpose interactive platform, actual causality reasoning can be integrated into these domains.

1 Introduction

Causality has produced centuries of interdisciplinary theorization. The first documented theories can be traced to ancient philosophers such as Plato (Phaedo, 360 BC) and Aristotle (Posterior Analytics), reaching to current theories by computer scientists such as Pearl [36] and Halpern [12]. Focusing on a goal-driven categorization of causality, we distinguish two notions: *type* (general) causality and *actual* (token) causality [13, 35]. Type causality is concerned with general causal relations and aims to forecast the *effect* of a cause [29]. Reasoning with this notion aids in expanding predictive fields such as medicine [22] and economics. In contrast, actual causality theories focus on explaining an observed event, that is, inferring *causes from effects* [12]. Thus, such theories are useful in assigning blame, explaining, or preventing similar events in the future. As such, actual causality directly benefits debugging software, hardware [6], database queries [31], and system models [6, 3]. According to its retrospective attribution, actual causality lies at the heart of explanation-based social constructs such as accountability [8], responsibility [5], and explainability [32].

Embedding such constructs into modern digital systems is indispensable [21, 32, 9, 8]. For instance, since Cyber-Physical Systems such as drones or airplanes may harm people when they "fail", building such systems to be *accountable* is a necessity [18, 26, 20, 37]. *Accountability* in this context refers to developing a system's (forensic) capabilities in holding misbehaving parties responsible for violations. In the case of a drone crash, it is imperative to find and address the root cause to prevent future mishaps; in aircraft accidents, accountability is part of the judicial process to assign liability and responsibility. We call a system *accountable* if it can help answer questions regarding the actual cause of some (usually undesired) event. The accountability of a system requires at least two properties: it must provide *evidence*, usually in the form of logs, which helps understand the causes of the undesired event. Moreover, some mechanism must be in place that can argue *causality*. Also, systems with AI components (e.g., machine learning applications) constitute another relevant application of explanation-based constructs. Since such systems are tasked with making daily decisions or predictions for humans, interest in explaining their results is growing [32, 33]. Similar to accountability, explainability is inherently causal. Hence, this paper aims to connect technical domains such as the abovementioned with explanation-based social concepts via enabling automated actual causality reasoning.

As a technique of knowledge representation and reasoning, actual causality is well formalized as a result of Halpern and Pearl's (HP) definition of actual causality [11], which is efficiently checked [19] and thus suitable for the socio-technical world. However, to the best of our knowledge, explicit actual causality theories and operationalizations have not been utilized, in an automated fashion, to enable socio-technical purposes such as accountability and explanation. Although the HP definition describes a cause in a way that matches human thinking, it was either applied in relation to domain-specific technical artifacts (e.g., a lineage of a query [30], counter-example of a model checker [28]) or demonstrated using simple philosophical examples [13]. In this paper, we aim to answer the question *How can actual causality theories be operationalized?* This would entail establishing a general framework and automating the parts that can be automated. Actual causality reasoning, while used across different disciplines, currently lacks a clear methodology and especially tools to build, transfer, and reason over causal models. To this end, we propose a unifying methodology to enable automated causality reasoning and demonstrate its utilization for forensic investigation, fault diagnosis, and explainable AI (xAI).

We argue that a semi-automated framework of actual causality serves as a starting point to achieve the ambitious goal of enabling complex interdisciplinary concepts such as accountability. Especially for operationalization, a unifying framework diminishes the barrier to embedding causality reasoning in new domains because it al-

[1] Technical University of Munich, Germany, email: lastname@in.tum.de

lows *reuse*. As we shall see in this paper, the framework consists of tasks that interweave social and technical boundaries. Some of these tasks reuse domain-specific methodologies and knowledge sources. Also, the framework automates solely technical tasks so that they are *reused* among different domains. Consequently, attention shifts to how such technical tasks serve goals, such as enabling accountability. To this end, this paper contributes the following **a)** a generalized unifying framework to operationalize causal reasoning; **b)** a general-purpose, open-source, interactive platform called the *Actual Causality Canvas (short: the Canvas)* [2], which automates parts of the actual causality framework; and **c)** three use cases that instantiate the framework for accountability, fault diagnosis, and xAI purposes.

2 Preliminaries

The HP definition of (actual) causality is a widely cited approach based on counterfactual reasoning and models of structural equations [35, 11]. The counterfactual view of causality was proposed by Hume (1748) [16]. This view can be understood by a basic example: If B would not have happened, had A not occurred, then we call A a cause for B. Obviously, reasoning about causality in this straightforward way is not always sufficient. Therefore, HP tried to formalize this concept and provide a theoretical notion of causality.

HP provides a method to causally reason over knowledge represented in a *causal model*. This model uses *variables* that can take on different values to describe the world. The way some of these variables causally influence each other, that is, affect their values, is defined by so-called *structural equations* [35]. The variables are split into *exogenous* and *endogenous* ones. Exogenous variables, governed by factors external to the model, represent the environment, and hence, they cannot be part of a cause. Meanwhile, endogenous variables represent the factors we consider as possible causes; their values are determined by the structural equations of other endogenous and exogenous variables. To keep the paper accessible to a broader audience, we present an informal definition of a binary causal model. In this study, we limit our focus only on acyclic models representing binary variables.

Definition 1 ***Binary Causal Model*** *[11]*
A binary causal model M is a tuple $M = (\mathcal{U}, \mathcal{V}, \mathcal{F})$, where
- *$\mathcal{U}, \mathcal{V}$ are sets of exogenous and endogenous binary variables,*
- *$\mathcal{F}$ associates with each variable $X \in \mathcal{V}$ a function F_X that determines the binary value of X given the values of all other variables.*

The values of the exogenous variables in $\mathcal{U}$ are defined by a *context* $\vec{u}$, which is an assignment of values based on a particular event. Given a causal model and a context $(M, \vec{u})$, HP defines an actual cause (a set of events and their values denoted as $\vec{X} = \vec{x}$) of an event (or a combination of such) φ using three conditions. We show the informal interpretation of these conditions in Definition 2.

Definition 2 (Actual Cause (Informal)). *$\vec{X} = \vec{x}$ is an actual cause of φ in $(M, \vec{u})$ if the following three conditions hold:*
AC1. *Both $\vec{X} = \vec{x}$ and φ need actually to happen.*
AC2. *Changing the original values of $\vec{X}$ to a different setting $\vec{x}'$ while "fixing" some of the remaining variables at their original value (in a contingency set $\vec{W}$), φ must not occur anymore.*
AC3. *$\vec{X}$ is minimal; no subset of $\vec{X}$ satisfies AC1 and AC2.*

With AC1, it is ensured that the events $\vec{X} = \vec{x}$ are only considered as a cause of φ if both occurred. Because it is convenient and common to sort out irrelevant causes, AC3 is a minimality condition. The core of Definition 2 lies in AC2, which resembles counterfactual reasoning; it holds if there exists a setting $\vec{x}'$ of the variables in $\vec{X}$ different from the original setting $\vec{x}$ and another set of variables $\vec{W}$, which we use to keep variables at their original value (when the effect happened), such that φ does not occur anymore. This matches the counterfactual definition of causality: if $\vec{X} = \vec{x}$ does not happen anymore, i.e., variables $\vec{X}$ take on different values $\vec{x}'$, and thus φ does not occur anymore as well, we can call $\vec{X} = \vec{x}$ a cause (provided that AC1 and AC3 also hold). The role of the contingency set $\vec{W}$ becomes more clear when considering the following example (Lewis [29]): "Suzy and Billy both throw a rock on a bottle which shatters if one of them hits. We know that Suzy's rock hits the bottle slightly earlier than Billy's and both are accurate throwers." Halpern models this example with these variables (excluding exogenous variables): ST for "Suzy throws," with values 1 (she throws) and 0 (she does not throw); BT for "Billy throws," with values 1/0 (he throws/ he does not throw); BH for "Billy's rock hits the (intact) bottle," with values 1/0 (it does/does not); SH for "Suzy's rock hits," with values 0 and 1; and BS for "bottle shatters," with values 1/0 (it does/does not). The equations are as follows:
- BS is 1 iff one of SH and BH is 1, i.e., $BS = SH \vee BH$,
- SH is 1 iff ST is 1, i.e., $SH = ST$,
- $BH = 1$ iff $BT = 1$ and $SH = 0$, i.e., $BH = BT \wedge \neg SH$.

Figure 1 visualizes how the variables influence each other as defined by the equations, which is referred to as a *causal graph* or a *causal network*. Each node A represents an endogenous variable; an edge from A to B means that B "depends" on A.

$$\begin{tikzcd}
ST \arrow[r] & SH \arrow[d] \arrow[dr] & \\
BT \arrow[r] & BH \arrow[r] & BS
\end{tikzcd}$$

Figure 1. Rock-throwing Example (Source: [13])

Assuming a context $\vec{u}$ that sets $ST = 1$ and $BT = 1$, we have $SH = 1$, $BH = 0$, and $BS = 1$ as the *actual evaluation* of the model. Let us consider the following causal queries: Are $ST = 1$, $BT = 1$, or both a cause for $BS = 1$? We begin by checking whether $ST = 1$ is a cause. AC1 is fulfilled, as both $ST = 1$ and $BS = 1$ actually happened. Since ST can take only two different values, the only possibility for a setting $\vec{x}'$ for ST is 0, i.e., Suzy does not throw. A trial with $\vec{W} = \emptyset$ (an empty contingency set) shows that AC2 does *not* hold: if $ST = 0$, then $SH = 0$ such that BH changes to 1, as we did not change BT, and ultimately, BS is still 1. However, the HP definition allows us to define $\vec{W} = \{BH\}$. Now AC2 holds because $BS = 0$, and so does AC3, as our cause consists of a single event. This example challenges causality definitions because it shows a case of *preemption* –a confusing situation where multiple possible causes can coincide. HP deals with such cases with the idea of a *contingency set*. Now, let us consider $BT = 1$ as a cause for $BS = 1$; we can immediately see that AC1 is fulfilled as well. Similar with ST, the only possible setting $\vec{x}'$ for BT is 0. However, this does not affect BS, and it is also not possible to find a $\vec{W}$ such that AC2 holds; that is, $BT = 1$ is *not* a cause. Also, $ST = 1 \wedge BT = 1$, i.e., the conjunction of both, is not a cause since it would not fulfill AC3: there is a subset, namely $ST = 1$, which is a cause by itself.

Responsibility is an extension of the definition by Chockler et al. [5]. It quantifies the concept as a metric to allow for a comparison of causes in the form of a degree of responsibility.

[2] https://github.com/tum-i22/causal-canvas

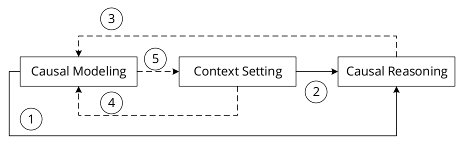


Figure 2. A Process View of the Framework

3 A Framework of Actual Causality

Having looked at the theoretical foundation of actual causality, in this section, we consider the technical and methodological aspects of the theory. Later, we see how the *Canvas* automates each aspect.

3.1 Conceptual Tasks

We identified the three main decoupled activities of actual causality: *modeling, context setting, and reasoning*. Figure 2 shows a process perspective of operationalizing these components. The solid arrows (1 and 2) are inherited from the theories themselves, i.e., we need a model and a context to reason about a cause. However, we also argue that interconnecting these components (dashed arrows 3, 4, and 5) may address some of their challenges. In the following, we focus on the key tasks and questions of each phase; this also serves as a requirement elicitation step for the *Canvas*.

Causal modeling makes our understanding of causal factors explicit. Most probably, as with any modeling process, the resulting model is incomplete, meaning "there is no right model" [12]. Halpern and Pearl themselves have shown several times the difficulties of coming up with a proper model and its considerable influence on the result of cause evaluation [14]. However, a causal model is nothing but the modeler's assumption of how factors relate and influence each other, thus necessitating the ability to edit the models incrementally. As such, we designed the *Canvas* as an interactive learning system where a modeler refines their causal model (according to Definition 1), hence *dashed arrow* 3 in Figure 2.

Since causal models are flexible (from the rock-throwing example [29] to the model checker's results [28]) and intuitive, many approaches exist for model discovery. Model discovery refers to the act of creating *causal models* from some source of knowledge. Understandably, model discovery by *transformation* is domain-specific. In a specific area, we need to identify existing models matching the properties of a causal one. For this paper, such models have to be *acyclic*, *causal*, *and expressed in propositional logic*. Optionally, they may also be *probabilistic* and *Boolean*. Then, a formal model transformation function can be plugged into the process to create the causal model. For instance, the domain of system security has about 30 different graphical *threat models* [24], many of which already express (type) cause-effect relations and can also be used to infer actual causality. Among the best-known threat models are attack trees [41] and attack graphs [42]. Usually, attack trees are constructed by security experts to assess the risk on a system. The ability to automatically generate attack graphs from other sources even eliminates the manual process [42, 17]. For *accountability* purposes, we can re-use such knowledge to reason about attacks that have already occurred. The same applies to hazard models such as fault trees and their automated generation [45]. In Section 4, we will see more sources of causal models facilitated by how the *Canvas* supports their transformation (details in Section 3.2).

Regardless of whether the model is created manually, transformed from other sources, or automatically generated, we emphasize that it should be augmented with *preemption* relations. As we have seen in Section 2, *preemption* should be expressed in the model when possible (the connection between Suzy's and Billy's hits). This reflects a discrepancy among coinciding disjunctive events in confusing situations. Preemption relations can stem from different requirements or facts. They can reflect a *temporal* order of events as was the case in the rock-throwing example; they can be *functional*; for instance, a command by a drone remote-controller takes priority over an autopilot command; they can also be *contractual*; for example, a sensor manufacturer is obliged to notify a drone admin about a patched software library. These relations are crucial to conclude a cause in some cases. However, we think that they have not been sufficiently highlighted in the literature. Hence, we believe they should be noted explicitly as part of our approach. Semantically, preemption relations are expressed with a negation dependency from the *preempting* (P) factor to the "less affecting" factor (Q), i.e., $Q := \ldots \wedge \neg P$.

To conclude, the questions to be answered as part of this task in a particular domain are the following: *Q1: Which transferable sources (models or data) of causal knowledge exist? Q2: What is the formal mapping between the source and destination syntax and semantics? Q3: How can preemption relations be identified and expressed?*

Context setting is the act of describing an event's circumstances as an assignment of values to *exogenous* variables. For example, based on the black-box recordings of aircraft that collided near Ueberlingen [43], the investigators knew that the ground air traffic controller (ATC) had alerted the first aircraft's crew of traffic but on a wrong direction. In the accident's causal model, such information would set the value of a variable like *Air traffic control correctly alerts crew* to *false*. In a digital forensic investigation of cyber-attacks, experts try to retrieve trustworthy log files from different systems to *set the context*. The logged events aid in understanding the occurrences. For example, a log statement like *.."MACHINE-ID" : "8a7","CMDLINE" : "gdb –nx –batch -ex attach.."* is interpreted as an admin with a specific ID has attached a debugger to a running process; this sets the model variable *admin attached debugger* to *true*. These examples are meant to show that *context setting* varies among domains; however, there are established methods to help in this task. With diverse sources, from recordings, eyewitness reports, to logs, we see two primary methodological approaches to *context setting*.

The first approach is considered in scenarios where a line of "trust" exists between an agent, such as a system-admin, and a system like a company, or a citizen and traffic police. In such situations, we have an intuition about typical misbehavior; for example, an admin leaks sensitive data, or a driver goes over the speed limit. Our knowledge of such patterns can be presented as causal models that guide our monitoring effort (hence dashed arrow 5 in *Figure 2*). In such cases, an approach like this (have a model, monitor it) can be deterrent. Furthermore, this way, we address the principal challenge of logging and auditing capabilities, i.e., the granularity of logging or monitoring.

Things are not that simple, however. It is not safe to assume that we always have an intuition about the typical pattern of unwanted behavior ("unknown unknowns"). In the second approach, systematic processes normally start by analyzing sources of truth and narrowing the events. Then, they structure the information so that it can be transformed into a causal model that embeds the context, hence arrow 4 in Figure 2. An example of this method is why-because-analysis (WBA) [25, 26]. This approach does not address the granularity of monitoring since it only deals with after-the-fact sources.

The two approaches are not mutually exclusive; we can leverage both for the same system. For example, known typical malicious insider (security) attacks can be monitored, and unknown external attacks are investigated. As we will see in the case of xAI, context setting can be neither and is as simple as field assignment.

Causal reasoning includes both checking and inference. Causal checking involves verifying if a hypothesized cause is an actual cause of an effect. Inference, on the other hand, means finding a cause with no hypothesis. Both notions must be available as part of actual causality operations. Checking is already beyond NP [1], and intuitively, the inference is at least as hard. The complexity results have limited the application of the actual causality theory; however, because of our recent open-source approach, efficient methods to check and infer causality are available [19].

Causal reasoning is mainly motivated by a goal of liability attribution [15], future prevention [40], or explanation [32]. Regardless of the target, causal reasoning answers a causal *query*, which consists of a context, a hypothesized cause (in the case of checking), and an effect. Since causal reasoning is automated in the *Canvas*, the crucial question is *What is the query for each goal?*

For liability attribution purposes, we are interested in hypothesized causes that include humans. For example, *is admin "Bob" the cause of stealing the document?* For such purposes, we focus on the responsibility (Section 2) of the cause in the case of multiple causes. Additionally, we tend to consider negligence or failure to do an expected job as a potential cause in such situations. For example, *s the ATC's failure to use a cell phone the cause of the collision?*

Future *prevention* requires identifying all sufficient causes regardless of actuality or minimality [29, 26] and putting countermeasures in place. To this end, a causal query would collect all causes by trying different hypotheses regardless of their responsibility.

According to a recent survey by Miller [32], humans seek contrastive *explanations*. In other words, people would not phrase their causal queries as *Why did event P happen?* but rather as *Why did P happen instead of Q?* [32]. Miller also concludes that explanations are selected and social. We think a contrastive query can be constructed by phrasing the effect φ in a way that expresses this distinction. For example, φ will be a formula like $\neg Q$.

To conclude, query formulation is a crucial part of this phase. We use the same language to formalize a causal query for different purposes. However, we adapt to the goal and include responsibility, collect all causes, or phrase the effect in contrast to reality.

3.2 The Actual Causality Canvas

Recall that we are interested in solving problems fixated around causality, such as finding out why a drone crashed or explaining a classifier result. Instantiating the three activities—model, context, and reasoning— we can solve such problems. We have seen that each phase has its own challenges and methodological decisions. However, we believe that a unified platform that enables each phase is crucial for deploying more causality socio-technical applications. Such a platform must be general enough to accommodate different practices, models, and queries. As such, we present the Actual Causality Canvas, an extensible, open-source, interactive tool that automates the abovementioned tasks and activities. It is in part a modeling tool that supports typical modeling activities and provides methods to transfer domain-specific models to causal ones. Also, the Canvas allows an analyst to perform interactive causal analysis of a particular event. Obviously, graphical editors are common in many domains; however, the Canvas's contribution lies in encapsulating the crucial tasks needed to specifically answer humanlike causal queries. Figure 3 shows the main components of the Canvas; the right-hand side shows the standard features for the three steps.

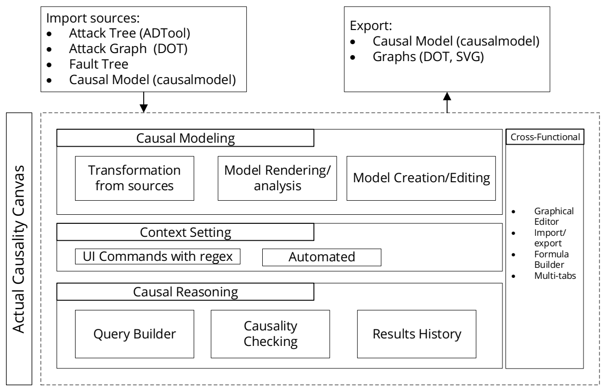
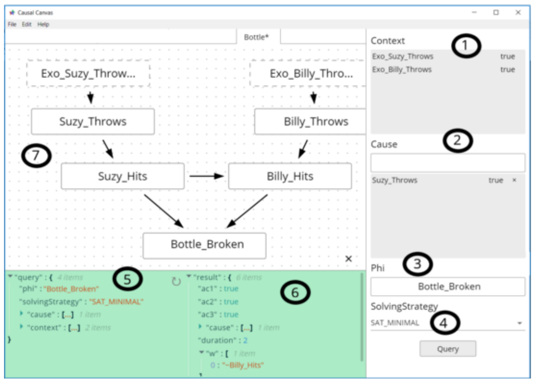


Figure 3. The Components of the Actual Causality Canvas

The generality of the modeling mode in the Canvas stems from building it on the basis of Definition 1 (extension .causalmodel). We used a machine- and human-readable format (JSON) for this purpose; the newly created causal models are written and saved using this format. The Canvas already transforms sources such as an attack tree modeled using ADT tool [23], or a fault tree modeled using EMFTA tool [7], or any graph formatted using DOT. Intuitively, the Canvas can also read already transformed or created causal models. The import functionality is implemented using a plugin-based architecture, keeping the door open for an easy extension to include new sources of knowledge. Alternatively, wrappers can be written to generate a (.causalmodel) file directly from other sources. Besides model transformation, the Canvas renders the models using different layout algorithms like $d3$ and $dagre$. A set of visual and textual tools to create nodes, edges, and formulas *from scratch* are implemented. Furthermore, since human readability is a crucial aspect, the Canvas is equipped with features that enable the user to grasp larger models by focusing on parts of the graph.

Context setting is enabled with a specific field in the (.causalmodel) format. Programmatically, the context can then be set by writing the values into the respective field in the file. Alternatively, the Canvas provides a *command input function* that allows the user to set the values of the *exogenous variables* (context). This is implemented using a practical filter-and-set functionality that uses regular expressions to select the variables.

For the reasoning mode (Figure 4), the Canvas is equipped with a solving back-end based on the tool provided in [19]. This tool offers different solvers for actual causality that differ in technology (SAT or Brute-force) and accounting for responsibility with a minimal $\vec{W}$ and either check or infer causality. The back-end is embedded with the Canvas bundle, and it promptly answers queries (less than 6 s for models of 8000 nodes). The reasoning mode is activated using a specific button that displays a special screen for the query construction. The different elements of the query (context, cause, and effect) are easily manipulated in this screen. Once a query is ready, a request to the solver is sent, and the result is then shown back to the user. The result details whether each Definition 2 condition passed or not, along with a $\vec{W}$. To realize the requirement of interactivity, the Canvas tracks all causal queries in the same session; the user can navigate back and forth within them. This way, a user can adapt their queries and play with different assumptions, contexts, and effects.



```
Context
Exo_Suzy_Throws      true
Exo_Billy_Throws     true

Cause
Suzy_Throws          true  ×

Phi
Bottle_Broken

SolvingStrategy
SAT_MINIMAL
```

```
▾ "query" : { 4 items
    "phi" : "Bottle_Broken"
    "solvingStrategy" : "SAT_MINIMAL"
  ▸ "cause" : [...] 1 item
  ▸ "context" : [...] 2 items
}
```

```
▾ "result" : { 6 items
    "ac1" : true
    "ac2" : true
    "ac3" : true
  ▸ "cause" : [...] 1 item
    "duration" : 2
  ▾ "w" : [ 1 item
      0 : "~Billy_Hits"
```

Figure 4. The Reasoning Mode in the Canvas:
1- Context, 2- Hypothesized Cause, 3- Effect Formula, 4- Solving Strategy, 5- Query Request, 6- Query Result, and 7- Model

4 Use Cases

In this section, we show how the framework to operationalize actual causality is instantiated in xAI, accident investigation, and drone crash diagnosis. For each use case, we detail the modeling process, the context setting, and the reasoning aspect.

4.1 Explainable AI

Explanation and interpretability in AI (xAI) and machine learning systems are attracting attention because they are necessary for regulation compliance, system improvements, and trust enhancement [33]. The work in xAI mainly focused on model-based approaches that approximate the true criteria of classifiers as gradient-based (binarized like in [38]) or decision tree-based models [33]. Recent articles by Miller [32] and Mittelstadt et al. [33] suggest that existing approaches have not yet been built on relevant definitions from philosophy, social science, and cognitive science. Instead, they provide "general scientific explanations" The authors argue that humans would seek an "everyday contrastive, social explanation" that explains "why particular facts occurre" [32, 33]. Following Miller, we see HP as an enabler of such explanations. We understand that there is more to xAI than what we are suggesting here; however, we only show how explicit representations from the literature (like the one in our example) can be incorporated into the Canvas, knowing that this is incomplete. We discuss the three requirements: a representation of the classifier's behavior, i.e., a causal model; the exact information about the classified point, i.e., the context; and the reasoning machinery. We show one simple transformation to causal models and straightforward query construction.

Causal model. To illustrate this approach, we use an example provided by Miller [32] and highlighted again in [33]. The example (displayed in Table 1) considers the features and parameters learned by an algorithm to classify types of arthropods. Some features express binary facts such as whether an arthropod has a stinger or not while others are integer-based, e.g., the number of legs. Since we only consider binary models, we use a binary representation of the integer values. Although there may be other elegant ways to handle decimal features, we believe that a binary model captures what we want to explain. For instance, the fact that an arthropod has eight legs makes the algorithm classify it as a spider; the number *eight* is not needed in an arithmetic way.

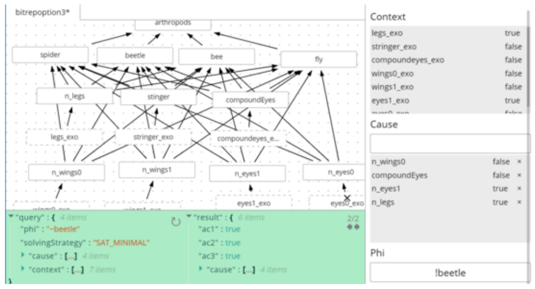
Depending on the algorithm, different representations of the learned models exist. We use the one presented in the example: a tabular set of features and the values corresponding to one class. Each class and each binary feature is presented by an *endogenous* variable. To accommodate for its values, a nonbinary feature variable is presented by a set of bits. For example, the "number of wings" $(N.W)$ feature has the values 0, 2, or 4, and then it will be represented with *two* bit-variables, $N.W_1, N.W_0$. The different values of the two bits correspond to an index of the real values, i.e., 00 means 0, 01 means 2, and 10 means 4. We are not transforming the decimal values in the table into their binary representation but rather assigning them a binary value of some number of bits. The number depends on the count of distinct values presented. In the example, $N.W$ has three values, and hence, we use two bits. This way, we use fewer variables. We also express each feature (or feature bit) with an exogenous variable.

Table 1. A Lay Model for Classifying Arthropods[32]

Type	No. Legs	Stinger	No.Eyes	Compound Eyes	Wings
Spider	8	✗	8	✗	0
Beetle	6	✗	2	✓	2
Bee	6	✓	5	✓	4
Fly	6	✗	5	✓	4

In addition to the variables, we create the propositional equations from the table. For instance, a *spider* is an arthropod with 8 legs, *and* not a stinger, 8 eyes, no compound eyes, and no wings. Then, each class variable is a conjunction of the features' value, e.g., $spider = (no.wings = 0) \wedge (!stinger)$. Note that $(no.wings = 0)$ is presented as explained above. As such, we create a causal structure (shown in Figure 5) of the factors leading to classifying an arthropod. Lastly, although not part of the example, the variance of the features' importance (e.g., weights) is a candidate for a preemption relation.

Context setting and causal reasoning. We can now provide *explanations* about specific classifications. The specific case (e.g., an image J of an arthropod) is seen as a vector of features. This is precisely the *context* of a causal query. Formally, the context is an assignment of exogenous variables; we can easily see that it maps to the values of the features we consider. For example, *image* J contains 8 legs, no stinger, 8 eyes, no compound eyes, and no wings. What remains is the phrasing of the causal query. Miller [32] argues that the human perception of an explanation often refers to a contrastive question, i.e., why P rather than Q. We think such a question can be formulated by focusing on what is the effect φ. The most basic contrastive question can then be seen as $(\neg Q)$. The result of the causal query can then be considered a contrastive explanation. For example, the answer to the question *Why is image J labeled as a spider instead of a beetle?* [32] is the list of causes (as seen in Figure 5): the compound eyes, $N.W$, number of eyes, and number of legs. Obviously, the decision-tree nature of the example simplifies it; however, we stress that our goal is to show the ability to incorporate ideas from the literature of approximating classifier behaviors into simpler models and augment it with contrastive reasoning capabilities.



```
Context
legs_exo             true
stringer_exo         false
compoundeyes_exo     false
wings0_exo           false
wings1_exo           false
eyes1_exo            true

Cause
n_wings0             false  ×
compoundEyes         false  ×
n_eyes1              true   ×
n_legs               true   ×

Phi
!beetle
```

```
▾ "query" : { 4 items
    "phi" : "~beetle"
    "solvingStrategy" : "SAT_MINIMAL"
  ▸ "cause" : [...] 4 items
  ▸ "context" : [...] 7 items
}
```

```
▾ "result" : { 6 items
    "ac1" : true
    "ac2" : true
    "ac3" : true
  ▸ "cause" : [...] 4 items
```

Figure 5. The Causal Graph of the Example

4.2 Aircraft Accident Investigation

On July 1, 2002, two aircraft (Tupolev Tu154M and a Boeing 757-200) collided in mid-air in southern Germany (Ueberlingen), killing all the people on board [34]. An investigation by the German Federal Bureau of Aircraft Accident Investigation (BFU) and a WBA by a research group documented the many interweaving factors surrounding the accident [34, 43, 40]. Briefly, a series of coinciding events led to the collision, including an exceptional heavy load on the ground ATC, conflicting advisory commands of the ATC and the collision avoidance systems (TCAS) to the Tu154M crew, system degradation of the short term collision avoidance (STCA), and communication issues due to a maintenance operations at the ATC office.

Causal model and context setting. The WBA is a formal procedure introduced by Ladkin to investigate accidents and propose countermeasures for future prevention [25, 26, 40]. The result of a WBA is a graph called the why-because-graph (WBG), which structures causal factors (nodes) and their relations (edges) via analysis of official reports. The Ueberlingen WBG contained 95 factors (seen in [44]). With some adaptations, we use WBG as a source for causal models. We transform a node into the WBG to an endogenous variable in the causal model. Leaf nodes are considered as an exogenous variable and are always true because the WBA creates them from reports (actually happened). Lastly, we create the equations for each variable by manual inspection: was is it conjunction or disjunction of variables that led to it? During this step, we also considered *preemption* relations, especially when events coincide. For example, two systems are implemented to avoid midair collisions—ATC aided with STCA, and *additionally*, the aircraft's TCAS. Accordingly, the TCAS is the last resort that should resolve last-minute issues [34]. Thus, a failure by the ATC *preempts* a failure by the TCAS. Another example of preemption relations was added among the factors that led to the late ATC intervention (denoted as e_{49} in [44]). There were five coinciding factors, two of them were e_{56} *Heavy load on the ATC* and e_{62} *Crossing routes*. Arguably, people tend to consider *exceptional* events as probable causes and not as regular events [12]. Thus, we argue that the exceptional heavy load on the ATC (because of a late landing on a nearby airport and a faulty phone system) preempts other factors such as e_{62}. According to this argumentation, we added preemption relations among these events.

Causal reasoning. Since WBA aims to produce a list of countermeasures, we simulated our first check to automate the manual WBA sufficiency test [26], which checks if the effect eventually happens given the occurrence of all the root causes. Specifically, we checked Q_1 : *Is* $\vec{X}$ *a cause of the collision?* where $\vec{X}$ is the set of 31 leaf events, which passed with an empty $\vec{W}$. Next, we looked for a minimal cause of the accident. Interactively, we found a minimal cause of 14 variables, which were mainly the events resulting in the ATC intervention delay. This cause conforms with the immediate cause reported by the BFU [34]. We formalized causal queries on a more abstract level of details and found minimal causes on a coarser level than the 14 events. Although knowledge around this accident already existed, the advantage of the canvas is that it automates the interactive analysis to investigate complex situations with large causal graphs. We saw how accountability is enabled by domain-specific methodologies such as WBA.

4.3 Drone Crash Diagnosis

Drones, such as quadcopters, recently found widespread use; however, their safety is a significant concern. In the case of drone failure, it is essential to identify the cause and prevent it in the future. In this use case, we consider a realistic example where the Canvas is used to model a system from scratch and assist in investigating incidents.

Drones have several physical and software components, including actuators, sensors, and controllers. The components that interact with the physical world are called *actuators*, e.g., the electrical engine. *Sensors* are devices that measure physical properties; for example, GPS measures the location and altitude. Finally, *software components* are virtual units that organize all hardware components and process the information to keep the drone stable in flight. For example, the sensor fusion module receives readings from sensors and estimates an approximate value based on the readings. The course of a flight comprises several coinciding events related to different components. The diversity of such events and their causal connection render the diagnosis difficult. As we see in this use case, the *Canvas* is a practical method, especially when investigation from scratch.

Causal model. Each node in our model describes an action of a specific component, such as the failure of the GPS sensor or when the drone was being pushed by the wind. The model is built based on domain knowledge or data-driven approaches. In previous work, we deduced a *fault tree* from the drone's architecture [45]. Here, we use a similar fault tree while adding preemption relations based on the results from a practical course we held with computer science students [46]. The students used the Canvas to create their causal models. The preemption rules originate from the nature of the control loop that is being executed repeatedly during flight. In this sense, the failure of the actuators preempts that of the controller software, which then preempts the failure of the sensors. Moreover, among the software components, path tracking failure preempts path planning failure. After adding these relations to the fault tree (imported to the Canvas), we obtain our model.

Context setting and reasoning. All the nodes in our causal model are events. Simply put, an event describes an action performed by a specific component. For example, the number of detected satellites by the GPS sensor dropping below 9 is an indication of a *gpsFailed* event. For context setting, first, detection analysis should be run over the data to detect the events that occurred. If an exogenous event is found in the flight logs, its value is set to true in the *Canvas*. The endogenous variables, such as *sensorsfailed* or *actuatorsfailed*, will be computed based on their respective equations implied by fault tree semantics. We consider two scenarios based on real flight logs collected from users of an open-source quadcopter [10].

In the first scenario, we analyze a case of engine failure that resulted in a crash. Engines have an essential role in keeping the drone in the air, and their failure leads to altitude loss. If the commanded signal to an engine is set to the maximum value for more than a second, then one can assume that the engine has failed. We set *engine1Failed* to true on the basis of our assumption. Moreover, *altitudeLoss* can be detected when the altitude drop rate is more intense than a threshold. This is seen in the mentioned flight log. Other nodes such as *accelerometerFailed* or *windPushed* are set to false either because they did not occur in that specific log, or there were no relevant sensors to record them. Now, constructing a query with a hypothesized cause *engine1failed* for *altitudeLoss* returns true. Other hypothesized causes, such as *gpsFailed*, result in a negative response in the Canvas. In a second scenario, which was also seen in the real flight logs, both *the path tracking and path planning* modules of a drone failed. Our objective was to query which one was the actual cause. We set the value of these two events to true. Also, the value of *altitudeLoss* is set to true since this event occurred according to the log. Although it is not trivial that *pathTrackingFailed* is the actual cause, one can easily deduce this using the *Canvas*. This is because of the preemption relation between the *pathPlanningFailed* and *pathTrackingFailed* nodes. This preemption rule originates from the domain knowledge where path tracking is closer to the final physical output of the drone than the path planning module in the control loop.

Although the logs are not labeled, i.e., the causes are not known to us, the added value of the *Canvas* lies in its ability to import fault trees and compute the causality in large models where it is nonintuitive for the investigator to deduce causality between events.

5 Evaluation

In addition to utilizing the framework and *Canvas* in different use cases, we briefly report our evaluation of the *Canvas*.

Display performance. We tested how the *Canvas* performs when displaying different, randomly generated models. The models vary in size between 10 and 4000 nodes. We tested critical functions such as the graph layout, graph navigation, and zooming. All tests were executed on an Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz, 2001Mhz, 4-core(s) Processor, and 16GB RAM. The *Canvas* uses two algorithms to lay out the imported graphs, $dagre$, and $d3-force$. For the quality of rendering, the $dagre$ layout is suitable for smaller causal models (fewer than 50 nodes). The nodes here are nicely separated with sufficient space. The $d3-force$ layout is suitable for almost any model size up to 600 nodes. For the rendering *time*, the layout methods differ as well. $Dagre$ is usable only for models smaller than 400 nodes, whereas $d3-force$ responds within a reasonable time with models up to 4000 nodes (taking around 8 s to render). The *Canvas* automatically switches its layout method on the basis of the model size. The other features perform well with causal models up to around 800 nodes, after which some lag is seen with zooming and navigation. As human cognition is limited, operators will not grasp large models. Thus, we verified that the Canvas facilitates focusing on parts for models up to 4000 nodes. We manually tested inspecting such models by highlighting and zooming into their parts. This is an effective feature where the user highlights a reachable part of the graph based on a filtering function. Of course, we do not see a point of using a graphical editor with very large models, but the goal is to document the performance of the *Canvas*.

Usability. We also conducted a preliminary user study with 10 computer science graduate students (working in five groups). As part of a practical course, they used the *Canvas* to develop their custom causal models and queried the actual causes based on real drone crash data [46]. An overview of the HP definition, illustrated by examples bundled in the *Canvas*, was sufficient for the students to grasp the concept of causality and use it in their drone crash scenarios. We collected the students' feedback through written forms and face-to-face interviews. The five groups reported an *effective* usage of the *Canvas* for the task. Some of their feedback included *"powerful utility that is easy to operate"* and *"a great tool to quickly analyze if the proposed model is indeed correct."* They stressed its power, especially when dealing with large models or confusing situations. The students also suggested some enhancements that we are considering such as an undo feature, a custom highlight of preemption relations, and a programmable interface that exposes all phases to other systems.

6 Related Work

Previous versions of the HP definition are applied as a supplement to other technologies. So far, these applications, to our knowledge, adapted the theory to a domain-specific context, with simplifications and restrictions on the definition. This is clearly sufficient for the particular use case, but we argue that a general approach toward actual causality may enable new socio-technical applications; such an approach is lacking in the literature. Examples of using HP are seen in the domain of databases [30, 4, 39], where causality is used to explain a query by enhancing the provenance information. Database work uses domain-specific concepts such as lineage and database repairs; it also limits the theory to a single-equation model [30] and a no-equation model in [4, 39]. To verify models, the authors in [2, 27, 3] blinded the theory with model checking and enhanced a counterexample with causal reasoning. The relaxation of the theory is based on the fact that no equations are required. In [2, 27], the authors focused on the efficiency of causal reasoning as part of bounded model checking. In contrast with our approach, these applications cannot be used outside their domains because of restrictions (e.g., single-equation) or dependency on other concepts (e.g., counter-examples). Also, they only deal with a specific part of the theory, i.e., reasoning.

In the domain of accident (aircraft and railways) investigation, WBA tools and methodologies are relevant to our work [25, 26, 40]. The WBA Software Toolkit provides functionalities that support an incident investigation, especially in modeling and structuring the occurred factors. Our approach, on the other hand, differentiates modeling and context since it is plausible to use models of recurring behavior among incidents. Also, since WBA aims to list all the sufficient causal factors of an accident, the toolkit does not provide an actual causality reasoning function.

Similarly, threat and hazard modeling tools, such as ADT for attack trees [23] and EMFTA for fault trees [7], present the user a model editor and analysis tools. However, explicit context setting and actual causality reasoning are not part of the editors. Also, the ability to import other knowledge sources and transform them into causal models is not supported in all these tools.

A significant body of work is published around xAI; for an overview of post-hoc human explanations, see [33, 32]. In our use case, we did not propose a complete solution like the local explanation in [38]. However, our goal was to emphasize the connection between our approach and xAI. Still, significant work is needed in the domain of modeling for xAI, possibly using our framework.

7 Conclusions and Future Work

This paper provides a unifying framework that generalizes existing approaches to accountability and explainability, which applies to different contexts. As modern systems could harm people, damage their assets, or decide their loan adequacy, such systems ought to be at least explainable. To that end, our framework is intended to solve explanation-based problems for a wide range of systems in the future. Advancing operationalizations, the framework is bundled as an interactive platform. We have shown how different knowledge sources can be transformed into structural-equations models and then used for an automated analysis—using HP actual causality. We conclude that our framework is generalizable enough to accommodate explanation-based socio-technical constructs, and with tool support, it is amenable to be incorporated into different domains.

Conceptually, our future work lies in expanding our approach with additional general components, such as probabilities, and studying its corresponding domain-specific notions. We also aspire to develop our list of use cases with new real-world applications.

ACKNOWLEDGEMENTS

This work was supported by the German National Science Foundation (DFG) under grant no. PR1266/3-1, Design Paradigms for Societal-Scale Cyber-Physical Systems.

REFERENCES

[1] Gadi Aleksandrowicz, Hana Chockler, Joseph Y. Halpern, and Alexander Ivrii, 'The computational complexity of structure-based causality', in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, (2014).

[2] Adrian Beer, Stephan Heidinger, Uwe Kühne, Florian Leitner-Fischer, and Stefan Leue, 'Symbolic causality checking using bounded model checking', in *Model Checking Software - 22nd International Symposium, SPIN*, (2015).

[3] Ilan Beer, Shoham Ben-David, Hana Chockler, Avigail Orni, and Richard J. Trefler, 'Explaining counterexamples using causality', *Formal Methods in System Design*, **40**(1), 20–40, (2012).

[4] Leopoldo Bertossi, 'Characterizing and computing causes for query answers in databases from database repairs and repair programs', in *International Symposium on Foundations of Information and Knowledge Systems*, pp. 55–76. Springer, (2018).

[5] Hana Chockler and Joseph Y. Halpern, 'Responsibility and blame: A structural-model approach', *J. Artif. Intell. Res.*, **22**, 93–115, (2004).

[6] Hana Chockler, Joseph Y Halpern, and Orna Kupferman, 'What causes a system to satisfy a specification?', *ACM Transactions on Computational Logic (TOCL)*, **9**(3), 20, (2008).

[7] Julien Delange. Emfta: an open source tool for fault tree analysis.

[8] Joan Feigenbaum, James A. Hendler, Aaron D. Jaggard, Daniel J. Weitzner, and Rebecca N. Wright, 'Accountability and deterrence in online life', in *Web Science 2011, WebSci '11, Koblenz, Germany - June 15 - 17, 2011*, pp. 7:1–7:7, (2011).

[9] Joan Feigenbaum, Aaron D. Jaggard, and Rebecca N. Wright, 'Towards a formal model of accountability', in *2011 New Security Paradigms Workshop, NSPW '11*, pp. 45–56, (2011).

[10] ArduPilot Discuss Forums. Ardupilot discourse.

[11] Joseph Y. Halpern, 'A modification of the Halpern-Pearl definition of causality', in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*, pp. 3022–3033, (2015).

[12] Joseph Y. Halpern, *Actual causality*, The MIT Press, Cambridge, Massachussetts, 2016.

[13] Joseph Y. Halpern and Judea Pearl, 'Causes and explanations: A structural-model approach - part I: causes', in *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001*, pp. 194–202, (2001).

[14] 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).

[15] Mark Hopkins, 'Strategies for determining causes of events', in *AAAI/IAAI*, (2002).

[16] David Hume, *An Enquiry Concerning Human Understanding*, 1748.

[17] Amjad Ibrahim, Stevica Bozhinoski, and Alexander Pretschner, 'Attack graph generation for microservice architecture', in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pp. 1235–1242. ACM, (2019).

[18] Amjad Ibrahim, Severin Kacianka, Alexander Pretschner, Charles Hartsell, and Gabor Karsai, 'Practical causal models for cyber-physical systems', in *NASA Formal Methods*, eds., Julia M. Badger and Kristin Yvonne Rozier, pp. 211–227, Cham, (2019). Springer International Publishing.

[19] Amjad Ibrahim, Simon Rehwald, and Alexander Pretschner, 'Efficient checking of actual causality with sat solving', *Dependable Software Systems Engineering*, **53**(1), 241 – 255, (2019).

[20] Severin Kacianka, Amjad Ibrahim, Alexander Pretschner, Alexander Trende, and Andreas Lüdtke, 'Extending causal models from machines into humans', in *4th Workshop on Formal Reasoning about Causation, Responsibility, & Explanations in Science & Technology*, (2019).

[21] Severin Kacianka, Florian Kelbert, and Alexander Pretschner, 'Towards a unified model of accountability infrastructures', in *Proceedings of CREST@ETAPS 2016,*, pp. 40–54, (2016).

[22] Samantha Kleinberg and George Hripcsak, 'A review of causal inference for biomedical informatics', *Journal of Biomedical Informatics*, **44**(6), 1102–1112, (2011).

[23] Barbara Kordy, Piotr Kordy, Sjouke Mauw, and Patrick Schweitzer, 'Adtool: security analysis with attack–defense trees', in *International conference on quantitative evaluation of systems*, pp. 173–176. Springer, (2013).

[24] Barbara Kordy, Ludovic Piètre-Cambacédès, and Patrick Schweitzer, 'Dag-based attack and defense modeling: Don't miss the forest for the attack trees', *Computer science review*, **13**, 1–38, (2014).

[25] Peter Ladkin and Karsten Loer, 'Why-because analysis: Formal reasoning about incidents', *Bielefeld, Germany, Document RVS-Bk-98-01, Technischen Fakultat der Universitat Bielefeld, Germany*, (1998).

[26] Peter B Ladkin, 'Causal reasoning about aircraft accidents', in *International Conference on Computer Safety, Reliability, and Security*, pp. 344–360. Springer, (2000).

[27] Florian Leitner-Fischer, *Causality Checking of Safety-Critical Software and Systems*, Ph.D. dissertation, University of Konstanz, Germany, 2015.

[28] Florian Leitner-Fischer and Stefan Leue, 'Causality checking for complex system models', in *Verification, Model Checking, and Abstract Interpretation, 14th International Conference, VMCAI 2013, Rome, Italy, January 20-22, 2013. Proceedings*, (2013).

[29] David Lewis, 'Causation', *Journal of Philosophy*, **70**(17), 556–567, (1973).

[30] Alexandra Meliou, Wolfgang Gatterbauer, Joseph Y. Halpern, Christoph Koch, Katherine F. Moore, and Dan Suciu, 'Causality in databases', *IEEE Data Eng. Bull.*, **33**(3), 59–67, (2010).

[31] Alexandra Meliou, Wolfgang Gatterbauer, Katherine F. Moore, and Dan Suciu, 'Why so? or why no? functional causality for explaining query answers', *CoRR*, **abs/0912.5340**, (2009).

[32] Tim Miller, 'Explanation in artificial intelligence: Insights from the social sciences', *Artificial Intelligence*, (2018).

[33] Brent Mittelstadt, Chris Russell, and Sandra Wachter, 'Explaining explanations in ai', in *Proceedings of the conference on fairness, accountability, and transparency*, pp. 279–288. ACM, (2019).

[34] German Federal Bureau of Aircraft Accident Investigation. Investigation report ax001-1-2/02, 2004.

[35] Judea Pearl, 'Causation, action and counterfactuals', in *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge, De Zeeuwse Stromen, The Netherlands, March 17-20 1996*, pp. 51–73, (1996).

[36] Judea Pearl and Dana Mackenzie, *The book of why: the new science of cause and effect*, Basic Books, 2018.

[37] Simon Rehwald, Amjad Ibrahim, Kristian Beckers, and Alexander Pretschner, 'Accbench: A framework for comparing causality algorithms', in *CREST@ETAPS 2017, Uppsala, Sweden, 29th April 2017.*, pp. 16–30, (2017).

[38] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, 'Why should i trust you?: Explaining the predictions of any classifier', in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144. ACM, (2016).

[39] Babak Salimi and Leopoldo Bertossi, 'From causes for database queries to repairs and model-based diagnosis and back', (2014).

[40] Dipl-Inform Jan Sanders, 'Introduction to why-because analysis', *Dipl.-Inform, February*, (2012).

[41] Bruce Schneier, 'Attack trees', *Dr. Dobb's journal*, **24**(12), 21–29, (1999).

[42] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M Wing, 'Automated generation and analysis of attack graphs', in *Proceedings-IEEE Symposium on Security and Privacy*, p. 273. IEEE, (2002).

[43] Jörn Stuphor. Handout of the 2002 ueberlingen mid-air.

[44] Jörn Stuphor. The wbg of 2002 ueberlingen mid-air.

[45] Ehsan Zibaei, Sebastian Banescu, and Alexander Pretschner, 'Diagnosis of safety incidents for cyber-physical systems: A uav example', in *2018 3rd International Conference on System Reliability and Safety (ICSRS)*, pp. 120–129. IEEE, (2018).

[46] Ehsan Zibaei and Alexander Pretschner. Automated diagnosis of drone crashes. https://www22.in.tum.de/en/teaching/automated-diagnosis-drone/. Accessed: 2019-11-25.