CIRCE: a Scalable Methodology for Causal Explanations in Cyber-Physical Systems

Samuel Reyd, Ada Diaconescu, Jean-Louis Dessalles
Telecom Paris, LTCI, IPParis, Palaiseau, France
name.surname@telecom-paris.fr

***Abstract*—Cyber-physical systems (CPS) are increasingly complex and harder for human users to understand. Integrating explainability methods within their design is a key challenge for their acceptability and management. We consider that causal explanations can provide suitable answers to address this issue. Most approaches to causal explanations, however, rely on global system models, often built offline, which implies heavy computations, delays, and interpretability issues when answering questions at runtime. We propose CIRCE: a scalable method for Contextual, Interpretable and Reactive Causal Explanations in CPS. It is an abduction method that determines the cause of a fact questioned by users at runtime. Its originality lies in finding a cause instead of an entire causal graph to explain CPS behavior and employing a classic local Explanatory AI (XAI) technique, LIME [1], to approximate this cause. We validate our method via several simulations of smart home scenarios. Results indicate that CIRCE can provide relevant answers to diverse questions and scales well with the number of variables. Our approach may improve the efficiency and relevance of causality-based explanations for CPS and contribute to bridging the gap between CPS explainability and classic XAI techniques.**

I. Introduction

CPS are software systems that interact with a physical environment [2]. Modern CPS integrate an ever-increasing number of interconnected components and operate within unpredictable environments. The ensuing complexity makes it difficult for humans to understand, adopt, and interact with such CPS. This may lead to low acceptability from users, as well as to increasing costs of engineers and administrators who can no longer prevent system failure. Furthermore, autonomic or self-adaptive CPS [3], [4] dynamically adjust themselves to changes in their internal resources or external environment to attain their goals. This requires runtime explanations that are difficult (or impossible) to compute offline.

Explainability has always been a key concern for CPS design [5]. Several approaches have been proposed – e.g. defining specific explanation layers [5], highlighting the impact of variables on the system decision process [6], [7], or learning the causal links amongst system variables [8]. Our research contribution focuses on the latter explanation type.

The idea behind most causality-based explanation methods is to learn a representation of the underlying causal model using observations and interventions [8], [9]. Most often, this consists of a Causal Bayesian Network (CBN), which highlights dependencies between variables, hence providing a suitable base for explanations and debugging.

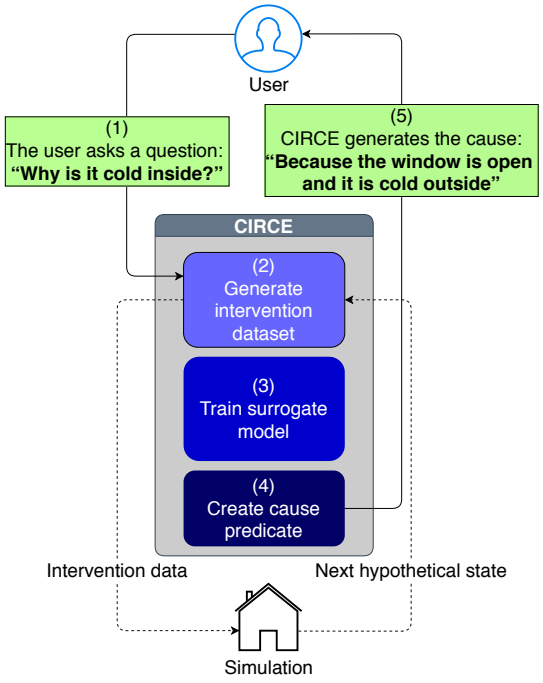

Fig. 1. CIRCE method overview

However, CBN are expensive to learn automatically, as each variable must compute its dependency with all subsets of variables. This also makes them hard to maintain, as system changes may require the entire graph to be rebuilt. Moreover, CBN are ill-suited for generating relevant explanations at runtime, as they ignore the runtime context within which the question was raised. For example, consider that a room's temperature is influenced principally by the outside temperature if its window is open; and mostly by its heater power if its window is closed. In case of a user complaining about the room temperature (e.g. 'why is it cold?'), we only want explanations to mention the outside temperature in contexts where the window is open; and only mention the heater state in contexts where the window is closed. A CBN would retrieve the entire set of possible causes, along with their probabilities, without filtering out contextually irrelevant causes.

Our solution aims to address these shortcomings by: i) only relying on partial, or "local" causal models that are relevant to the question asked (to improve scalability and limit delays); ii) generating such local causal models at runtime (to consider

the up-to-date system behaviour); and iii) taking the runtime context into account (to help ensure relevant answers).

CIRCE leverages the causal potential of classic XAI techniques for ML-models to generate explanations. We suppose that the targeted CPS can be modeled as a discrete Markovian state sequence (i.e. each state only depends on the previous one). This implies that all direct effects of a given cause are observable at the next timestep. The method could be extended to consider several timesteps (future work). We also suppose that we have access to a transition function, i.e. a function that maps any system state to the next one (the transition function is implemented as a system simulation in our experiments). We do not make further assumptions on the nature of this function. Especially, contrary to most explainability methods [10], [**?**], [11], our approach does not apply only to rule-based systems, and we do not rely on such rules if they exist. Notably, our approach could work with a stochastic system.

We implemented a set of simulations of smart houses to highlight the capabilities of our method. We report qualitative results on a single smart home as an illustration and then on a neighborhood to assess the scalability of our approach. Results indicate that CIRCE can provide relevant explanations to various types of questions; that the quality of explanations is not affected by the number of dimensions considered; and that its processing time scales linearly with the number of dimensions.

II. Background

We consider explanations that answer questions of the form: "Why do we have T in the current state?". We provide answers of the form: "We have T in the current state because C was true in the previous state". Here, T and C are logical predicates – i.e. boolean statements about system variables. T is the observed fact, or target, that the user is questioning. C is a "but-for" cause of T, that CIRCE provides as an answer. A "but-for" cause C of T implies that changing the boolean value of C would also change the boolean value of T – i.e. if C weren't true then T wouldn't be true either. Hence, CIRCE only identifies a cause C for answering a questioned fact T, at runtime, rather than modeling the CBN for the entire system. These explanations contribute to guiding users to solve the problems related to the query or to increasing trust in complex systems with better interpretability of surprising behaviors.

Finding a but-for cause implies identifying the set of system variables that can be forced to alternative values such that the consequence no longer holds – this has exponential complexity. Instead, We propose searching for an approximate cause by determining the variable values that are most likely to influence T. To achieve this, we adopt a classic technique from the field of Explainable Artificial Intelligence (XAI) – i.e. LIME [1] – developed for explaining predictions of Machine Learning (ML) models; and we adapt it to the CPS context. Figure 1 depicts an overview of the proposed CIRCE method.

An ML model is a function that aims to approximate a hypothetical perfect function that solves a task (e.g. labeling images). The ML model is often a parametric function whose parameters are learned to fit some data (e.g. pairs of images and their expected labels). For highly complicated tasks this parametric function may be highly complex (e.g. a deep neural network). Consequently, when the function's behavior is not as expected, it is difficult to understand why. To address this, local XAI aims to identify which parts of the input contributed to the prediction of the ML model (e.g. which pixels or groups of pixels contributed to labeling a given image as a 'frog'). The LIME method generates a dataset of perturbations of the original image, labels them with the output of the original ML model, and trains a linear regression (LR) – called local surrogate model – to approximate the original ML model. The LR's behavior mimics the original ML model while facilitating the interpretation of its predictions in terms of pixel importance – i.e. its output is a weighted average of the value of each pixel. To get a simpler interpretation of the original image, LIME uses its LR prediction to approximate the actual ML model.

Figure 2 shows how we adapt LIME to provide explanations within the CPS domain. LIME performs an LR to explain the output of a black-box ML-model for a given input. Instead, we learn a decision tree to explain why a target predicate T (i.e. an observed fact, or system state) occurs given the previous state. We choose decision trees over LR because it allows us to generate a cause instead of an explanation based on variable importance (see fig. 2). To avoid repeated manipulations of the real system, we determine the links between system states via a simulation.

Despite using an ML model, our approach does not detect spurious correlations in the environment as causes. This is because the model is trained on data generated by applying interventions on the system state. For instance, consider a spurious correlation in the environment between a user's presence in a room and the room temperature (a heater would activate when a user is present in the room). Our approach uses interventions to force the value of the user's presence or the heater activation independently. It becomes possible to see that the heater is the cause of the high temperature and not the user's presence.

III. Related Work

A. Post hoc local black-box XAI

Recently, significant work was dedicated to explaining the prediction of complex machine learning models [12]. Our approach is inspired by methods from the field of post hoc, local, black-box XAI. This encompasses a set of techniques that are applied after the original model training, analyzing and explaining a single prediction, without considering the nature of the original ML model.

LIME (Local Interpretable Model-agnostic Explanations) [1] is one of the most influential of these methods. Our approach adapts this method from ML-model interpretability to cause generation in a CPS.

Another highly influential method is SHAP (SHapley Additive exPlanations) [13]. It leverages methods inspired by the computation of Shapley values [14] to refine certain steps of

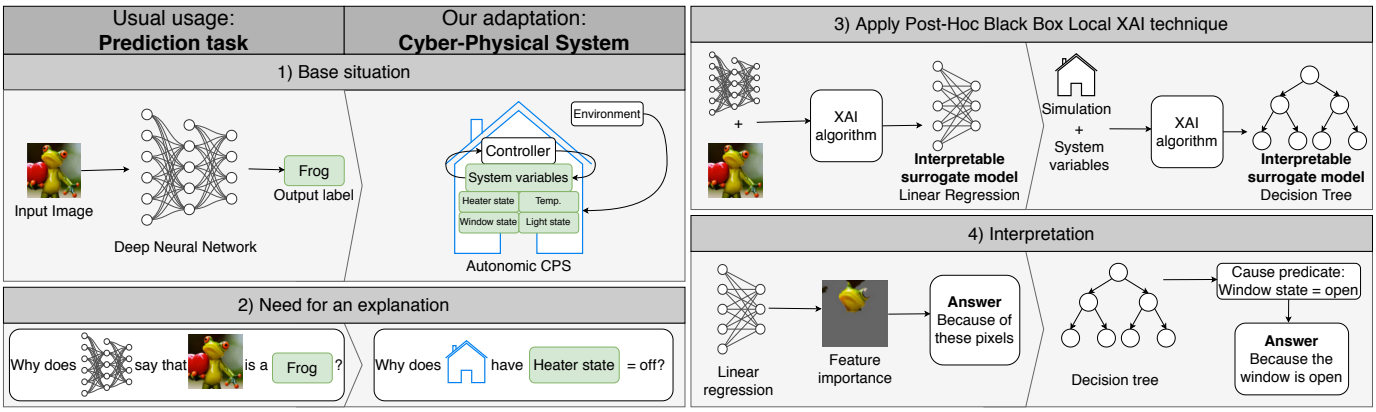

Fig. 2. Adapting Local Post-Hoc Black-Box XAI techniques to generate causal explanations for CPS

the LIME methodology. Notably, the sampling of training data is done by "ignoring" coalitions of dimensions. This is done by replacing certain dimensions with values drawn uniformly from the dataset used to train the original model. Additionally, they use Shapley coefficients for weighting the data during the training of the surrogate model.

B. CPS explainability

Explainability has long been a focal point for Cyber-Physical Systems (CPS). Foundational works such as MAP-Ex [5] propose a comprehensive design framework encompassing an explanation layer. They offer a general basis for explainable CPS, but lack specificity regarding the actual explanation methodology. Subsequent approaches refined the concept of explainability.

Similarly to our approach, others have defined explanations as causes of a problem. Ziesche et al. [15] define explanations as the origin of an anomaly. They only consider a limited number of possible causes and train a classification model; instead of using the system dynamics to identify causes online (as CIRCE does). Houze et al. [10], define explanations as an abduction trace produced by a cognitive conflict solving process. The need for explanation is triggered by a cognitive conflict between a predicate that is believed or wished to be true but that is false in the current model. This conflict is solved by negating possible causes with an abduction mechanism and a simulator. The definition of explanation relies on causality and abduction. This approach necessitates an external abduction mechanism to suggest causes. Our approach generates candidate causes and complements this approach.

Some other definitions of explainability for CPS do not rely on causality between the system variables but rather on the system's decision process. XSA [6] uses the same local XAI method as we do, i.e. LIME [1], to highlight the impact of the system variables. They compute feature importance for the system action selection function instead of conditions on the system variables. Sukkerd et al. [7], offers a method for verbal justifications of actions in multi-objective planning systems. These explanations aim to highlight which goal is being optimized and what reward the system hopes to obtain.

In [11], the authors propose to focus on contrastive explanations in rule-based systems. They compute what the user might have expected and why the expected outcome was not attained. In addition, [**?**] proposes user-specific modifiers to enhance explanation methods. Our approach complements these methods by providing the abduction mechanism necessary for their operation, which they circumvent by relying on externally provided rules.

C. Causality modeling in CPS

Several approaches focus on modeling systems dynamics with Bayesian networks. In [16], the authors learn a CBN-based model of an environment using observation and well-selected interventions. In [8], authors restrict interventions to certain variables. The approach has low scalability, and restricting interventions raises accuracy issues. The authors mitigate these issues in [9]. First, they use correlation analysis to avoid unnecessary intervention testing. Second, they employ aggregated variables (assuming these are available) to allow dividing the correlation graph into sub-graphs and pruning variable associations. These improvements enhance the scalability of the method. Finally, they incorporate indirect interventions to improve the accuracy of the learned graph.

As these methods rely on the causal relations of the entire system, they face more significant scalability, adaptability, and interpretability challenges than our proposal, which only focuses on finding the cause of one current fact. This makes CIRCE better suited for providing quick, context-sensitive answers to runtime questions even for large systems.

D. Finding a cause

A but-for cause [17], [18] can be defined as a counterfactual, i.e. "C is the cause of T" can be expressed as "but for C, T would not have occurred". Formalizing this notion requires a structural causal model (SCM) denoted as $\mathcal{M} = \langle X, U, F \rangle$. Here, $X = \{X_1, ..., X_N\}$ represents endogenous variables, $U = \{U_1, ..., U_N\}$ stands for exogenous noise, and $F = \{F_1, ..., F_N\}$ denotes structural assignments or causal mechanisms.

In the context of an SCM $\mathcal{M}$ and a given context x, the truth of a predicate P about the system is expressed as $(\mathcal{M}, x) \vDash P$. The forced assignation of variables $Y \subset X$ to values $y \in \mathbb{R}^{|Y|}$ is denoted as $Y \leftarrow y$. The truth value of a predicate P under the assignation of values y to variables Y is represented as $(\mathcal{M}, x) \vDash [Y \leftarrow y]P$.

The but-for cause C of a target predicate T is defined as the subset of X that satisfies the three conditions:

- $(\mathcal{M}, x) \vDash (C = c)$ and $(\mathcal{M}, x) \vDash T$
- $\exists c'$ such that $(\mathcal{M}, x) \vDash [C \leftarrow c']\neg T$.
- no subset of C satisfies the previous conditions

As mentioned in [19], this definition of cause lacks specifications on the causal behavior when interventions are performed on the variables that are not part of the cause. Notably, [19] proposes to consider the notion of causal sufficiency when defining a cause. A set of variables C is causally sufficient to a predicate T if changing any value of variables not in C still implies T. In an SCM $\mathcal{M}$ and a context x, C is causally sufficient for T if for any settings y' of variables outside C, $(M, x) \vDash [C \leftarrow c, Y \leftarrow y']T$.

While cause generation is computationally more efficient than computing a full causal network, it remains a challenging task. Finding the exact cause would require testing all possible values of all sets of values. Approximating the cause of a target predicate T can be seen as finding the conditions on X that imply T after computing effects with the causal mechanisms. Thankfully, significant effort has been produced in the field of post hoc local black-box XAI to find conditions on input variables that imply a given output of a function.

IV. Proposed solution

In this section, we explain our proposed solution. We first formalize the problem that we aim to address IV-A, then we describe how we adapt LIME to the CPS context IV-B, and finally we explain how we generate a cause from the obtained decision tree IV-C.

A. Problem formalization

We model each CPS state as a set of variables, formalized as an N-dimensional vector, which evolves over time to follow state changes. We consider a set of state observations $\mathcal{S} = \{s^t \in \mathbb{R}^N | t \in [\![1, k]\!]\}$, where s^t is the observed system state at time t, and k the number of observations. As explained in section I, we assume the availability of a transition function $f : s^t \mapsto s^{t+1}$ (e.g. via simulation).

Our explanation method answers a question "Why $T(s^{t_0})$?" (i.e. why did the system's state s^{t_0} verify the properties expressed by predicate T?) by "Because $C(s^{t_0-1})$" (i.e. because the properties expressed by predicate C held in the system's previous state s^{t_0-1}; and if these properties hadn't been true in the previous state then we wouldn't have had $T(s^{t_0})$). Each time a question is asked, we consider a SCM where the set of endogenous variables is composed of the observed variables of s^{t_0} and s^{t_0-1}. Hence, half of the variables (those taken from s^{t_0}) are computed based on exogenous consideration, and the other (those taken from s^{t_0-1}) are computed based on the transition function f.

We use predicates that are conjunctive clauses $P = P_1 \wedge ... \wedge P_m$, where each term $P_i = X_i \mathcal{P}_i \theta_i$ is a first-order logic formula, with X_i a system variable, θ_i a threshold value, and $\mathcal{P}_i \in \{\leq, >\}$. For instance, predicate $P = Tmp > 15$ is true when applied to state s^t if $Tmp > 15$ at t, Tmp being the system's temperature variable. We can compose the transition function f with a predicate T and obtain the function $T \circ f$ that applies T to the state right after the one on which the function is applied.

To determine C, we train a decision tree that approximates the function $T \circ f$ around the data point s^{t_0-1} following the LIME method. We build on LIME to sample the training data and train the decision tree. We then use this decision tree to derive a cause predicate C.

B. Adapting LIME to search for causes in CPS

Based on LIME [1], we generate a training dataset L consisting of pairs of perturbed system states x^i and their labels $y^{(i)}$: $L = \{(x^i, y^{(i)}) \in \mathbb{R}^N \times \{0, 1\} | i \in [\![1, n']\!]\}$, where N is the number of variables of each system state, and n' the number of samples in the dataset. This dataset will be used to train the surrogate model (i.e. the decision tree). Each perturbed state x^i is obtained by modifying several variable values of state s^{t_0-1} – i.e. the state observed before the question was asked at t_0.

We employ the sampling method used in SHAP [13]. On the one hand, we draw a random number of variables $J \subset [\![1, N]\!]$ to be perturbed. On the other hand, we draw a state $s' \in \mathcal{D}$, where $\mathcal{D}$ is a dataset of states generated by running the simulation. While $\mathcal{D}$ is reused across questions, s' and J are drawn for each sample. We obtain a perturbed data point x^i where $\forall j \in J : x^i_j = s'_j$ and $\forall j \notin J : x^i_j = s^{t_0-1}_j$. We label our perturbed data points x^i by applying one step of the simulation and by computing our target predicate, i.e. $\forall i \in [\![1, n']\!] : y^{(i)} \coloneqq T(f(x^i))$. To get as many positive samples ($y^{(i)} = 1$) as negative ones ($y^{(i)} = 0$), we generate samples until we have $n'/2$ positive instances and $n'/2$ negative ones, and we discard the remaining ones.

A shortcoming of this sampling technique is that it does not consider the conditional distribution of the perturbed features given the original ones. This implies that some samples might be associated with impossible states (we plan to address this issue in future work).

Finally, we train a decision tree classifier on this generated dataset. Each training data sample x^i is weighted with a distance kernel that gives less importance to states that are distant from the original state.

C. Expressing the cause

According to the classic definition [20], a 'but-for cause' of a predicate T, in a context s^{t_0-1}, is the minimal set of variables $X_C \in \mathcal{X}$ that can be set to a counterfactual value c' such that setting values of s^{t_0-1} to c' implies $\neg T$. We propose the following intuition for an approximated but-for cause. In

the context s^{t_0-1}, an approximated but-for cause of T is the minimal set of variables $X_C \in \mathcal{X}$ that contains a region R such that when sampling a random state s' close to s^{t_0-1} and values $c' \in R$, setting values of s' to c' very probably implies $\neg T$. We propose to consider as a cause the predicate C that describes the region R on variables X_C.

According to [19], a 'sufficient but-for cause' of T in a context s^{t_0-1} is a set of variables X_C that is a but-for cause of T and that is sufficient for T, i.e. changing any values of s^{t_0-1} from a variable outside X_C still implies T. Therefore, we will say that X_C is approximately necessary to T in context s^{t_0-1} if there is a region R defined on variables X_C such that, when sampling a state in R, changing any values of variables outside X_C very probably implies T. We also propose to consider as an approximate sufficient but for cause the predicate C that defines the region R on variables X_C.

To generate a candidate for an approximated sufficient but-for cause, we must exhibit a region R that satisfies the previous conditions. We derive such a region from the surrogate decision tree. Figure 3 exemplifies a decision tree. Each node that is not a leaf defines a split of the training dataset using a variable and a threshold. The values for this variable in the data that lie below this threshold are associated with the left child and the rest with the right child. We can compute a probability of T for each node as the ratio of positive-to-negative $y^{(i)}$ associated with the x^i in this node. The "class" of this node is then the most probable between either T (a majority of $y^{(i)} = 1$) or $\neg T$ (a majority of $y^{(i)} = 0$). We also compute an impurity score, which describes how much the distribution of $y^{(i)}$ is balanced. In figure 3, the nodes with a high impurity score are not associated with any class. For instance, node 2 has an impurity score of 0.45. If the score used is the Gini impurity score (a commonly used one), then it corresponds to $\mathbb{P}(y = 0) \approx .343$ or $\mathbb{P}(y = 1) \approx .343$, which does not reflect a given class convincingly. To build a decision tree, each node recursively selects the split that allows for the lowest impurity in its children.

To generate a candidate for the cause predicate C, we iterate down the tree following s^{t_0-1}. We stop at the first node with the two following conditions. First, all descendants have a high impurity or a positive majority class. Second, an ancestor node has a sibling with low impurity score and negative class. We therefore introduce two hyperparameters for our method: a high impurity threshold and a low impurity threshold.

For instance, in the tree from figure 3, an approximated minimal but-for cause is shaded, and an approximated sufficient but-for cause is outlined in green ($X_1 \leq \theta_1 \wedge X_2 \leq \theta_2 \wedge X_3 > \theta_3$). The region defined by node 4 is a good approximation of a but-for cause since sampling a state from outside this region, i.e. in the one described by node 5, will yield a high probability of implying $\neg T$. The region defined by node 7 is a good approximation of a sufficient but-for cause since sampling a state from outside this region, i.e. in the one described by nodes 5 or 6, will yield a high probability of implying $\neg T$, while sampling from within this region (nodes 7, 8 and 9) always yields a high probability of T.

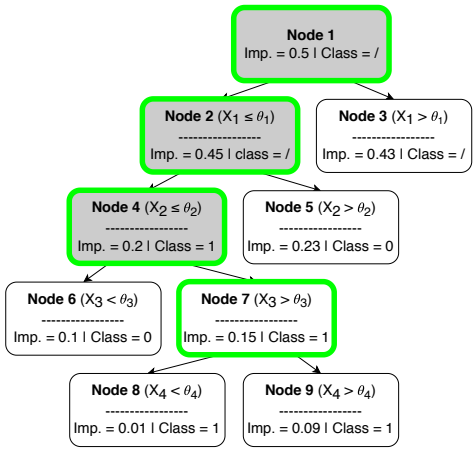

Fig. 3. Cause selection process for an exemplary decision tree. The irrelevant nodes have not been developed. The nodes with green outlines constitute a sufficient cause and those with darker backgrounds constitute a regular but-for cause.

TABLE I
SIMULATION RULES (CF. VARIABLES IN SUBSEC. V-A)

Rule ID	Precondition	Effect
R_{IT1}	$WS = 1$	$IT = OT$
R_{IT2}	$WS = 0 \wedge HS = 1$	$IT = 20$
R_{IT3}	$WS = 0 \wedge HS = 0$	$IT = \text{mean}(IT_{nei})$
R_{IT*}	$d(TH, L) < 2.5$	$IT = 35$
R_{HS1}	$IT \leq 20$	$HS = 1$
R_{HS2}	$IT > 20$	$HS = 0$
R_{LS1}	$N_u > 0$	$LS = 1$
R_{LS2}	$N_u = 0$	$LS = 0$
R_{WS1}	$N_u > 0 \wedge IT < 5$	$WS = 0$
R_{WS2}	$N_u > 0 \wedge IT > 25$	$WS = 1$
R_{WS3}	$N_u = 0 \vee 5 \leq IT \leq 25$	$WS = WS$

The user can restart the process if the output does not satisfy them. This could happen if the direct cause is still surprising or too obvious to the user. The user could then ask for the cause of the original target predicate again or of the system's output, at time $t_0 - 1$.

V. EXPERIMENTS

We tested our approach via several smart-home scenarios developed within a rule-based simulation. While the actual simulation is a quite simplified version of a realistic smart home, we can build an SCM on it to illustrate our proposal and highlight its potential accuracy and scalability.

A. Simulations

A house is represented as a grid of $R \times R$ rooms. Each room includes: a thermometer, with position TH (coordinates TH_x and TH_y), giving the inside temperature IT; a heater, with state HS (on/off); a window, with state WS (open/close); and a light, with position L (at coordinates L_x and L_y) and state LS (on/off). Each room monitors the number of users N_u within that room. A shared variable gives the outside temperature OT. We simulate a neighborhood with M houses.

Each room monitors 9 variables. Hence, the total number of variables in the system is $9R^2M + 1$.

At each timestep, we compute the value of each variable using the rules in Table I. For instance, Rule R_{IT*} models an anomaly that occurs when the thermometer is too close to the light – causing temperature measurements to feature high, non-plausible values. This rule only affects the measured temperature. The system simulation keeps track of the true temperature using the other rules. The threshold for applying R_{IT*} is a distance $d(TH, L) < 2.5$. Values of TH and L have a range between 0 and 10. In rule R_{IT3}, mean(IT_{nei}) refers to the average temperature of the rooms that are neighbors of the current room. The outside temperature follows a sine function of time with a minimum of 5 at and a maximum of 20 with a sine period of 24, i.e. $OT = 12.5 + 7.5 \times \sin(\frac{2\pi t}{24} + 24)$, which generates a 24h cycle. The period between measures is one hour. To make the simulation less deterministic, we add various random events to the deterministic rules above (e.g. users changing rooms, changing light or thermometer position, etc.)

B. Questions

To evaluate our method, we identified four situations that may occur in the simulation. In situation 1, a user wants to understand why the heater is on. They first ask the system about the heater state. The cause of the heater being on is the inside temperature of the room being lower than the heater threshold at the last timestep. Though this is a correct answer from a causal point of view, it does not satisfy the user. They ask for a cause of the temperature being lower than this threshold. The cause is now that the window was open, and the outside temperature was lower than this threshold. Situation 2 exemplifies a less apparent cause concerning the temperature inside. The user wonders why the room is chill and asks why its temperature is lower than 15. This time, the cause is that the heater was off and that some neighboring rooms were cold as well. In situation 3, the user wants to know why the window is closed at a timestep t_0. As it was already closed at $t_0 - 1$, the method answers that the window remained closed instead of providing an event that caused the window's closure. The user therefore asks the same question again about one timestep earlier, i.e. "why was the window close at timestep $t_0 - 1$?". Finally, situation 4 evaluates CIRCE's ability to find causes of abnormal behavior of the system, illustrated by the use of rule R_{IT*} in the simulation. The user asks why the measured temperature is higher than a given threshold. The expected cause is the proximity of the light and the thermometer.

The six questions asked in these four situations are reported in table II as well as the expected answers (from a causal point of view) and the rule used to compute the variable whose value is questioned.

C. Evaluation

To evaluate our method, we simulated 2000 steps and iterated through the data to find each timestep t where a situation holds (i.e. the questioned fact holds at time t, the expected cause holds at time $t - 1$ and the appropriate rule is executed between time t and time $t - 1$). At each of these timesteps, we apply CIRCE and compare its output to the expected cause. We first conduct a qualitative evaluation by selecting one timestep for each of the four situations. We report and analyze CIRCE's outputs for each of the 6 questions. We then quantitatively test the performance of CIRCE by running the method on many timesteps and automatically computing a score for the similarity between the expected cause and CIRCE's output. We then identify 10 parameters that can influence the method and test how the performance evolves when changing the values of these parameters. Finally, we evaluate CIRCE temporal complexity by comparing the run time of the method when some of these parameters vary.

To measure the performance of CIRCE, we select 200 timesteps for each situation and compute a score that reflects the quality of CIRCE's outputs. We only used Q1.1, Q1.2, Q3.1, and Q3.2 for these experiments. We selected these questions because the expected causes can easily be expressed as a set of clauses of the form $X\mathcal{P}\theta$. We compare the generated clauses with the expected ones using the F1-score, a measure that compares a predicted set of elements A to an expected set of elements B. It is computed as the harmonic average of the recall measure and the precision measure, i.e. $F1 = 2\frac{P \times R}{P+R}$. The recall computes the ratio of predicted elements that are correct, i.e. $R = \frac{|A \cap B|}{|A|}$. The precision computes the ratio of expected elements that are found, i.e. $P = \frac{|A \cap B|}{|B|}$. To match two clauses, we compare the variables by matching the house, the room, and the measure (e.g. inside temperature). We then match the comparator and check that the generated threshold has a relative error to the expected one of less than 10%.

The 10 parameters whose effects are evaluated are the number of houses, the number of rules, the type of label balancing strategy for the training dataset, the size of the training dataset, the type of impurity measure, the maximal depth of the surrogate decision tree, the distance measure, the distance kernel, the low impurity threshold, and the high impurity threshold.

To vary the number of rules, we implemented 4 additional rules: Rule R_{HS3} states that if the window is open, the heater switches off. Rule R_{HS4} states that if the heaters of the neighboring rooms are all on, the heater switches off. Rule R_{WS3} states that if the windows of the neighboring rooms are all open, the window closes. Rule R_{HS5} states that if there is no user in any neighboring room, the heater switches off. When we let the nulber of rules vary, we have three prossible settings. The original one with the 11 rules from Table I. We then have a setting with 13 rules, including the 11 original ones, R_{HS3} and R_{HS4}. Finally, the setting with 15 rules include the 11 original ones and the four additional ones.

We used a version of our sampling method where we do not balance positive and negative samples in the surrogate decision tree training dataset. The initial balancing method is referred to as "sample" and the modified one as "none". We

TABLE II
THE FOUR QUESTIONS USED TO ILLUSTRATE OUR APPROACH, THE EXPECTED CAUSE, AND THE RULES APPLIED TO OBTAIN THE OBSERVED VALUE.

Scenario	Question	Expected cause	Rule ID
Situation 1	Q1.1: Why is the heater on?	The temperature was lower than 20.	R_{HS1}
	Q1.2: Why is the temperature lower than 20?	The window was open and the outside temperature was lower than 20.	R_{IT2}
Situation 2	Q2: Why is the temperature lower than 15?	The heater was off and some neighbor rooms had a low temperature.	R_{IT3}
Situation 3	Q3.1: Why is the window closed?	It was closed and the temperature was lower than 25	R_{WS3}
	Q3.2: Why is the window closed?	A user was in the room and the temperature was lower than 5.	R_{WS1}
Situation 4	Q4: Why is the temperature higher than 30?	The light was close to the thermometer.	R_{IT*}

used 3 different impurity measures, namely the Gini impurity measure, the entropy measure, and the log loss measure. We tested a few different values for the max depth of the tree, which results from a stopping criterion when constructing the surrogate decision tree. The absence of a stopping criterion based on tree depth is called value "-1". We tested 3 distance measures to evaluate the dissimilarity of the perturbed data to the original state when generating interventions. The L0 distance $d_{L0}(a,b) = \sum_{i=0}^{N} \mathbb{I}(a_i \neq b_i)$ computes the number of different components. The L1 distance $d_{L1}(a,b) = \sum_{i=0}^{N} |a_i - b_i|$ computes the absolute differences between components. The L2 (or Euclidean) distance $d_{L2}(a,b) = \sum_{i=0}^{N} (a_i - b_i)^2$ computes the squared differences between components, hence giving more weight to larger discrepancies. We tested several functions as a distance kernel for weighting the instances while training the model. The constant kernel refers to the function $x \mapsto 1$, the inverse to $x \mapsto \frac{1}{x}$, the inverse_exp to $x \mapsto e^{-x}$, the inverse_sqrt to $x \mapsto \frac{1}{\sqrt{x}}$, the linear to $x \mapsto x$, the inverse to $x \mapsto \frac{1}{x}$, and the squared inverse to $x \mapsto \frac{1}{x^2}$.

D. Experimental setup

We implemented our simulation in Python and the decision trees using the `sklearn` library[1]. Since this implementation only handles numerical values, it does not take categorical values into account. Our overall approach could yet do so by using another implementation of decision trees. We use boolean values by converting them into floats.

As base values for our parameters, we use 3 houses, a sample size of 2000, the inverse kernel, the Gini impurity index as an imurity measure, a maximum depth of 8, an Euclidean distance, a minimum impurity threshold of 0.3, and a maximum impurity threshold of 0.5. The quantitative evaluation is done using these values. When we let one parameter vary, all others are set to their reference values.

VI. RESULTS

A. Qualitative results

Table III shows the causes generated for each of the four situations. CIRCE was able to generate the expected cause for questions Q1.1, Q1.2, Q2, Q3.1, and Q3.2. CIRCE still succeeds in approximating the answer to question Q4 by selecting relevant dimensions and thresholds, i.e. both the light x coordinate and the thermometer x coordinate have low values.

[1]https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

TABLE III
CAUSES GENERATED BY OUR SYSTEM

Question	CIRCE output
Q1.1	Room 2,0 inside_T $\leq$ 19.99
Q1.2	Room 2,0 window open & outside_T $\leq$ 12.02
Q2	Room 2,1 inside_T $\leq$ 9.427 & room 2,2 heater off & room 1,1 inside_T $\leq$ 15.66
Q3.1	Room 1,0 window closed & room 1,0 inside_T $\leq$ 28.61
Q3.2	Room 1,0 number of users $>$ 0.5 & room 1,0 inside_T $\leq$ 10.78
Q4	Room 0,2 light on & room 0,2 thermometer x $\leq$ 3.5 & room 0,2 light x $\leq$ 1.5

As in the expected cause of Q2, the system did not mention part of the precondition of rule R_{IT3}, i.e. $WS = 0$. This is because the outside temperature is low. With the window open, we would still have a cold inside temperature. Hence, despite being part of the precondition, $WS = 0$ is not part of the cause of the inside temperature being cold. This illustrates that our approach does not aim at detecting preconditions of rules but contextual causes.

B. Quantitative results

We ran CIRCE on 200 of the selected timesteps for our test questions. We obtain an F1 score of 100±0% for Q1.1, an F1-score of 85±30% for Q1.2, an F1-score of 74±22% for Q3.1, and an F1-score of 74±37% for Q3.2. These results indicate that CIRCE mostly generates good answers but lacks stability, illustrated by an important standard deviation. However, the standard deviation might be amplified by the fact that we compute F1 scores between small sets. Even though most examples are correct, once one output differs from the expected result, it will get a low F1 score.

Figure 4 shows the mean and std of the F1 score obtained when varying some parameters. For almost any setting, question Q1.1 yields perfect results. Most of the time, question Q1.2 yields around 80% of F1 score. Questions Q3.1 and Q3.2 are often around 75% F1-score. In most experiments, the Q3.1 F1-score shows an important sensibility to the parameter variations.

The number of houses has little effect on the F1-score. This highlights the scalability of our approach in terms of the quality of the generated candidate causes. The method does not perform well when the number of samples of the training dataset for the surrogate decision tree is too small.

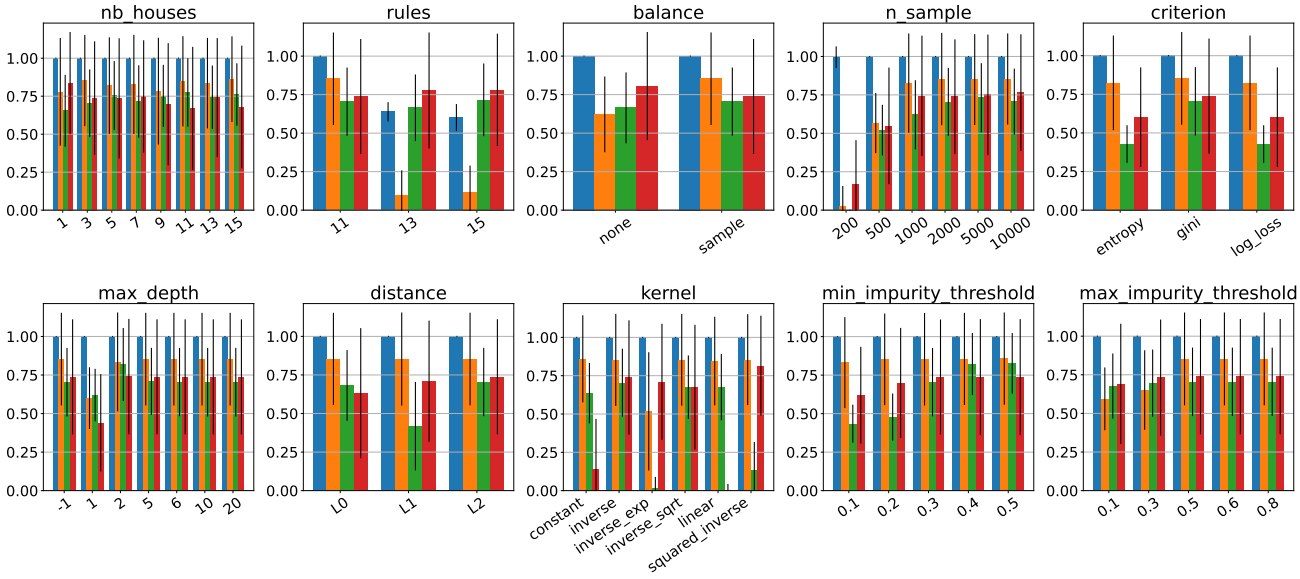

Fig. 4. Performances when varying several parameters of our approach and simulation. The reported measures are the mean and standard deviation of the F1 score. For each value, Q1.1 is the first bar in blue, Q1.2 the second in orange, Q2.1 the third in green, and Q2.2 the fourth in red.

However, the F1-score stabilizes around a size of 1000 samples, which suggests that the method does not require an enormous amount of data. Similarly, the performance is not influenced by additional rules that do not impact the variables of interest, as shown by the F1-score of Q3.1 and Q3.2 when adding the new rules. The reported performance on question Q1.1 deteriorates because we kept the same expected cause while the actual cause contains an additional clause (due to rule R_{HS3}). When looking at CIRCE's outputs for this question, we can see that the additional clause is often correctly generated. The performance on question Q1.2 also deteriorates. This is because the rule R_{HS3} doesn't allow the heater to be on when the window is open. Therefore, situation 1 is not possible when this rule is enabled. CIRCE naturally generates outputs that are far from the expected cause, since it doesn't hold in this case. Overall, this suggests good scalability in terms of the F1-score when augmenting the complexity of the simulation, even though more relevant situations might be useful for better evaluation of this parameter.

Results indicate that using the Gini impurity index, the L2 distance, and balancing the labels of the training dataset yields the best results. A high value of the low impurity threshold and a value above 0.1 of the high impurity threshold are to be preferred. The maximal depth of the tree has little impact on the performance (as long as above 1) which suggests that the cause selection process does not overfit the data as decision trees tend to do, i.e. allowing for a more complex decision tree does not yield more complex candidate causes.

Finally, varying the distance kernel yields highly diverse results. As for many parameters, Q1.1 has high performances regardless of the kernel. As we would expect, kernels that give more importance to instances close to the original one benefit question Q3.2. However, we can see that a constant kernel or even kernels that give more importance to instances far from the original one have little effect on Q1.1 and a positive effect on Q3.1. This is due to the sampling methodology and will be discussed in more detail in section VII.

C. Scalability

To evaluate the temporal scalability of our approach, we report, in figure 5, the training time of the surrogate decision tree when varying some parameters. For most parameters, we report little to no effect. For instance, the balancing of the dataset labels, the nature of the impurity measure, and the type of distance have unnoticeable effects. The number of rules impacts question Q1.1, probably because it makes the model extend the tree deeper. Similarly, there is an important drop when using lower values of the maximal depth. This is due to the tree not being extended. Finally, the training time also drops when using an inverse exponential kernel. This is due to the tree not extending enough, as highlighted by the low performance for questions Q1.2 and Q3.1, and the high performance for question Q3.2.

These experiments show a noticeable linear relationship between the number of houses (hence with the number of dimensions considered by our approach) and the training time of the surrogate decision tree. There is a similar linear relationship between the size of the training dataset and the training time. However, as we can see in figure 4, the quality of our approach reaches a limit with a dataset size of around 1000. Hence, these experiments reveal a linear time scalability of our approach.

We focused on reporting the training time of the model instead of the overall time taken by the method because most of the time is consumed by querying the simulation to label the training dataset. The training time is not especially relevant

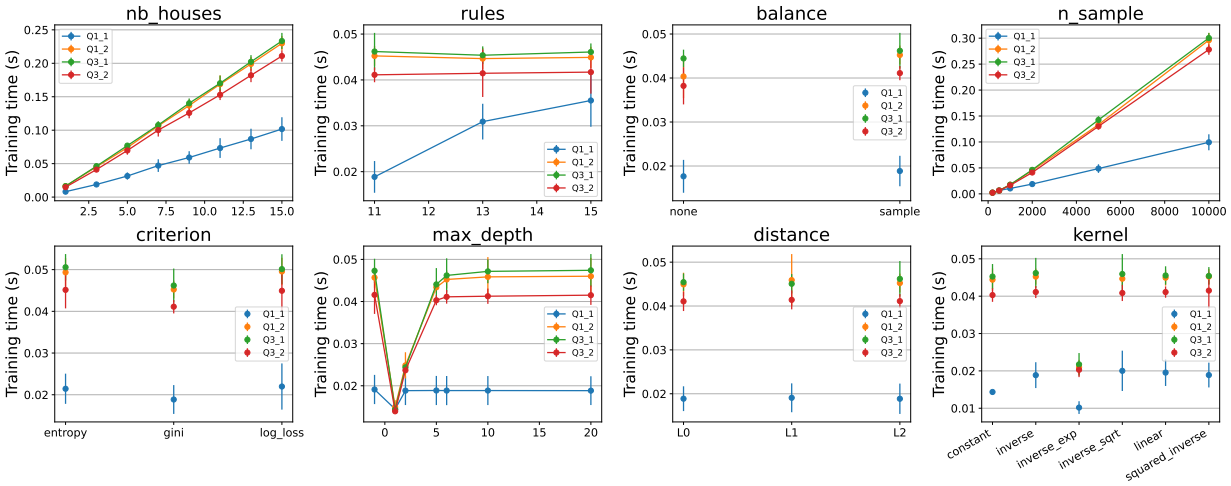

Fig. 5. Training time of the surrogate decision tree when varying several parameters of our approach and simulation. The reported measures are the mean and standard deviation of the training time in seconds.

to the efficiency or scalability of the approach. It rather reveals that our approach requires an efficient implementation of the function f that simulates the dynamics of the environment.

VII. Discussion

A. Comparison with CBN-based approach

For each question, we obtained a cause as a predicate based on the system variables. This predicate has a limited number of clauses. It is only made of system variables, numbers, comparators, and the logical connector "and". They can be easily transcribed in natural language. Most CBN-based approaches focus on methodologies to build the graph, rather than answering a user question based on the obtained graph. However, it is usually implied that a cause can be extracted as the set of ancestors of the target variable in the graph. At best, the graph may include an approximate regression of the probability of the target variable given a set of values for its ancestors. For instance, we could expect a CBN-based approach to answer question Q1.2 with the set of variables $\{IT, OT, WS, HS, TH, L\}$.

Moreover, building a CBN in an environment without any information on its structure would typically scale as the square of the number of variables [9]. Our approach scales linearly as shown in section VI-C.

B. Limits of the sampling methods

Our approach for approximating a cause consists of sampling perturbations of a state and labeling these perturbations with the output of $T \circ f$. We sampled the perturbations by selecting dimensions to perturb and a point from the simulation dataset to instantiate the perturbed dimensions. Both these operations are done with a uniform distribution. The quality of the approximated cause could be degraded if the obtained dataset is highly unbalanced. Consequently, our approach requires causal predicates composed of a few dimensions and intervals that do not contain all frequent values. In sum, uniform sampling should yield a sufficient number of counterfactuals.

This consideration is probably the reason for the instability of question Q3.1. The expected cause of question Q3.1 is a closed window and a temperature lower than 25. But in practice, the temperature is rarely higher than 25. The dataset of perturbations might contain too few counterfactuals due to the temperature and probably too many counterfactuals due to noise. When the number of houses rose, so did the F1-score of Q3.1. This is even more noticeable when looking at the precision score. This implies an average decrease in incorrect clauses generated as the number of dimensions increases. This behavior is probably due to spurious correlations in the training dataset of the decision tree. When we increase the number of dimensions, these spurious correlations have less chance of arising. We do not detect spurious correlations in the simulation as causes because our datasets are based on interventions, but we suffer from spurious correlations introduced in the intervention dataset itself.

The scarcity of counterfactuals due to the expected cause has another consequence. When we use a steep kernel (i.e. a kernel that gives way more importance to data close to the reference instance), question Q3.1 performance unexpectedly drops. Using kernels with opposite behaviors yields even better results for Q3.2. Indeed, our default distance is Euclidean and gives the same importance to each dimension. Since the range of the temperature variable is wider, it gets more weight. As noted in the previous paragraph, counterfactuals due to high temperatures (e.g. a true temperature of 15 and a perturbed temperature of 28) may not be common enough. When we use a steep kernel, we give them even less importance. This is why steep kernels are harmful to situations similar to question Q3.1.

C. Scope of interpretability

We focused on explanations that reveal causal relationships in the system. We did not consider the relevance of the information given to the user. Future work will explore additional functions for filtering irrelevant causes [10], [21]. Moreover, these causal relationships are discovered between the variables that the system monitors. For our approach to be interpretable, we need these variables to be understandable by the users. If the system monitors fine-grained variables, the user might not understand their meaning and not find help in our explanations. Further, a system monitored with too granular variables might imply causes with a high number of variables involved. On the one hand, CIRCE would be less interpretable, due to the expected cause being less interpretable itself. On the other hand, this might raise performance issues for the decision tree training that would probably require more training data.

VIII. Conclusion

We presented a novel way to conceptualize causes in the context of autonomic CPS: the approximate sufficient but-for cause. We proposed a novel method for generating candidate causes. The method overcomes many limitations of the classic causality-based approach for causal explanations. First, the generated candidate causes have good interpretability. Second, the computation scales linearly with the number of variables in the system, allowing the possibility of online cause generation. We provide a method based on XAI to generate such causes.

The approach has some limitations. Though many of the assumptions made in this work are usual, they are not realistic. We supposed a Markovian system where all variables depend only on the previous state of the system. This makes our approach not well suited for finding long-term causes. The need for a simulation is demanding and limits CIRCE's ability to reflect real-world causal relationships. We also suppose a user query can be used as a condition on the system variables, which could be hard to do in practice. Although accurate most of the time, CIRCE tends to be unstable, especially when counterfactuals correspond to rare values of the observed variables.

We presented a set of experiments designed to illustrate CIRCE's abilities and evaluate its scalability. The simulation used in these experiments remains relatively simple. A more complex use case could be beneficial for further evaluation.

In future work, we will address some of these limitations. First, we could consider various time scales. We will use a simulation in nearly continuous time and include long-term dependencies. When applying our method, we will consider different time spans to sample previous steps and run the simulation over the considered period before applying the target predicate. We will search for solutions to improve our sampling methodology and mitigate the issues discussed in section VII. Finally, we may mitigate the instability by training several surrogate decision trees and by selecting a cause with voting strategies.

References

[1] M. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," in *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, Jun. 2016, pp. 97–101.

[2] E. Lee, "Cyber physical systems: Design challenges," in *Proc. 11th IEEE international symposium on object and component-oriented real-time distributed computing (ISORC)*, 2008, pp. 363–369.

[3] I. Chun, J. Park, W. Kim, W. Kang, and S. P. H. Lee, "Autonomic computing technologies for cyber-physical systems," in *Proc. The 12th International Conference on Advanced Communication Technology (ICACT)*, vol. 2, 2010, pp. 1009–1014.

[4] J. Kramer and J. Magee, "A rigorous architectural approach to adaptive software engineering," *Journal of Computer Science and Technology*, vol. 24, no. 2, pp. 183–188, 2009.

[5] M. Blumreiter, J. Greenyer, G. F. J. C. V. Klös, M. Schwammberger, A. V. C. Sommer, and A. Wortmann, "Towards self-explainable cyber-physical systems," in *Proc. 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 2019, pp. 543–548.

[6] M. Camilli, R. Mirandola, and P. Scandurra, "Xsa: explainable self-adaptation," in *Proc. 37th IEEE/ACM International Conference on Automated Software Engineering*, 2023, pp. 1–5.

[7] R. Sukkerd, R. Simmons, and D. Garlan, "Towards explainable multi-objective probabilistic planning," in *Proc. 4th International Workshop on Software Engineering for Smart Cyber-Physical Systems*, 2018, p. 19–25.

[8] K. Fadiga, É. Houzé, A. Diaconescu, and J.-L. Dessalles, "To do or not to do: finding causal relations in smart homes," in *Proc. IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, 2021, pp. 110–119.

[9] K. Fadiga, E. Houzé, A. Diaconescu, and J.-L. Dessalles, "Improving causal learning scalability and performance using aggregates and interventions," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 18, no. 3, pp. 1–18, 2023.

[10] E. Houzé, J.-L. Dessalles, A. Diaconescu, D. Menga, and M. Schumann, "A decentralized explanatory system for intelligent cyber-physical systems," in *Proc. SAI Intelligent Systems Conference*, 2022, pp. 719–738.

[11] L. Herbold, M. Sadeghi, and A. Vogelsang, "Generating context-aware contrastive explanations in rule-based systems," *arXiv preprint arXiv:2402.13000*, 2024.

[12] D. Minh, H. X. Wang, Y. F. Li, and T. N. Nguyen, "Explainable artificial intelligence: a comprehensive review," *Artificial Intelligence Review*, pp. 1–66, 2022.

[13] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. 31st Conference on Neural Information Processing Systems (NIPS 2017)*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, 2017, pp. 4765–4774.

[14] L.-S. Shapley, "17. a value for n-person games," in *Contributions to the Theory of Games (AM-28), Volume II*. Princeton University Press, 2016, pp. 307–318.

[15] F. Ziesche, V. Klös, and S. Glesner, "Anomaly detection and classification to enable self-explainability of autonomous systems," in *Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021, pp. 1304–1309.

[16] S. Megancka, P. Leraya, and B. Manderick, "Learning causal bayesian networks from observations and experiments: A decision theoretic approach," in *Proc. International Conference on Modeling Decisions for Artificial Intelligence*, 2006, pp. 58–69.

[17] J. Pearl, *Causality*. Cambridge university press, 2009.

[18] J. Peters, D. Janzing, and B. Schölkopf, *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

[19] S. Beckers, "Causal sufficiency and actual causation," *Journal of Philosophical Logic*, vol. 50, no. 6, pp. 1341–1374, 2021.

[20] J. Y. Halpern and J. Pearl, "Causes and explanations: A structural-model approach. part i: Causes," *The British journal for the philosophy of science*, 2005.

[21] M. Sadeghi, L. Herbold, M. Unterbusch, and A. Vogelsang, "Smartex: A framework for generating user-centric explanations in smart environments," *arXiv preprint arXiv:2402.13024*, 2024.