

Automated Self-Explanation of Expected versus Perceived Behavior for Interacting Digital Systems

Mohammad Alkhiyami¹, Gianluca Martino², and Goerschwin Fey³

Hamburg University of Technology, Hamburg, Germany

¹mohammad.alkhiyami@tuhh.de, ²gianluca.martino@tuhh.de ³goerschwin.fey@tuhh.de

Abstract—Modern interacting digital systems are becoming increasingly complex, making it difficult to ensure their actual behavior aligns with design-time expectations, particularly in uncertain or dynamic environments, even when specifications are correct. This misalignment affects system scalability, reliability, and increases maintenance costs.

We introduce a conceptual framework for identifying and self-explaining mismatches between expected and observed system behavior, together with an algorithm that generates explanations and case studies that apply the conceptual framework for explanation generation in an interacting digital systems setting.

Index Terms—Formal methods, model checking, action-oriented explanation, logic and verification, knowledge representation and reasoning, and cyber-physical systems.

I. INTRODUCTION

Modern interacting digital systems exhibit increasing complexity. Due to their complexity, it becomes difficult for systems and operators to comprehend and ensure their actual behavior aligns with design-time expectations, especially in uncertain environments. As a result, future interacting digital systems need to incorporate mechanisms to justify their actions for users to increase trust [1]. Explanations to other systems are also needed to optimize the interaction. These explanations provide valuable insights for preventing similar unpredictable behaviors in the future or for updating the system’s knowledge base to enhance reliability and user confidence [2], [3].

The core idea is to have an abstracted model of expected behavior aligned with the real system, so deviations between these show the need for triggering explanations. Prior work has introduced similar concepts, e.g. under the term self-explanation in systems of systems [4], providing a conceptual foundation for enabling one system to clarify its behavior to another. Building on this foundation, our work advances the idea by contributing a formal algorithmic framework that generates explanations for mismatches between expected and perceived behavior, together with case studies that demonstrate the applicability and scalability of this approach.

An example of interacting digital systems is a wind park and a wind turbine. The wind turbine’s expected behavior is to follow the operator’s command in the wind park. However, inconsistent behaviors could occur due to many factors that might affect the turbine, e.g., local conditions cause a wind turbine to adjust blades overriding a command. We suppose

This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) project no. 513623283 as part of the Research Training Group CAUSE.

that the turbine’s expected behavior is bound to a specific set of states and the transition relation between these states is defined at design time forming predefined execution paths. These paths are subject to various conditions induced onto the overall system and its environment, which we refer to as assumptions. Modeling assumptions and systems by temporal constraints and Mealy machines, we leverage model checking and Satisfiability Modulo Theories (SMT) to explain deviations between perceived and expected execution. An explanation in this conceptual framework is defined as a subset of assumptions preventing the expected and perceived behaviors of interacting digital systems from matching.

We introduce the conceptual framework through examples and suggest an algorithm that can self-explain mismatches in interacting digital systems’ behavior.

The resulting explanations are actionable, meaning they enable the addressee to take appropriate actions by focusing on relevant conditions [5], [6]; causal, in the sense that they identify the reasons behind specific outcomes and help understand how and why similar effects might arise under different conditions [7]; and counterfactual, as they illustrate what could have been done differently in a given situation to produce a different outcome [8].

The contributions of our work are:

- A modeling approach for expected and actual behavior without needing complete specifications.
- A reduction of self-explanation to bounded model checking and SMT solving.
- Case studies illustrating the versatility and investigating the scalability of the approach.

The rest of the paper is organized as follows: Section II discusses related work. Section III describes our modeling approach. Section IV describes the conceptual framework. Section V describes the algorithm. Section VI introduces case studies. Section VII reports experimental results, and Section VIII concludes the paper.

II. RELATED WORK

Research on explanation spans philosophy, social sciences, and computer science. Philosophical work often frames explanation in terms of causality. Lewis [9] formalized causality using counterfactuals, while others distinguished explanatory, normative, and motivating reasons for action [10]. Probabilistic accounts extend this to stochastic systems such as Markov chains [11]. DARPA has highlighted the importance

of cognitive systems that can reason and explain their behavior rather than simply execute predefined commands [12]. In contrast, our conceptual framework moves beyond philosophical or probabilistic accounts to concrete system-level mismatches, where explanations are derived from violated assumptions in interacting digital systems.

In computer science, there is a close relation between monitoring, fault localization, self-repair and Model-Based Diagnosis (MBD). Reiter’s foundational theory [13] defined diagnosis as identifying minimal faulty components explaining discrepancies, while De Kleer extended this to multiple-fault scenarios with assumption-based truth maintenance [14]. Belief management approaches have also been applied to dynamic robotics domains, for example in IndiGolog, where inconsistencies between a robot’s modeled beliefs and observed events are explained through history-based diagnosis [15]. Optimization techniques such as RC-Tree improve minimal hitting set computation [16]. SAT-based techniques compare MBD and error explanation approaches [17], and have been applied to optimize fault localization in circuits [18], [19]. Kalech introduced social diagnosis for multi-agent systems [20], while others investigated distributed diagnosis under privacy constraints [21]–[23]. Complementary to such localization, work on self-repairing hardware addresses autonomous recovery from failures. This includes bio-inspired architectures for safety-critical CPS [24], reconfiguration using genetic algorithms [25], and neuromorphic self-repair mechanisms [26].

While these approaches restore functionality after faults, our framework instead clarifies why mismatches occur. Both perspectives are synergistic. Likewise, whereas MBD techniques primarily localize faulty components or actions, often in hardware or agent coordination settings, our framework shifts to explanation generation: mismatches are explained not by identifying faulty components but by isolating violated assumptions observable in execution traces.

Also related is abductive diagnosis, where explanations are hypotheses that account for observations. For example, [27] defines abductive hypotheses in propositional Horn logic and introduces “therapy” as the interleaving of diagnosis and repair. These logical approaches provide a foundation for reasoning about causes, but they remain limited to static system descriptions. Our framework builds on abductive reasoning but extends it with causal, counterfactual, and actionable explanations using model checking and SMT solving, enabling explanation of mismatches in dynamic and interacting systems.

Work in formal methods and debugging has explored diagnosis at the specification and circuit levels. SAT-based debugging with unsatisfiable cores localizes design errors [19], while scenario-based diagnosis techniques target inconsistencies in temporal logic specifications [28]. History/prophecy variable methods and Craig interpolation expand the toolbox for explaining mismatches [29]. Other work on finite-state machine testing [30] and sequential circuit debugging emphasizes distinguishing states or identifying minimal diagnoses. Safarpour and Veneris introduced bounded model debugging to manage long error traces more effectively [31], and further ad-

vanced automated design debugging using abstraction and refinement techniques [32], [33], achieving significant scalability improvements in practice. These methods improve verification and debugging, but they largely remain focused on structural or specification-level inconsistencies. In contrast, our conceptual framework addresses system-level behavioral mismatches, generating explanations through conflicting assumptions rather than through specification-level fault models.

Overall, our conceptual view for a system explaining its behavior is entirely different from diagnosis settings, even though similar reasoning techniques can be used.

Finally, research on self-explainability emphasizes trust and usability. Empirical studies in intelligent systems [1], clinical decision support [2], and context-aware applications [3] show that explanation style (e.g., detailed, why-based, complete) strongly impacts user trust. Recent frameworks such as MABEX [34], self-explanation in systems-of-systems [4], and design methodologies for self-explanatory systems [35] provide conceptual foundations but often lack formal mechanisms and automation for explanation generation. Our view is similar to the one in [4]; however, we go beyond by providing an actual algorithm and related case studies to compute explanations. Work on intelligent environments [36] and ambient applications [37] highlights the importance of context-sensitive explanations but relies on metadata or predefined models. Our framework integrates formal verification with explanation generation, producing actionable, causal, and counterfactual explanations for mismatches in interacting digital systems — a gap not fully addressed in prior conceptual or empirical work.

III. MODELING AND BEHAVIOR

This section introduces the notation employed to model interacting systems through Mealy machines and assumptions, and explains how Bounded Model Checking (BMC) is used to reason about their behavior.

A. Modeling

We adopt a hybrid modeling approach in which interacting digital systems are modeled as Mealy machines, while assumptions are used to capture aspects of behavior not explicitly defined in the state-transition model. This allows incomplete or underspecified models to be handled in a practical manner by representing missing details as assumptions.

Definition (Mealy Machine). A Mealy machine generates output from current input and state, denoted by:

$$M = (Q, \Sigma, O, \delta, \lambda, q_0),$$

where: Q is a finite set of states; Σ is a finite input alphabet; O is a finite output alphabet; $\delta : Q \times \Sigma \rightarrow Q$ is the state transition function, which defines how the machine transitions between states based on the current state and input; $\lambda : Q \times \Sigma \rightarrow O$ is the output function, which defines the output for a given state and input pair; and $q_0 \in Q$ is the initial state of the machine.

Definition (Assumption). An assumption is a *set of permissible sequences of inputs and outputs* describing temporal dependencies.

For example, in the case of a wind turbine, an assumption might state that “wind speed = 90 m/s” at a given time step, or more generally, “wind speed < 100 m/s” across all time steps. Based on such assumptions, the Mealy machine follows a specific behavior (or belief). For instance, if the wind speed input exceeds a certain threshold at a particular time step, the turbine is expected to transition to a blade adjustment mode. Practically, we map assumptions directly to SMT constraints, while temporal logic may be used as a more expressive formalism. These assumptions may partially specify system behavior, reflect environmental conditions, or system expectations.

B. Behavior

BMC uses a boolean formula to decide whether a series of states in a transition system satisfies a given set of assumptions up to a specified bound k [38]. If no such path exists for a given bound k , the bound is increased and the satisfiability check is repeated for longer executions. Unlike traditional BMC approaches that are formulated using Kripke structures, our work uses Mealy machines and assumptions to model system behavior. We use Satisfiability Modulo Theories (SMT) expressions with BMC.

Definition (BMC Unrolling). Using the above model for a Mealy machine, the formula for unrolling a model to represent paths up to length k is the following:

$$[M]_k = q_0 \wedge \bigwedge_{i=0}^{k-1} (\delta(q_i, \sigma_i) = q_{i+1}). \quad (1)$$

Let Γ denote assumptions for model M , e.g., constraining the input data σ_i in Equation 1 and let each $\gamma \in \Gamma$ be encoded over a finite trace of length k by an SMT formula $[\gamma]_k$. We encode the whole assumption set by the conjunction of its members:

$$[\Gamma]_k = \bigwedge_{\gamma \in \Gamma} [\gamma]_k. \quad (2)$$

The expected behavior of the system under these assumptions is then:

$$[B]_k = [M]_k \wedge [\Gamma]_k. \quad (3)$$

The operator $[\cdot]_k$ maps a model or a set of assumptions to a formula to be solved by an SMT solver. In our setting, we always seek the minimal k that leads to a mismatch, ensuring that explanations correspond to the shortest inconsistent trace.

The formulation presented here is a foundational one. More advanced model-checking techniques, such as interpolation-based model checking, property-directed reachability, and others, can be directly applied on top of this setup. However, for clarity and focus, we keep the explanation at the standard BMC level using unrolling and satisfiability.

Our methodology does not use hyperproperties in the formal sense; however, it does address a related class of relational behaviors. We compare the observed execution traces with the expected ones to identify mismatches, which requires reasoning over multiple traces in concurrently operating interacting models.

IV. CONCEPTUAL FRAMEWORK

The conceptual framework can, in general, involve multiple interacting systems, each of which may act as an explainer (the source or producer of an explanation) or an addressee (the recipient or target of an explanation). While our framework can be applied to an arbitrary number of interacting systems as we exemplify in VI-B, the following description considers two interacting systems for simplicity. One system assumes the role of the explainer, and the other the addressee. The explainer aims to improve transparency of the interaction by eliminating ambiguities that may arise during task execution. For example, in the context of wind park management, the wind park control acts as the addressee, receiving explanations from the wind turbines. This framing assumes that interaction involves a flow of information intended to clarify, justify, or make sense of some aspect of the explainer’s internal state, actions, or outputs for the addressee. Both the addressee and the explainer are modeled as Mealy machines plus assumptions. Here, M_E represents the explainer’s model (the state space of the explainer) while M_A represents the state space of the explainer from the addressee’s perspective. Both the addressee and the explainer maintain a set of assumptions, denoted Γ_A and Γ_E . In addition, assumptions model their connectivity denoted as Γ_C , which concerns the exchange of commands and feedback between the addressee and the explainer. Combining these assumptions $\Gamma_E, \Gamma_A, \Gamma_C$ with the behavioral models M_E, M_A determines the belief about expected behavior.

We distinguish between different sets of assumptions not only for modeling clarity but also to guide the explanation process. Γ_C establishes the link between the explainer and the addressee, Γ_A represents the addressee’s expectations, while Γ_E describes environmental and internal conditions known to the explainer but unknown to the addressee. We treat Γ_A and Γ_C as hard assumptions that must always hold, while Γ_E forms the set of soft assumptions from which explanations are generated. This separation defines the search space in which the SMT solver identifies explanations.

A. Explanation

When the expected behavior of the interacting digital system is consistent with the behavior as perceived by the addressee, no explanation is required. However, if an inconsistency arises, an explanation \mathcal{E} must be generated.

This can directly be analyzed using the assumptions $\Gamma_A, \Gamma_E, \Gamma_C$ with the models M_A and M_E . Specifically, in our windpark example, both the windpark’s model of the turbine M_A and the turbine’s model M_E are expected to produce consistent output transitions after the turbine receives a respective command from the wind park. We refer to this consistency condition as the explanation target, denoted $T \subseteq \Gamma_C$.

Definition (Explanation). If the explanation target T cannot be achieved, an explanation is a set of assumptions that causes this mismatch. Specifically, we consider assumptions Γ_E of the explainer to understand the mismatch, i.e., identifying a minimal subset of assumptions $\mathcal{E} \subseteq \Gamma_E$. By removing this

subset from the assumptions of the explainer, i.e., $\Gamma_E \setminus \mathcal{E}$, the explanation target T can be achieved.

Additionally, a weighting function can provide penalty values for removed assumption, i.e., $w : \Gamma_E \rightarrow \mathbb{N}$.

B. Formalization using BMC

For generating explanations, we use reasoning similar to the one in BMC to encode our conceptual framework. Thus, we define our model $[M_C]_k$ of the interacting digital system as the joint unrolling of $[M_A]_k$ and $[M_E]_k$:

$$[M_C]_k = [M_A]_k \wedge [M_E]_k, \quad (4)$$

The conjunction of all assumptions is $[\Gamma]_k$:

$$[\Gamma]_k = [\Gamma_A]_k \wedge [\Gamma_E]_k \wedge [\Gamma_C]_k, \quad (5)$$

The believed behavior of the interacting digital system under a finite execution trace of length k is defined as:

$$[B_C]_k = [M_C]_k \wedge [\Gamma]_k, \quad (6)$$

Now, assume that Γ_C has the explanation target T requiring both models to end in the same final state. In that case, $[\Gamma_C]_k$ would contain a constraint of the form $[T]_k = (q_{A,k} == q_{E,k})$. If $[B_C]_k$ is satisfiable, then the explaining digital system can behave consistently with the behavior expected by the addressee and no explanation is required. Otherwise, if $[B_C]_k$ is *unsatisfiable*, the perceived behavior must lead to a different state than expected. We consider this inconsistency to be caused by some subset of the explainer's assumptions Γ_E . The weighting function serves to find the explanation:

$$\mathcal{E}_{\min} = \operatorname{argmax}_{\gamma \in \Gamma_E \cap U_{\min}} \{w(\gamma)\}, \quad (7)$$

where U_{\min} is a minimal unsatisfiable core.

However, practically, there may be more than one minimal unsatisfiable core, these cores may not overlap, and the SMT solver may not return a minimal, but some larger unsatisfiable core. Our algorithm handles these cases using a greedy approach to compute an explanation \mathcal{E} from some unsatisfiable core U .

V. ALGORITHM TO COMPUTE EXPLANATIONS

Our algorithm implements the BMC-based reasoning procedure to explain mismatches in interacting digital systems behavior. Inputs to the algorithm are: M_E , the explainer's model; M_A , represents the state space of the explainer from the addressee's perspective; Γ_E , the explainer's assumptions; Γ_A , the addressee's assumptions; and Γ_C , the connectivity's assumptions. It produces the explanation \mathcal{E} as an output.

The algorithm starts by initializing \mathcal{E} as an empty set (Line 1), then defines the interacting digital system's model as a conjunction of $[M_A]_k$ and $[M_E]_k$ (Line 2). A conjunction operation of all assumptions in the interacting digital system is performed according to Equation (5) (Line 3). The algorithm then formulates the perceived behavior of the interacting digital system by following Equation (6) (Line 4). To check whether the expected behavior matches the perceived behavior,

Algorithm 1 Explainer ($M_E, M_A, \Gamma_E, \Gamma_A, \Gamma_C$)

```

1:  $\mathcal{E} \leftarrow \emptyset$ .
2:  $[M_C]_k \leftarrow [M_E]_k \wedge [M_A]_k$ .
3:  $[\Gamma]_k \leftarrow [\Gamma_A]_k \wedge [\Gamma_E]_k \wedge [\Gamma_C]_k$ .
4:  $[B_C]_k \leftarrow [M_C]_k \wedge [\Gamma]_k$ .
5: while CHECK_SAT( $[B_C]_k$ ) = UNSAT do
6:    $\Gamma_V \leftarrow \text{ExtractConflictingAssumptions}()$ .
7:   OrderConflictingAssumptions( $\Gamma_V$ , weights).
8:    $\Gamma_R \leftarrow \text{RemoveAssumption}(\Gamma_V)$ .
9:    $\mathcal{E} \leftarrow \mathcal{E} \cup \Gamma_R$ .
10:   $[B_C]_k \leftarrow [M_C]_k \wedge [\Gamma \setminus \mathcal{E}]_k$ .
11: end while
12: return  $\mathcal{E}$ .

```

the algorithm performs a BMC check on $[B_C]_k$ (Line 5). If the result is UNSAT, it indicates that the system behaved unexpectedly and that an explanation is needed. The algorithm extracts the conflicting assumptions $\Gamma_V \subseteq \Gamma_E \cap U$, where U is a— not necessarily minimal or unique—unsatisfiable core (Line 6). The set Γ_V serves as the basis from which the explainer generates the explanation \mathcal{E} using a greedy approach. To determine this subset, the explainer performs the following steps (Lines 7 to 10):

- 1) Apply a weighting strategy w over Γ_V to identify the conflicting assumptions that have the greatest impact on the system's unexpected behavior.
- 2) Remove the highest-weighted assumption from Γ_V . The removed conflicting assumptions are collected in the set Γ_R .
- 3) Update the explanation and belief: Add Γ_R to \mathcal{E} , and update the perceived behavior using the revised assumption set.
- 4) Repeat steps 2 and 3 while the following formula is unsatisfiable:

$$[B_C]_k = [M_C]_k \wedge [\Gamma \setminus \mathcal{E}]_k.$$

The explanation is returned in (Line 12). What makes this explanation causal is that it establishes a relation between the cause (the conflicting assumptions) and their effect (the inconsistency), making it clear which conflicting assumptions caused the inconsistency. Moreover, our explanation is counterfactual as well, because it works on exploring alternative scenarios showing that if a minimal set of conflicting assumptions are removed, the inconsistency resolves. This process of eliminating problematic assumptions explores the existence of counterfactual situations by answering the question "what would happen if this assumption is not included" and by proving that the system would be consistent without it, confirming that an alternative set of assumptions would resolve the issues. Thus, the algorithm implements a form of counterfactual reasoning. In practice, there may be several potential minimal explanations, each of which would restore consistency by collecting a different group of conflicting assumptions. Currently, our algorithm uses a heuristic ordering technique to choose one such explanation. However, exploring all minimal explanations remains an area for future exploration. The source code implementing the proposed algorithm, together with all experimental setups, is publicly available [39].

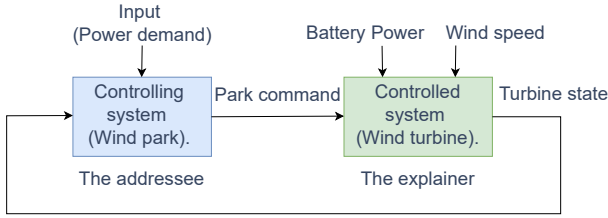


Fig. 1. Illustration of the interaction between a wind park (controlling system) and a wind turbine (controlled system) within an explanatory framework.

VI. CASE STUDIES

To demonstrate our framework, we provide two examples that illustrate its application in generating explanations.

A. Wind Park & Turbine

The wind park acts as the addressee, issuing commands to the turbine, which acts as the explainer, executing commands subject to internal and environmental conditions (e.g., wind speed, battery power).

1) *Interaction Model*: Figure 1 illustrates the interaction when the park issues a command based on power demand, the turbine processes this command along with sensor inputs. The park perceives only three outward states (*shutdown*, *reduced speed*, *normal operation*) representing the turbine’s expected behavior from the park perspective, whereas the turbine transitions through additional internal states (e.g., *power ramp up*, *power ramp down*, *synchronization connect*, *synchronization disconnect*, *acceleration*, *deceleration*, *blade adjustment* and *yaw alignment*) to realize these observed behaviors.

2) *Assumptions*: Γ_A represents the commands issued by the park to the turbine. Γ_C represents the synchronization assumptions that link the park’s commands with the turbine’s observable outputs. Γ_E represents the turbine’s own assumptions, including environmental and internal conditions such as wind speed and battery power.

3) *Mismatch Example*: A mismatch arises when the park requests normal operation while wind speed exceeds the threshold of 100 m/s; the turbine enters blade adjustment mode instead. The turbine identifies “wind speed > 100 m/s” as the conflicting assumption and reports it to the park. This forms the basis of an explanation according to our framework.

B. Traffic Controller & Autonomous Vehicles

We next consider a centralized traffic controller and multiple autonomous vehicles traveling in sequence along the same direction. In this setting, the traffic controller acts as the addressee, while each vehicle functions as an explainer.

1) *Interaction Model*: The controller issues traffic commands based on external demand, while each vehicle processes these commands alongside its own local conditions such as pedestrian detection, lane occupancy, road friction, and battery health. The traffic controller can issue three outward commands (*stop*, *slow*, and *proceed*). These commands define the expected behavior of vehicles from the controller’s perspective. Hence, the addressee’s model M_A is abstract,

TABLE I
ASSUMPTIONS OF THE WIND PARK SCENARIO

Γ	Description
Γ_A .	1) The wind park issues a persistent command (Normal Operation).
Γ_C .	1) Turbine outward states are exclusive (Shutdown, Reduced Speed, Normal Operation). 2) If the park issues a Normal Operation command, the turbine must operate in Normal Operation.
Γ_E .	1) Wind speed exceeds the threshold ($\text{wind_speed} > 100$ m/s). 2) Battery power is within safe limits ($\text{battery_power} > 10\%$). 3) If wind speed exceeds the threshold, the turbine must transition to Blade Adjustment. 4) If battery power falls below the threshold, the turbine transitions to Reduced Speed mode.

TABLE II
ASSUMPTIONS OF TRAFFIC SCENARIO

Γ	Description
Γ_A .	1) The traffic controller issues a persistent command (PROCEED).
Γ_C .	1) Vehicle outward states are exclusive (STOP, SLOW, PROCEED). 2) If the controller issues PROCEED, both vehicles are expected to outwardly PROCEED. 3) Safety coupling: if Vehicle 1 stops, then Vehicle 2 must also stop.
Γ_E .	1) Vehicle 1 detects a pedestrian ($\text{ped1}=1$). 2) Vehicle 2 does not detect a pedestrian ($\text{ped2}=0$). 3) If a pedestrian is detected, the vehicle must outwardly STOP. 4) Lane ahead of Vehicle 1 is free ($\text{occ1}=0$). 5) Lane ahead of Vehicle 2 is free ($\text{occ2}=1$). 6) If a vehicle outwardly stops, its lane is assumed occupied. 7) Road friction sufficient for Vehicle 1 ($\text{friction_ok1}=1$). 8) Road friction sufficient for Vehicle 2 ($\text{friction_ok2}=1$). 9) Battery functional for Vehicle 1 ($\text{battery_ok1}=1$). 10) Battery functional for Vehicle 2 ($\text{battery_ok2}=1$).

consisting of only three states corresponding to the outward commands. In contrast, each vehicle’s model M_E includes nine internal states (*idle*, *align*, *gap_assess*, *accel*, *sync_merge*, *cruise*, *crawl*, *hold*, and *brake*). The transitions between these states depend not only on the controller’s commands but also on environmental assumptions, e.g., whether a pedestrian is detected, a lane is occupied, road friction is sufficient, and the vehicle’s battery is functional. This demonstrates how incompletely specified behavior can (1) be sufficient within our conceptual framework for explanation, (2) temporal assumptions are used to further constrain the (expected) behavior, and (3) the framework handles more than two systems.

2) *Assumptions*: Γ_A represents the commands issued by the traffic controller to the vehicles. Γ_C represents the synchronization assumptions that align the controller’s commands with the vehicle’s outward responses. Γ_E represents the vehicles’ own assumptions, including environmental and safety conditions such as pedestrian detection, lane occupancy, road friction, and battery status.

3) *Mismatch Example*: A mismatch arises when the controller issues a *proceed* command, but a pedestrian is detected, so the vehicle transitions to *brake*. From the controller’s perspective, this represents an unexpected deviation. The framework then requires the vehicle to generate an explanation by identifying the violated assumption (“pedestrian detected”) that caused the mismatch.

VII. EXPERIMENTAL RESULTS

We show the explanatory performance of our approach by illustrating two test scenarios based on the case studies introduced previously. These scenarios evaluate how well the system’s explanations align with the perceived behavior of the interacting components. We focus on the scenarios when the system’s expected behavior mismatches the actual observed behavior, so an explanation is needed.

A. Wind Park Control & Wind Turbine

In this experiment, we implement the mismatch scenario of Section VI-A3. Table I shows the assumptions. The *UNSAT core* is $\{\Gamma_{A.1}, \Gamma_{C.2}, \Gamma_{E.3}\}$. So the algorithm considers the conflicting assumption in Γ_C as an explanation target T , then follows the four reconciliation steps described in Section V, considering that the violated assumptions are stored in a vector. After removing $\Gamma_{E.1}$, none of the remaining assumptions in the vector are violated and T is satisfied, indicating that \mathcal{E} contains the conflicting assumption in Γ_E .

B. Traffic Controller & Autonomous Vehicles

This experiment implements the mismatch scenario of Section VI-B3 using the assumptions in Table II. The *UNSAT core* is $\{\Gamma_{A.1}, \Gamma_{C.2}, \Gamma_{E.1}, \Gamma_{E.3}\}$, resolved by $\mathcal{E} = \{\Gamma_{E.1}\}$. The test results show that in mismatch cases, the algorithm can successfully isolate the responsible assumptions and return them as an explanation.

C. Scalability of the Explanation Algorithm

Let us consider Q_A as the finite set of states for the addressee’s model, while Q_E is the finite set of states for the explainer’s model.

1) *Depth scaling*: We extend the wind park scenario to evaluate the explanation algorithm under increasing k . We constructed four versions of the composite model M_C , each with $Q_A = 3$ and varying values of Q_E : 10, 17, 24, and 31 states, respectively. These represent increasingly detailed turbine models, each covering more operational scenarios.

- **1st configuration**: The baseline model described in Section VI, which includes the following states: *normal operation, power ramp up, power ramp down, synchronization connect, synchronization disconnect, acceleration, deceleration, reduced speed, blade adjustment, yaw alignment, and shutdown*.
- **2nd configuration**: Extends the baseline by adding states such as *rotor position verification, dynamic balance check, subsystem self-test, sensor recalibration, power system pre-check, and control logic initialization*.
- **3rd configuration**: Further adds *wind speed evaluation, grid pre-synchronization check, brake release, bearing warm-up, pitch calibration, inertia synchronization, voltage regulation, and frequency matching*.
- **4th configuration**: Further enriches the model with *rotor position verification, dynamic balance check, subsystem self-test, sensor recalibration, power system pre-check,*

TABLE III
SCALABILITY UNDER INCREASING MODEL COMPLEXITY (EXAMPLE 1).

$ Q_E $	$ Q_A $	CPU [s]	Memory [MB]	SMT Clauses	SMT Vars	k
10	3	0.22	25	4,931	1,063	7
17	3	2.42	36	83,325	3,219	14
24	3	33.38	62	575,346	6,992	21
31	3	71.52	102	531,517	11,871	28

control logic initialization, and safety systems engagement.

These modifications preserve the turbine model’s semantics while enlarging the decision space, enabling a controlled scalability evaluation. The SMT model encodes both Mealy states and environmental variables (e.g., wind speed, battery status), which shape guards and transitions. The bound k (Table III) grows with behavioral complexity, reflecting longer traces. Solver metrics (Table III) show CPU and memory rising with $|Q_E|$; SMT variable count grows consistently, while clauses fluctuate due to solver optimizations.

2) *Breadth Scaling*: For breadth scalability, we evaluate the traffic scenario by increasing the number of autonomous vehicles from 1 to 10, while keeping each vehicle’s internal model fixed ($|Q_E| = 9$). Unlike the turbine setting, the bound k remains constant at 11, since the per-vehicle progression depth does not change. Solver metrics (Table IV) show that CPU time and memory usage grow nearly linearly with the number of vehicles. Clause and variable counts also scale proportionally with the number of vehicles, reflecting the additive nature of parallel models.

The algorithm consistently produced correct explanations across all cases, indicating that the approach remains robust under both breadth and depth scaling.

TABLE IV
SCALABILITY UNDER INCREASING MODEL COMPLEXITY (EXAMPLE 2).

Vehicle No	$ Q_E $	$ Q_A $	CPU [s]	Memory [MB]	SMT Clauses	SMT VAR	k
1	9	3	0.07	54.50	1,952	737	11
2	18	3	0.14	56.00	3,582	1,353	11
3	27	3	0.20	59.39	5,195	1,969	11
4	36	3	0.26	62.88	6,951	2,585	11
5	45	3	0.30	66.27	8,701	3,201	11
10	90	3	0.60	87.82	17,115	6,281	11

VIII. CONCLUSION

We demonstrated the capability of our framework and algorithm to automatically explain mismatches between expected and perceived behavior. Two case studies underpin the capabilities of the approach in system modeling. Future work includes the exploration of other self-explanation aspects and on real-time application for self-explanation.

For other self-explanation aspects, the conceptual framework can be modified to, e.g., explain inconsistencies in other parts of the models beyond assumptions. By this, explanations may describe model transformations or situations to be avoided.

For real-time application in a self-explaining system, solving intermediate BMC instances is infeasible. Instead we plan to explore storing pre-computed explanations for expected problematic situations.

REFERENCES

- [1] B. Y. Lim, A. K. Dey, and D. Avrahami, "Why and why not explanations improve the intelligibility of context-aware intelligent systems," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. Chi '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 2119–2128.
- [2] A. Bussone, S. Stumpf, and D. O'Sullivan, "The role of explanations on trust and reliance in clinical decision support systems," in *2015 International Conference on Healthcare Informatics*, 2015, pp. 160–169.
- [3] T. Kulesza, S. Stumpf, M. Burnett, S. Yang, I. Kwan, and W.-K. Wong, "Too much, too little, or just right? ways explanations impact end users' mental models," in *2013 IEEE Symposium on Visual Languages and Human Centric Computing*, 2013, pp. 3–10.
- [4] G. Fey, M. Fränzle, and R. Drechsler, "Self-explanation in systems of systems," in *2022 IEEE 30th International Requirements Engineering Conference Workshops (REW)*, 2022, pp. 85–91.
- [5] M. Langer and I. Valera, "Leveraging actionable explanations to improve people's reactions to ai-based decisions," in *Bridging the Gap Between AI and Reality*, B. Steffen, Ed. Cham: Springer Nature Switzerland, 2025, pp. 293–306.
- [6] R. Singh, T. Miller, H. Lyons, L. Sonenberg, E. Velloso, F. Vetere, P. Howe, and P. Dourish, "Directive explanations for actionable explainability in machine learning applications," *ACM Trans. Interact. Intell. Syst.*, vol. 13, no. 4, Dec. 2023.
- [7] J. Woodward, *Making Things Happen: A Theory of Causal Explanation*. Oxford University Press, 01 2004.
- [8] R. Guidotti, "Counterfactual explanations and how to find them: literature review and benchmarking," *Data Mining and Knowledge Discovery*, vol. 38, no. 5, pp. 2770–2824, 2024.
- [9] D. Lewis, "Causation," *Journal of Philosophy*, vol. 70, no. 17, pp. 556–567, 1973.
- [10] M. Alvarez, "Reasons for action: Justification, motivation, explanation," in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed. Metaphysics Research Lab, Stanford University, 2017.
- [11] C. Baier, F. Funke, S. Jantsch, J. Piribauer, and R. Ziemek, "Probabilistic causes in markov chains," in *Automated Technology for Verification and Analysis*, Z. Hou and V. Ganesh, Eds. Cham: Springer International Publishing, 2021, pp. 205–221.
- [12] R. Brachman, "Systems that know what they're doing," *IEEE Intelligent Systems*, vol. 17, no. 6, pp. 67–71, 2002.
- [13] R. Reiter, "A theory of diagnosis from first principles," *Artificial Intelligence*, vol. 32, no. 1, pp. 57–95, 1987.
- [14] J. de Kleer and B. C. Williams, "Diagnosing multiple faults," *Artificial Intelligence*, vol. 32, no. 1, pp. 97–130, 1987. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0004370287900634>
- [15] S. Gspandl, I. Pill, M. Reip, G. Steinbauer, and A. Ferrein, "Belief management for high-level robot programs," in *IJCAI International Joint Conference on Artificial Intelligence*, 2011.
- [16] I. Pill and T. Quaritsch, "Rc-tree: A variant avoiding all the redundancy in Reiter's minimal hitting set algorithm," in *2015 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2015, pp. 78–84.
- [17] H. Rienner and G. Fey, "Model-based diagnosis versus error explanation," in *Tenth ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMCODE2012)*, 2012, pp. 43–52.
- [18] A. Feldman, I. Pill, F. Wotawa, I. Matei, and J. de Kleer, "Efficient model-based diagnosis of sequential circuits," in *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, ser. AAAI 2020 - 34th AAAI Conference on Artificial Intelligence. AAAI Press, 2020, pp. 2814–2821, publisher Copyright: Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.; 34th AAAI Conference on Artificial Intelligence, AAAI 2020, AAAI-20 ; Conference date: 07-02-2020 Through 12-02-2020. [Online]. Available: <https://aaai.org/Conferences/AAAI-20/>
- [19] A. Suelflow, G. Fey, R. Bloem, and R. Drechsler, "Using unsatisfiable cores to debug multiple design errors," in *Proceedings of the 18th ACM Great Lakes Symposium on VLSI*, ser. Glsvlsi '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 77–82. [Online]. Available: <https://doi.org/10.1145/1366110.1366131>
- [20] M. Kalech and G. A. Kaminka, "On the design of coordination diagnosis algorithms for teams of situated agents," *Artificial Intelligence*, vol. 171, no. 8-9, pp. 491–513, 2007.
- [21] A. Natan and M. Kalech, "Privacy-aware distributed diagnosis of multi-agent plans," *Expert Systems with Applications*, vol. 192, p. 116313, 2022.
- [22] R. Micalizio and P. Torasso, "Plan diagnosis and agent diagnosis in multi-agent systems," in *AI*IA 2007: Artificial Intelligence and Human-Oriented Computing*, R. Basili and M. T. Pazzienza, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 434–446.
- [23] N. Roos and C. Witteveen, "Models and methods for plan diagnosis," *Autonomous Agents and Multi-Agent Systems*, vol. 19, no. 1, pp. 30–52, Aug. 2009. [Online]. Available: <https://doi.org/10.1007/s10458-007-9017-6>
- [24] S. S. Khairullah and C. R. Elks, "A self-repairing hardware architecture for safety-critical cyber-physical-systems," *CoRR*, vol. abs/1910.14127, 2019. [Online]. Available: <http://arxiv.org/abs/1910.14127>
- [25] L. Maguire, E. Coyle, and M. McGinnity, "Self-repair of embedded systems," *Engineering Applications of Artificial Intelligence*, vol. 17, no. 1, pp. 1–9, Feb. 2004, this paper arises from a wholly industrially funded PhD studentship (International Test Technologies Ltd) and introduces the concept of using UML to affect both the hardware and software self-repair of an embedded system, building on previous published work by the authors on intelligent testing, diagnosis and repair. This publication has recently led to discussions with Scisys Ltd. seeking to exploit the research in space applications.
- [26] Z. Han, A. N. M. N. Islam, and A. Sengupta, "Astromorphic self-repair of neuromorphic hardware systems," 2023. [Online]. Available: <https://arxiv.org/abs/2209.07428>
- [27] G. Friedrich, G. Gottlob, and W. Nejdl, "Hypothesis classification, abductive diagnosis and therapy," in *Expert Systems in Engineering Principles and Applications*, G. Gottlob and W. Nejdl, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 69–78.
- [28] I. Pill and T. Quaritsch, "Behavioral diagnosis of LTL specifications at operator level," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, ser. Ijcai '13. AAAI Press, 2013, p. 1053–1059.
- [29] C. Vick and K. L. McMillan, "Synthesizing history and prophecy variables for symbolic model checking," in *Verification, Model Checking, and Abstract Interpretation*, C. Dragoi, M. Emmi, and J. Wang, Eds. Cham: Springer Nature Switzerland, 2023, pp. 320–340.
- [30] R. Alur, C. Courcoubetis, and M. Yannakakis, "Distinguishing tests for nondeterministic and probabilistic machines," in *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, ser. Stoc '95. New York, NY, USA: Association for Computing Machinery, 1995, p. 363–372.
- [31] S. Safarpour, A. Veneris, and F. Najm, "Managing verification error traces with bounded model debugging," in *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2010, pp. 601–606.
- [32] S. Safarpour and A. Veneris, "Automated design debugging with abstraction and refinement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1597–1608, 2009.
- [33] —, "Automated debugging with high level abstraction and refinement," in *2009 IEEE International High Level Design Validation and Test Workshop (HLDVT)*, 2009, pp. 26–31.
- [34] M. Blumreiter, J. Greenyer, F. J. Chiyah Garcia, V. Klös, M. Schwammberger, C. Sommer, A. Vogelsang, and A. Wortmann, "Towards self-explainable cyber-physical systems," in *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 2019, pp. 543–548.
- [35] R. Drechsler, "Keynotes: Towards self-explaining digital systems: A design methodology for the next generation," in *2018 IEEE 3rd International Verification and Security Workshop (IVSW)*, 2018, pp. i–iii.
- [36] S. Autexier and R. Drechsler, "Towards self-explaining intelligent environments," in *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 2018, pp. 1–6.
- [37] B. Kordts, B. Gerlach, and A. Schrader, "Towards self-explaining ambient applications," in *Proceedings of the 14th Pervasive Technologies Related to Assistive Environments Conference*, ser. Petra '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 383–390.
- [38] E. Clarke, A. Biere, R. Raimi, and Y. Zhu, "Bounded model checking using satisfiability solving," *Formal Methods in System Design*, vol. 19, no. 1, pp. 7–34, 2001.
- [39] "Reproducing the experiments of automated self-explanation of expected versus perceived behavior for interacting digital systems," 2026. [Online]. Available: <https://doi.org/10.5281/zenodo.18220339>