Interface-Aware Signal Temporal Logic

Thomas Ferrère
IST Austria
thomas.ferrere@ist.ac.at

Dejan Nickovic
AIT Austrian Institute of Technology
Dejan.Nickovic@ait.ac.at

Alexandre Donzé
Decyphir, France
alex@decyphir.com

Hisahiro Ito
Toyota Motor Corporation
hisahiro_ito@mail.toyota.co.jp

James Kapinski
Toyota Research Institute of North America
jim.kapinski@toyota.com

ABSTRACT

Safety and security are major concerns in the development of Cyber-Physical Systems (CPS). Signal temporal logic (STL) was proposed as a language to specify and monitor the correctness of CPS relative to formalized requirements. Incorporating STL into a development process enables designers to automatically monitor and diagnose traces, compute robustness estimates based on requirements, and perform requirement falsification, leading to productivity gains in verification and validation activities; however, in its current form STL is agnostic to the input/output classification of signals, and this negatively impacts the relevance of the analysis results.

In this paper we propose to make the interface explicit in the STL language by introducing input/output signal declarations. We then define new measures of input vacuity and output robustness that better reflect the nature of the system and the specification intent. The resulting framework, which we call interface-aware signal temporal logic (IA-STL), aids verification and validation activities. We demonstrate the benefits of IA-STL on several CPS analysis activities: (1) robustness-driven sensitivity analysis, (2) falsification and (3) fault localization. We describe an implementation of our enhancement to STL and associated notions of robustness and vacuity in a prototype extension of Breach, a MATLAB®/Simulink® toolbox for CPS verification and validation. We explore these methodological improvements and evaluate our results on two examples from the automotive domain: a benchmark powertrain control system and a hydrogen fuel cell system.

CCS CONCEPTS

• **Computing methodologies** → *Simulation evaluation*; • **Theory of computation** → *Modal and temporal logics*;

1 INTRODUCTION

Cyber-physical systems (CPS) combine computational and physical components that interact with their environment via sensors and actuators. These systems are complex and often used in mission-critical applications. Verification and validation (V&V) for such systems is an essential activity that ensures high standards of safety and performance are met and sufficient correctness evidence is gathered for regulatory bodies.

Runtime monitoring is a formal, yet pragmatic method for performing analysis of individual system behaviors. When used at design-time, runtime monitoring makes the V&V process more systematic and rigorous, while remaining scalable (see [1] for an overview). Signal Temporal Logic (STL) [2] is a specification formalism for expressing real-time temporal safety and performance properties of CPS. In its standard, qualitative interpretation, STL acts as a binary classifier that partitions behaviors according to their satisfaction or violation of a specification.

Fainekos and Pappas proposed in [3, 4] to equip temporal logic with a quantitative interpretation, whose adaptation to STL was studied in [5]. This alternative semantics replaces the binary satisfaction relation with a real-valued *robustness degree* indicating how far an observed signal is from satisfying or violating the specification. Monitoring STL with quantitative semantics can be done efficiently [6] and is implemented in the Breach [7] and S-TaLiRo [8] tools. These tools use robustness computations to perform *falsification* [9, 10] and *specification mining* [11]. Falsification is a search-based testing method that explores a system parameter-space to find behaviors that violate a given specification. Specification mining can be either seen as a process of learning full specifications [12, 13] or as identifying parameters in specification templates [14–17]. Both approaches use robustness computations to guide the search for tightest parameters or for most informative formulas.

One shortcoming of STL is that the standard procedure for measuring the robustness of an observed behavior with respect to an STL specification sometimes gives unexpected and nonintuitive results. We illustrate this observation with the following scenario. A verification engineer is evaluating the system-under-test S depicted in Figure 1-(top). The system S receives requests and is expected to respond to each request with a grant within 2 time units. We say that there is a request (grant), whenever the signal *req* (*gnt*) is above the threshold 4. We formalize this requirement with the following STL specification: $\varphi \equiv \square((req \geq 4) \rightarrow \Diamond_{[0,2]}(gnt \geq 4))$.

The verification engineer designs a test input *req* consisting of a train of requests, executes it on S and observes the system output *gnt*. The signals *req* and *gnt* in the resulting trace w are depicted in Figure 1-(bottom). These two signals witness the violation of φ – in w the requests from *req* are never granted because *gnt* fails to reach the expected threshold 4. In particular, we can observe that *gnt* is 3 units away from reaching the expected threshold. As a consequence, we expect the robustness degree, denoted by $\rho(\varphi, w)$, to be equal to -3, measuring how bad the system reaction is with respect to the specification. Instead, the standard robustness of [4, 5] implemented in S-TaLiRo and Breach gives us $\rho(\varphi, w) = -1$.

The reason for this counter-intuitive result is that, in this example, the value of $\rho(\varphi, w)$ relates to *req*; that is, the robustness value -1 indicates that it suffices to reduce the amplitude of the pulses of *req* in w by 1 unit to move this signal below the threshold

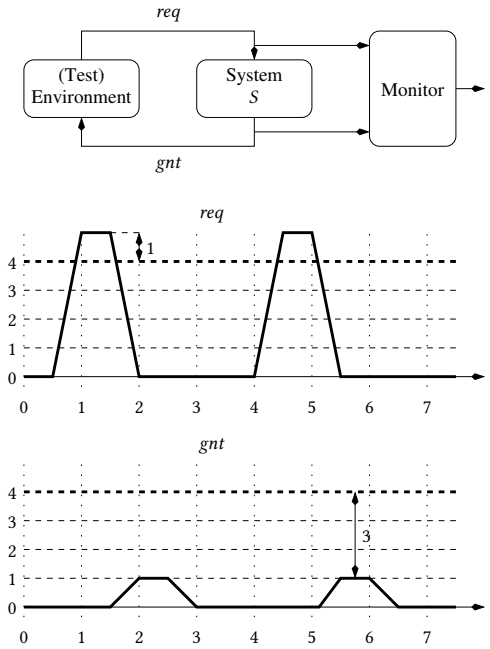

Figure 1: Request-grant property: (top) system S and its test environment, (bottom) signals *req* and *gnt*.

4, thus removing all requests from w. Over signals that do not issue any request, formula φ is vacuously satisfied. Thus the robustness value in this case does not relate to how robust the system is when measured at its output, but rather how close the input is to being vacuous (how near it is to not exercising the system at all).

In this work, we propose an extension of STL that classifies signals as *inputs* and *outputs*. This simple, yet fundamental addition to the temporal logic allows us to specify the system-under-test as an input/output relation and not a set of correct execution traces. Separating responsibilities between the system-under-test and its environment is a key aspect of hierarchical system design [18], and adding interface information to the specification is a significant enhancement, from a methodological point of view. In particular, the definition of the interface in a specification allows us to separately reason about the quality of inputs (does the input exercise the system in any meaningful way?) and the quality of outputs (how good is the reaction of the system to the given input?). We develop new notions of robustness and vacuity that capture the distance to satisfaction and violation measured at the input and output of the system separately, which we argue better capture the robustness of a system. We demonstrate the benefits of *interface-aware* specifications and associated notions of robustness and vacuity on several CPS analysis activities:

(1) Robustness-driven sensitivity analysis of CPS;
(2) Falsification-based testing;
(3) Robustness-guided fault localization and explanation.

For this last point we enhance the trace diagnostics algorithm of [19] by adding some information relative to the robustness analysis, and not just the pass/fail status of the property. This provides better debugging information and makes the robustness analysis process more transparent. We evaluate the results on one academic and one industrial case study, where both examples come from the automotive domain.

2 RELATED WORK

We consider requirements that qualify a system in terms of its input/output behavior, inspired by the ST-Lib library of specifications [20]. ST-Lib supports the formal specification of behavior requirements for automotive control systems. This library includes many of the types of performance and functional behaviors of interest to engineers developing embedded control applications. In particular, the library features STL formulas that capture overshoot, settling times, rise times, and steady-state behaviors, and most involve the interplay between input signals and expected response (output) behaviors.

The distinction between environment assumptions and system guarantees can help improve the performance of falsification in the presence of potentially vacuous input signals. This problem was studied in [21] for STL formulas of the form $\square(\alpha \rightarrow \beta)$ with the antecedent α ranging over inputs and the consequent β ranging over output signals. That work employs a Gaussian process regression to estimate the region of the input space in which α holds with high probability. A similar problem was addressed by Dokhanchi et al. [22]. That work proposes to improve the falsification procedure by first searching for a prefix input in which the antecedent α is satisfied and then searching for a suffix input such that over the corresponding output the consequent β is falsified. In both works the specific structure of the formula is used to identify parts of the specification related to input versus output. Our method provides a more versatile solution: by enriching the specification language with explicit signature declarations we are able to handle any specification that relates inputs to outputs, without any restriction on the form of the specification.

The more general problem of vacuity as an assessment of the quality of specifications has been studied in the context of discrete-time temporal logics. Vacuity detection in temporal logic was first studied in [23] for the ACTL fragment of CTL (computation tree logic). The vacuity properties of CTL* temporal specifications were further studied in the context of model checking in [24]. The idea of vacuity from model checking is extended to the testing setting in [25] as a notion that complements test coverage. We adopt a more practical approach, in which we define vacuity of the specification only with respect to a concrete input signal.

Explaining a specification violation is another important aspect of STL testing methodologies. Fault explanation in terms of *temporal implicants*, small segments of the observed behavior that witness the violation, was proposed in [26]. That method uses standard STL with qualitative semantics, and it neither captures input/output relationships between signals nor identifies the explanation of the worst case behavior. By contrast, our method uses both the input/output signature of the specification and its robustness degree to identify the precise subset of signals and time points that are responsible for the observed robustness value.

3 INTERFACE-AWARE STL

We first provide an introduction to *signal temporal logic* (STL), and then augment it with input/output declarations to arrive at an enhanced version of the language, which we call interface-aware STL (IA-STL). We then show how the interface information can be used to make the STL robustness measure more meaningful. For this, we first develop a notion of *relative robustness* that restricts temporal logic robustness to a subset of signals. We use this general notion of robustness to define two new measures in the context of IA-STL, which we call *output robustness* and *input vacuity*. Finally we study the properties of the new proposed notions and discuss their intended use.

Syntax and Semantics. We start with some background definitions. Let $S = \{s_1, \dots, s_n\}$ be a set of signal variables. We define the time domain $\mathbb{T}$ to be of the form $[0, d] \subset \mathbb{R}$. A *signal* or *trace* w is a function $\mathbb{T} \to \mathbb{R}^n$, which we also see as a vector of real-valued signals $w_i : \mathbb{T} \to \mathbb{R}$ associated to variables s_i for $i = 1, \dots, n$. In the following we assume that every w_i is piecewise monotone and bounded.[1] Given two signals $u : \mathbb{T} \to \mathbb{R}^l$ and $v : \mathbb{T} \to \mathbb{R}^m$, we define their parallel composition $u \| v : \mathbb{T} \to \mathbb{R}^{l+m}$ in the expected way. Given a signal $w : \mathbb{T} \to \mathbb{R}^n$ and variables $R \subseteq S$ with $R = \{s_{i_1}, \dots, s_{i_m}\}$ for some $1 \le i_1 < \dots < i_m \le n$, we define the projection of w onto R as follows: $w_R = w_{i_1} \| \dots \| w_{i_m}$.

Let Θ be a set of terms of the form $f(R)$ where $R \subseteq S$ are subsets of variables and $f : \mathbb{R}^{|R|} \to \mathbb{R}$ are interpreted functions. The syntax of STL is given by the grammar

$$\varphi ::= \top \mid f(R) > 0 \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \, \mathcal{U}_I \, \varphi_2 \,,$$

where $f(R)$ are terms in Θ and I are real intervals with bounds in $\mathbb{Q}_{\ge 0} \cup \{\infty\}$. As customary we use the shorthands $\Diamond_I \varphi \equiv \top \mathcal{U}_I \varphi$ for *eventually* and $\Box_I \varphi \equiv \neg \Diamond_I \neg\varphi$ for *always*. The timing interval I may be omitted when $I = [0, \infty)$ or $I = (0, \infty)$.

The semantics of STL is captured by the relation $\models$ between a time t, signal w, and formula φ, given by induction as follows:

$$\begin{aligned}
&(w, t) \models \top \\
&(w, t) \models f(R) > 0 && \text{iff} && f(w_R[t]) > 0 \\
&(w, t) \models \neg\varphi && \text{iff} && (w, t) \not\models \varphi \\
&(w, t) \models \varphi_1 \vee \varphi_2 && \text{iff} && (w, t) \models \varphi_1 \text{ or } (w, t) \models \varphi_2 \\
&(w, t) \models \varphi_1 \, \mathcal{U}_I \, \varphi_2 && \text{iff} && \exists t' \in t \oplus I,\ (w, t') \models \varphi_2 \text{ and} \\
& && && \forall t'' \in (t, t'),\ (w, t'') \models \varphi_1 \,.
\end{aligned}$$

Here we use the symbol $\oplus$ to denote the Minkowski sum between t and I, defined as follows: $t \oplus I = \{t + a \mid a \in I\}$. We write $w \models \varphi$ when $(w, 0) \models \varphi$.

We now define *interface-aware signal temporal logic* (IA-STL), by augmenting STL with input/output declarations. Formally, an IA-STL specification is a tuple (X, Y, φ) such that $X \cap Y = \emptyset$, where

- φ is an STL formula over variables in S;
- $X \subseteq S$ is the set of the *input* variables;
- $Y \subseteq S$ is the set of the *output* variables.

We remark that while desirable, we do not require the set of signals to be partitioned into input and outputs, in that we allow for signals in $S \setminus (X \cup Y)$ that are neither input nor output. This may be relevant for internal state signals whose role is not clearly defined. This also ensures backward compatibility and in general makes IA-STL a conservative extension of STL.

[1] In practice, we assume piecewise linear or piecewise constant signals.

Relative Robustness. We introduce a notion of robustness specialized according to two subsets of variables $X, Y \subseteq S$ such that $X \cap Y = \emptyset$. Let φ be an STL formula and w a signal trace. We define the X-robustness relative to Y, denoted $\rho_X^Y(\varphi, w, t)$ by induction as follows:

$$\begin{aligned}
\rho_X^Y(\top, w, t) &= +\infty \\
\rho_X^Y(f(R) > 0, w, t) &= \begin{cases} 0 & \text{if } R \not\subseteq X \cup Y \\ f(w_R[t]) & \text{else if } R \not\subseteq Y \\ \operatorname{sign}(f(w_R[t])) \cdot \infty & \text{otherwise} \end{cases} \\
\rho_X^Y(\neg\varphi, w, t) &= -\rho_X^Y(\varphi, w, t) \\
\rho_X^Y(\varphi_1 \vee \varphi_2, w, t) &= \max\left\{\rho_X^Y(\varphi_1, w, t), \rho_X^Y(\varphi_2, w, t)\right\} \\
\rho_X^Y(\varphi_1 \, \mathcal{U}_I \, \varphi_2, w, t) &= \sup_{t' \in t \oplus I} \min\left\{ \begin{array}{c} \rho_X^Y(\varphi_2, w, t'), \\ \inf_{t'' \in (t, t')} \rho_X^Y(\varphi_1, w, t'') \end{array} \right\},
\end{aligned}$$

where for all $a \in \mathbb{R}$, $\operatorname{sign}(a) \cdot \infty = +\infty$ if $a > 0$, $-\infty$ otherwise.

We note that the standard STL robustness ρ as defined in [3, 5] can be recovered by letting $\rho(\varphi, w, t) = \rho_S^{\emptyset}(\varphi, w, t)$. The notion of robustness that we define instead measures the robustness only on the subset of variables X, relative to some variables Y. Variables in X are measured, and their robustness is given by the terms they appear in; variables in Y are considered fixed and their robustness is taken to be infinite; variables neither in X nor in Y are considered to take arbitrary values, and their robustness is zero.

Output Robustness and Input Vacuity. We now specialize the notion of relative robustness to the context of input and output signals. Let (X, Y, φ) be an IA-STL specification.

- We call *output robustness*, denoted μ, the Y-robustness relative to $S \setminus Y$. By definition $\mu(\varphi, w, t) \equiv \rho_Y^{S \setminus Y}(\varphi, w, t)$.
- We call *input vacuity*, denoted ν, the X-robustness relative to $\emptyset$. By definition $\nu(\varphi, w, t) \equiv \rho_X^{\emptyset}(\varphi, w, t)$.

Table 1: Possible combinations of output robustness and input vacuity values, where $k \in \mathbb{R}_{\ge 0}$, and their meaning.

$\mu(\varphi, w)$	$\nu(\varphi, w)$	Interpretation
$+\infty$	$+k$	vacuously true
$+k$	0	nonvacuously true
$-k$	0	nonvacuously false
$-\infty$	$-k$	vacuously false

The output robustness represents the robustness of the specification relative to the trace, measured at the output. When positive (negative), it indicates by how much the output can be changed without falsifying (satisfying) the formula for the given input. When infinity (negative infinity), the output robustness indicates that φ is vacuously satisfied (falsified) by the given input.

The input vacuity measure represents the level of vacuity of φ with respect to the given input. When positive (negative) it indicates by how much the input can be changed without falsifying

(satisfying) the formula, regardless of the output. When equal to 0, the input vacuity indicates that the formula is non-vacuously true or false, based on trace w.

Table 1 summarizes the meaning of possible combinations of input vacuity and output robustness.[2]

We verify that, by construction, the vacuity of φ relative to w only depends on input variables X. Consider an IA-STL specification (X, Y, φ) and let us assume without loss of generality that X is of the form $\{s_1, \ldots, s_m\}$.

Proposition 3.1. *For any $t \in \mathbb{T}$, $u : \mathbb{T} \to \mathbb{R}^m$ and $v, v' : \mathbb{T} \to \mathbb{R}^{n-m}$ we have $\nu(\varphi, u\|v, t) = \nu(\varphi, u\|v', t)$.*

Thus in the following for $u = w_X$, we freely write $\nu(\varphi, u, t)$ in place of $\nu(\varphi, w, t)$.

We motivate the definitions of input vacuity and output robustness in the following Examples 3.2 and 3.3 by comparing these notions to the classical notion of robustness from [3, 5].

Example 3.2. We consider the formula φ from Section 1 in which *req* is declared as an input and *gnt* as an output signal, as well as the trace w from Figure 1-(bottom). Using traditional robustness, $\rho(\varphi, w) = -1$ — instead of measuring how far the system is from generating a valid grant, traditional robustness measures the quality of the request. The value can be interpreted as the cheapest way to satisfy φ by lowering the signal *req* by 1 and effectively removing all requests. In contrast, $\mu(\varphi, w) = -3$ represents the measure of how far the system is from responding to the requests by valid grants.

Example 3.3. In this example, we study again the bounded response property φ from Section 1, and we evaluate it on the trace w from Figure 2. We first note that the output robustness $\mu(\varphi, w) = \infty$, indicating vacuous satisfaction of φ. In other words, there is no modification of *gnt* that can result in the violation of φ for the given fixed input *req*. In order to measure the level of vacuity of φ with respect to *req*, we use instead input vacuity, which yields $\nu(\varphi, w) = 2$. This measure means that any change in *req* smaller than 2 preserves vacuous satisfaction of φ.

Properties. Let us now formally establish the relation between input vacuity and output robustness on the one hand and some distance measures on the other hand. The results that we present also justify and explain the classification of Table 1. In more detail, we define two notions of distances between traces: based on input signals alone, and based on output signals for a fixed input. From these we derive notions of distances between a trace and a formula, seen as the (Hausdorff) distance to the nearest trace that satisfies the formula. These distances provide semantic, hence precise, counterparts of input vacuity and output robustness measures.

Let $u, u' : \mathbb{T} \to \mathbb{R}^m$ be signals over variables $\{s_1, \ldots, s_m\}$. The absolute distance $d(u, u')$ is defined as follows:

$$d(u, u') = \sup_{t \in \mathbb{T}, f(R) \in \Theta, v : \mathbb{T} \to \mathbb{R}^{n-m}} |f((u'\|v)_R[t]) - f((u\|v)_R[t])| \,.$$

[2]We remark that the meaning of $\mu(\varphi, w) = \nu(\varphi, w) = 0$ is ambiguous, since it can be interpreted as w satisfying or falsifying φ. This is related to the fact that in general the satisfaction/violation boundary may feature both satisfying and falsifying traces, so that borderline traces cannot be classified. We omit from the table the cases $\mu(\varphi, w) = \nu(\varphi, w) = \pm\infty$ as they can only happen when measuring robustness with formulas that are obvious tautologies and contradictions built using subformula $\top$, and are consequently of no practical interest.

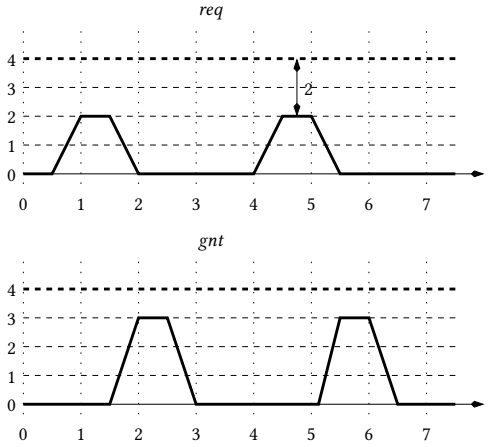

Figure 2: Vacuously satisfied request-grant property with signals *req* and *gnt*.

The absolute distance from u to some formula φ at time t, denoted $d(\varphi, u, t)$, is defined as follows:

$$d(\varphi, u, t) = \inf_{\substack{u' : \mathbb{T} \to \mathbb{R}^m,\ v : \mathbb{T} \to \mathbb{R}^{n-m} \\ (u'\|v, t) \models \varphi}} d(u'\|v, u\|v) \,.$$

Let $v, v' : \mathbb{T} \to \mathbb{R}^{n-m}$ be signals over variables $\{s_{m+1}, \ldots, s_n\}$. The distance $d_u(v, v')$ relative to signal u is defined as follows:

$$d_u(v, v') = \sup_{t \in \mathbb{T}, f(R) \in \Theta} |f((u\|v')_R[t]) - f((u\|v)_R[t])| \,.$$

The distance from u to some formula φ relative to v at time t, denoted $d_v(\varphi, u, t)$, is defined as follows:

$$d_v(\varphi, u, t) = \inf_{\substack{u' : \mathbb{T} \to \mathbb{R}^m \\ (u'\|v, t) \models \varphi}} d(u'\|v, u\|v) \,.$$

We next consider an IA-STL specification (X, Y, φ) and assume without loss of generality that X is of the form $\{s_1, \ldots, s_m\}$. We first show that the input vacuity $\nu(\varphi, u, t)$ for input signal u is a safe approximation of the absolute distance from u to φ (or $\neg\varphi$).

Lemma 3.4. *Let $u : \mathbb{T} \to \mathbb{R}^m$ be a signal and t be a time. We have $-d(\varphi, u, t) \leq \nu(\varphi, u, t) \leq d(\neg\varphi, u, t)$.*

As a result, ν captures the vacuity condition of the formula relative to the input signal and measures its level:

Theorem 3.5. *Let $u : \mathbb{T} \to \mathbb{R}^m$ be a signal.*

- *If $\nu(\varphi, u) > 0$ then $u\|v \models \varphi$ for all $v : \mathbb{T} \to \mathbb{R}^{n-m}$;*
- *If $\nu(\varphi, u) < 0$ then $u\|v \not\models \varphi$ for all $v : \mathbb{T} \to \mathbb{R}^{n-m}$.*

Theorem 3.6. *Let $u, u' : \mathbb{T} \to \mathbb{R}^m$ be signals such that $d(u, u') < |\nu(\varphi, u)|$. We have $u'\|v \models \varphi$ iff for all $v : \mathbb{T} \to \mathbb{R}^{n-m}$, $u\|v \models \varphi$.*

Next, we show that the output robustness $\mu(\varphi, w, t)$ is a safe approximation of the distance from u to φ (or $\neg\varphi$) relative to v.

Lemma 3.7. *Let $u : \mathbb{T} \to \mathbb{R}^m$ and $v : \mathbb{T} \to \mathbb{R}^{n-m}$ be signals and t be a time. We have $-d_u(\varphi, v, t) \leq \mu(\varphi, u\|v, t) \leq d_u(\neg\varphi, v, t)$.*

As a result, μ captures the satisfaction status of the formula for the combined input and output signals and measures the robustness of the formula in terms of the output signal:

Theorem 3.8. *Let $w : \mathbb{T} \to \mathbb{R}^n$ be a signal.*

- *If $\mu(\varphi, w) > 0$ then $w \models \varphi$;*
- *If $\mu(\varphi, w) < 0$ then $w \not\models \varphi$.*

Theorem 3.9. *Let $u : \mathbb{T} \to \mathbb{R}^m$ and $v, v' : \mathbb{T} \to \mathbb{R}^{n-m}$ be signals such that $d_u(v, v') < |\mu(\varphi, u\|v)|$. We have $u\|v \models \varphi$ iff $u\|v' \models \varphi$.*

Discussion. Request-response specifications of the form $\square(\alpha \to \beta)$ are commonly used to describe CPS requirements. The notion of vacuity in [22] is restricted to the above form. In this view, a vacuous trace is one that satisfies $\square(\neg\alpha)$, such that the request condition never being satisfied creates no response obligation.

Observe that it is not necessarily the case that α and β respectively range over input and output variables. Let x be an input and y an output signal. The specifications $\square((x \geq 0 \wedge y \geq 0) \Rightarrow \square_{[0,2]}(y \leq 5))$ and $\square(y \geq 0 \Rightarrow \square_{[0,2]}(y \leq 5))$ are both natural examples of request-response properties in which the future output obligations do not depend (only) on inputs, but (also) on the current state of the output. In these cases, our input vacuity measure is complementary to the definition of vacuity from [22]. Our notions of output robustness and input vacuity are general and can meaningfully apply to any STL formula conjoined with input and output declarations regardless of its syntactic form or semantic characteristics.

Signal variables can refer to quantities expressed in different units of measurement, and the corresponding signal units influence the robustness computation. In particular, the robustness value of a specification referring to multiple signal variables with different units is typically dominated by one of the signal variables. A change in the measurement unit of that signal variable can drastically change to robustness value by changing the variable values dominating the computation. This problem is orthogonal to the input/output characterization of signal variables, yet we remark that the notion of relative robustness that we introduce in this work provides the means to address this problem. In particular, relative robustness enables studying the effect of individual signals on the overall robustness with respect to a specification.

The relative robustness can also be used to define symmetrical notions of *input robustness* and *output vacuity*. The input robustness measures the robustness of the system in terms of margins at the input, for a fixed output. The output vacuity observes a vacuity condition related to the output obligations holding in the absence of input stimuli. We will refrain from further elaboration on these notions, as we found these measures less intuitive and less applicable as compared to the output robustness and input vacuity notions.

4 ROBUSTNESS-GUIDED TRACE DIAGNOSTICS

In this section, we present a two-part trace diagnostic procedure. The first part of the procedure is novel, we call it *worst-case diagnostics*. It marks the point (or set of points) corresponding to the worst-case value in the trace relative to the specification. The second part of the procedure is known from [19] as *epoch diagnostic*. It provides additional context, by highlighting all parts of the trace contributing to the violation/satisfaction.

Worst-Case Diagnostics. We define a procedure that, given a formula φ and trace w, returns the time(s) and signal variable(s) from which the robustness value $\rho(\varphi, w)$ originates. This *worst-case diagnostics*, denoted D_ρ, is obtained by induction on the structure of the formula and on time, based on the robustness values given by ρ. The operator D_ρ takes as argument a formula φ, trace w and time $t \in \mathbb{T}$, and returns a (set of) pair(s) in $\mathbb{T} \times S$, where $S = \{s_1, \ldots, s_n\}$ is the set of variables. It provides the time(s) and signal variable(s) that witness the robustness value of formula φ. Operator D_ρ is defined relative to some robustness indicator ρ that we take to be either μ, ν, or the standard robustness indicator of [3, 5]. For a given trace w, the worst-case diagnostics is computed by induction on the structure of the formula φ and on time t as follows. We let

$$
\begin{aligned}
D_\rho(\top, w, t) &= \emptyset \\
D_\rho(f(R) > 0, w, t) &= \{(t, r) \mid r \in R\} \\
D_\rho(\neg\varphi, w, t) &= D_\rho(\varphi, w, t) \\
D_\rho(\varphi \vee \psi, w, t) &= \begin{cases} D_\rho(\varphi, w, t) & \text{if } \rho(\varphi, w, t) > \rho(\psi, w, t) \\ D_\rho(\psi, w, t) & \text{if } \rho(\varphi, w, t) < \rho(\psi, w, t) \\ D_\rho(\varphi, w, t) \cup D_\rho(\psi, w, t) & \text{otherwise} \end{cases} \\
D_\rho(\varphi \,\mathcal{U}_I\, \psi, w, t) &= \begin{cases} D_\rho(\varphi, w, \tau_\varphi) & \text{if } \rho(\varphi, w, \tau_\varphi) < \rho(\psi, w, \tau_\psi) \\ D_\rho(\psi, w, \tau_\psi) & \text{if } \rho(\varphi, w, \tau_\varphi) > \rho(\psi, w, \tau_\psi) \\ D_\rho(\varphi, w, \tau_\varphi) \cup D_\rho(\psi, w, \tau_\psi) & \text{otherwise}, \end{cases}
\end{aligned}
$$

where $\tau_\psi = \operatorname{argmax}_{t' \in t \oplus I} \min\{\rho(\psi, w, t'), \inf_{t'' \in (t, t')} \rho(\varphi, w, t'')\}$ and $\tau_\varphi = \operatorname{argmin}_{t'' \in (t, \tau_\psi)} \rho(\varphi, w, t'')$.[3] We then let $D_\rho(\varphi, w) = D_\rho(\varphi, w, 0)$.

The following proposition makes it precise in what sense we can say that the worst-case diagnostics witnesses the robustness value of w relative to φ. Here $cl(\varphi)$ denotes the closure of φ, meaning φ and all of its sub-formulas.

Proposition 4.1. *Let w be a signal and φ a formula. For all $(t, r) \in D_\rho(\varphi, w)$, there exists $f(R) > 0 \in cl(\varphi)$ such that $r \in R$ and $|\rho(\varphi, w)| = |\rho(f(R) > 0, w, t)|$.*

Epoch Diagnostics. The epoch diagnostics procedure of [19] is based on the notion of *temporal implicant* of [26]. Informally, a temporal implicant is a property of the trace w that is a subset of the trace that entails φ. When $w \not\models \varphi$, to explain the violation of φ by w one uses instead an implicant of $\neg\varphi$.

We recover the epoch diagnostic by considering the *characteristic function* $\chi(\varphi, w, t) \in \{0, 1\}$, defined by $\chi(\varphi, w, t) = 1$ iff $(w, t) \models \varphi$. The epoch diagnostic is then obtained as $D_\chi(\varphi, w)$.

Example 4.2. We illustrate the combined worst-case and epoch trace diagnostics procedure with the STL specification φ from Section 1. We depict a signal that violates φ with diagnostics information overlaid in Figure 3. The worst-case diagnostics depends on the notion of robustness used. The circle and square markings denote the outcome of the worst case robustness procedure guided by the classical robustness and by the output robustness, respectively. In this instance, the epoch diagnostics indicates two periods responsible for the violation on each signal and the worst-case diagnostics indicates the following. The value of the classical robustness comes from the *req* signal during the first period, while the value

[3] In general, τ_ψ is a *set* thus $D_\rho(\varphi, w, \tau_\psi)$ stands for the union of $D_\rho(\varphi, w, t')$ for all t' in τ_ψ while $\rho(\psi, w, \tau_\psi)$ stands for the value of $\rho(\psi, w, t')$ for any t' in τ_ψ. The same remark applies to τ_φ, $D_\rho(\varphi, w, \tau_\varphi)$ and $\rho(\varphi, w, \tau_\varphi)$. The times at which we perform the diagnostics and evaluate the robustness may include limit points t^+ and t^- for $t \in \mathbb{T}$, as maxima and minima are sometimes found in the limit.

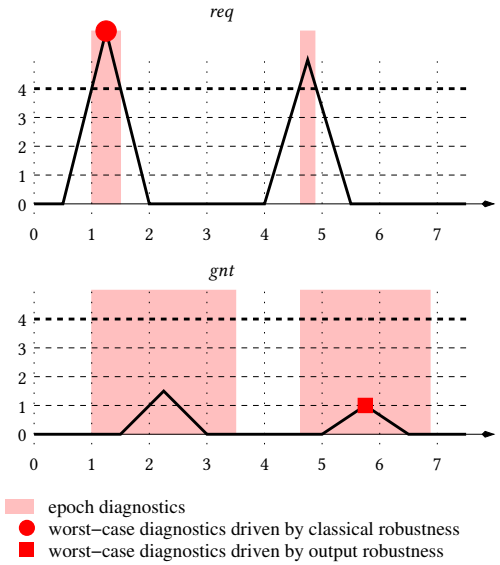

Figure 3: Combined epoch and worst-case diagnostics for the specification φ.

of the output robustness comes from *gnt* signal during the second period. The lightly shaded regions are the result of the epoch diagnostics computation and are independent of various definitions of robustness.

Discussion. The worst-case diagnostics procedure presented in this section is orthogonal to the notion of interface-aware specifications. A shortcoming of temporal logic robustness indicators, as argued in Section 1, is their opacity as to which signals are responsible for the observed robustness value. In Section 3, we partly remedied this situation by enabling the designer to provide *a priori* a subset of signal variables over which the robustness should be measured. This is particularly relevant when the specification describes an input/output relation, because of the different ways tolerance margins relate to the system robustness in that case. The worst-case trace diagnostics makes the robustness computation even more transparent by providing *a posteriori* signal variable(s) and time(s) from which the robustness value derives.

For interface-aware specifications the worst-case diagnostics makes it possible to identify input and output states that are responsible for a given input vacuity condition or output robustness value. A single time-variable pair may not be self-explanatory, and we chose to combine it with the epoch diagnostics of [19]. The epoch diagnostics provides additional context by highlighting which parts of the signal trace contribute to the observed violation. We believe the combined information of epoch and worst-case diagnostics increases the usability of interface-aware robustness indicators.

5 APPLICATIONS AND EVALUATION

We implemented in Breach (1) the STL language extension, adding the ability to declare the input/output interface of the specification, (2) procedures for computing the output robustness and input vacuity, (3) a procedure for computing combined epoch and worst-case fault explanation. Below we describe applications of IA-STL to address several common engineering tasks for two different model-based applications. The first application is a powertrain control (PTC) system. We use this publicly-available benchmark model to explore different uses of IA-STL and illustrate its benefits. The second application is a system model based on an automotive, hydrogen fuel cell (FC) system. We use this proprietary industrial model to validate the proposed approach.

5.1 Powertrain Benchmark System

The powertrain control (PTC) benchmark model, described in detail in [27], represents the dynamics associated with the control of the engine *air path* for an internal combustion engine used in an automobile. The air path model captures the dynamics of the air and fuel that pass into the engine, effects of combustion, and the expulsion of exhaust gases. A computer controller regulates the amount of air and fuel injected into the engine cylinders; the goal is to accurately control the ratio of air-to-fuel that enters the cylinders. Accurate control of this system is critical to ensure overall vehicle efficiency, to regulate the hydrocarbon emissions, and to maintain a high quality of vehicle response — the so-called *driveability*. The model is a *mean-value* engine model, meaning that it represents the dynamics of the fuel and air processes that take place in the engine cylinders across several combustion cycles; it does not represent individual combustion events that occur in the cylinders.

Inputs to the PTC air path system are accelerator pedal angle θ_{in} and engine speed ω. The output from the system model is the measured air-to-fuel ratio λ. Consider an overshoot requirement for the PTC system, adapted from the one described in [20]:

$$\varphi_{overshoot} \equiv \square_{[a,+\infty)}((\theta'_{in} - \theta_{in} > c) \Rightarrow \square_{[0,b]}(|\lambda - \lambda_{ref}| < \gamma \cdot \lambda_{ref}))\,.$$

Here γ defines the maximum allowed overshoot value relative to the reference value λ_{ref}, and c is a threshold value used to determine when θ_{in} produces a *step* input. The additional input signal θ'_{in} is shorthand for the value of θ_{in} shifted by d time units, such that $\theta'_{in}[t] = \theta_{in}[t+d]$ for all t. The antecedent clause of the implication in $\varphi_{overshoot}$ determines when a step occurs in the input, using a small time delay d to decide whether the input is in a step condition. The consequent clause indicates that the amount by which the output λ overshoots the reference value λ_{ref} should not exceed γ times the reference value. Here, b is the period over which the overshoot value should be monitored after a step in input is experienced. Note that the property is only checked after time $t = a$; before this time the system is in a transient state. For our experiments, we use $\gamma = 0.01$, $\lambda_{ref} = 14.7$, $a = 10.0$ sec., $b = 2.0$ sec., $c = 10.0$ deg., and $d = 0.1$ sec.

5.1.1 Robustness computation. We illustrate the difference between classical robustness, output robustness, and input vacuity on the PTC model. Figure 4 depicts a behavior of the PTC model, which satisfies the specification $\varphi_{overshoot}$ with:

- Classical robustness: 0.08;
- Output robustness: ∞;
- Input vacuity: 0.05.

We observe that the classical robustness measures how far λ is from reaching the overshoot condition. We recall that there is no

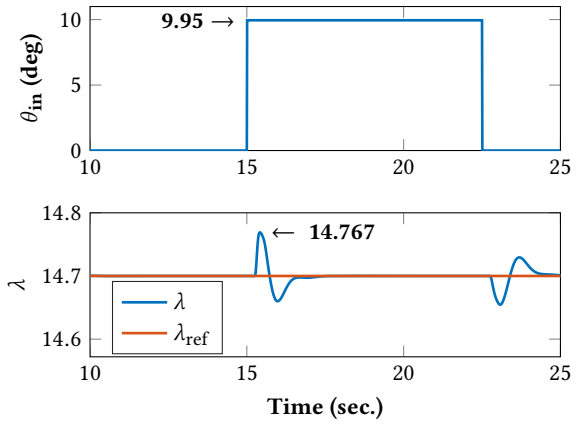

Figure 4: PTC robustness - an example of vacuity.

overshoot when $|\lambda - \lambda_{ref}| < \lambda_{ref} \cdot \gamma$. The maximal value of λ during the disturbance is 14.767. It is easy to see that λ is 0.08 away from an overshoot, the value reported by the classical robustness.

However, this value is misleading. If we observe closer the input θ_{in}, we can notice that steps go from 0.0 to 9.95, instead of going to 10.0, as required by the specification. As a result, no obligation is triggered in the output and the specification is vacuously satisfied by the input. In other words, with this specific input, the specification is guaranteed to hold regardless of what is observed in the output. This fact is appropriately reflected by the ∞ value of the output robustness. Finally, input vacuity of 0.05 indicates that a change of at least 0.05 in θ_{in} is needed to trigger any obligation in the output.

We remark that in this example, the classical robustness differs from both the output robustness and the input vacuity. In fact, it coincides with the value of the *output vacuity* measure that we mention in Section 3. Intuitively, the output vacuity measures the *strength* of the output with respect to the specification regardless of the input (and its vacuity). We believe that all such robustness measures are independent, in that there are situations where they all differ from each other on the same formula-trace pair.

5.1.2 Falsification. As the next application of IA-STL, we consider the falsification problem. Given the specification $\varphi_{overshoot}$ and the system model, the task is to identify an input signal θ_{in} that results in behaviors that do not satisfy $\varphi_{overshoot}$. We assume the class of input signals for θ_{in} to be piecewise constant signals with two discontinuities. The falsification problem consists in solving a search problem over two decision variables to identify a behavior that does not satisfy $\varphi_{overshoot}$. We restrict the search to a maximum of 100 iterations.

In the first experiment, we initialize the input signal θ_{in} to a step signal that rises from 0 to 10.1 degrees. This input satisfies the left-hand side of the implication in $\varphi_{overshoot}$ and non-vacuously exercises the requirement. We run two instances of the falsification problem with this initial condition. In the first instance, the cost function is the classical robustness, and in the second instance, the cost function is the output robustness. In this experiment, the classical robustness value is dominated by the output signal λ, and the two robustness measures coincide. As a consequence, the two falsification problem instances yield the identical result, identifying the same input that leads the system to violate the specification.

In the second experiment, we change the unit of θ_{in} from degrees to *revolutions*, where 1 revolution equals 360 degrees, and we scale the threshold c accordingly to $c = \frac{10}{360} = 0.028$. We note that the change of units does not affect the meaning of the specification in any way. We repeat the two falsification instances from the first experiment, and we observe an interesting phenomenon. The classical robustness applied to the initial simulation is now dominated by the measurements over the input θ_{in} and is equal to 0.028. This value corresponds to the segments of the input where θ_{in} equals to 0, and in these parts of the behavior, the implication in $\varphi_{overshoot}$ is satisfied and is exactly $c = 0.028$ away from violating the property. In fact, the classical robustness is dominated by the 0.028 measurement over the input, regardless of the size of the input step. The optimizer tries to find a good search direction by varying the size of the step but remains stuck in a plateau and is hence not able to find a violating trace. This is exposed by our experimental results, shown in Figure 5-(left), where all 100 iterations in the falsification procedure have the robustness value of 0.028.

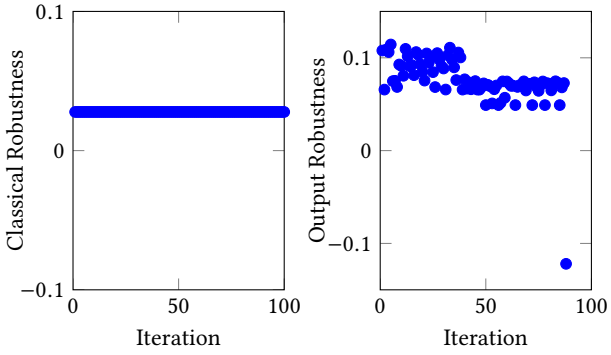

Figure 5: PTC falsification with (left) classical robustness and (right) output robustness as cost function.

In contrast, the falsification instance that uses the output robustness as its cost function ignores the quantitative measurements over the input. It appears to follow a gradient that eventually leads to a violating trace after 88 iterations of the falsification algorithm, as shown in Figure 5-(right).

5.1.3 Fault localization. In this section, we illustrate our combined epoch and worst-case fault explanation procedure applied to the PTC system. Figure 6 shows a trace in which the joint behavior of θ_{in} and λ lead to the violation of the overshoot requirement, yielding an output robustness of -0.203. The epoch trace explanation marks in the simulation trace a step in the pedal angle θ_{in} and the ensuing overshoot region in λ, occurring within 2.0 sec. The worst-case fault explanation identifies the worst-case of the overshoot with the red box, marking the time where λ reaches 15.05. This is the exact point in time where the absolute difference between λ and λ_{ref} is $|15.05 - 14.7| = 0.35$, which is by 0.203 higher than the allowed value of $\gamma \cdot \lambda_{ref} = 14.7 \cdot 0.01 = 0.147$.

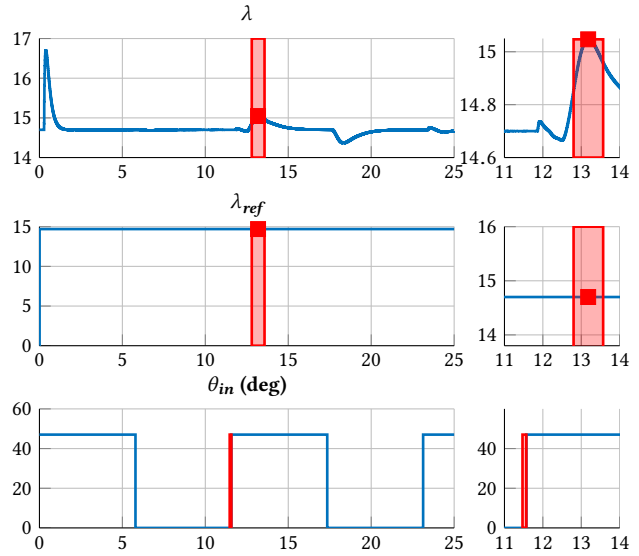

Figure 6: PTC fault explanation with zoom in on the worst behavior.

5.2 Fuel Cell System

The air path controller for the automotive hydrogen fuel cell (FC) system is a complex model made for a production development with close to 4,000 blocks and 400 discrete and continuous states. It is described in [28]. The FC system uses a mixture of air and hydrogen to produce electrical power. The FC *stack* is a system that takes hydrogen gas and air as input and produces electrical energy, which is used to charge the battery and power the motor. The air path controller is used to regulate how much air enters the FC stack. More precisely, the job of the air path controller is to regulate the amount of volumetric air flow and air pressure that is imparted to the FC stack. Accurate regulation is required to ensure sufficient power output from the FC stack and to ensure a high level of performance (driveability) from the vehicle.

The expected behavior is specified with 23 STL requirements.[4] In this section, we illustrate our results on a requirement R that specifies a criterion on how much a control system response is allowed to deviate from a reference behavior. The control system is given the signal request *REQUEST* and generates a response signal *RESPONSE*. Whenever a specific condition on the request signal occurs, represented by the Boolean predicate *cond*(*REQUEST*), the value of the signal *RESPONSE* is checked against its associated reference signal *CRITERIA*. The STL requirement is given by

$$\varphi \equiv \square(\mathit{CHECK\_FLAG} \rightarrow \mathit{RESPONSE} \geq \mathit{CRITERIA}),$$

where $\mathit{CHECK\_FLAG} \equiv \mathit{cond}(\mathit{REQUEST})$.

5.2.1 Robustness Sensitivity Analysis with Heat Maps. The model of the air path controller is parameterized with three parameters p_1, p_2 and p_3. These parameters affect the controller behavior and consequently the degree to which the controller satisfies or violates its requirements. Given parameter values a, b and c of p_1, p_2 and p_3, we denote by $w^{(a,b,c)}$ the signal resulting from the simulation of the system model with these parameter values.

The sensitivity of the model robustness to its requirements is studied by uniformly varying these three parameters, simulating the model for each combination of parameters and monitoring the simulation outcomes against the STL requirements. Figure 7 shows a 3D scatter plot, where each point denotes the satisfaction/violation status of φ for the given simulation trace and the parameter values.

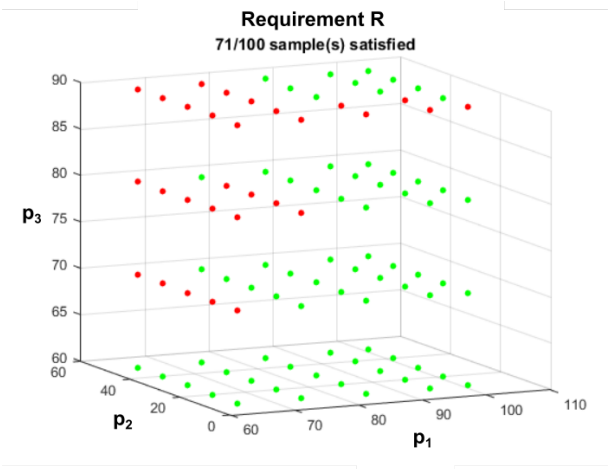

Figure 7: Satisfaction/violation of φ for specific combinations of parameter values.

In addition to the qualitative impact of parameter variation to the satisfaction of φ, one can perform a more quantitative sensitivity analysis by calculating the robustness degree of the simulation results with respect to the specification for each choice of parameter values. The heat map obtained by computing the robustness degree $\rho(\varphi, w^{(a,b,90)})$ is shown in Figure 8. Neither the scatter plot nor the heat map informs us about the robustness of the FC system. This is because the unitless Boolean input *CHECK_FLAG* with a normalized range of 1.0 dominates the computation. T is 1.0: the reported overall robustness is either 0.5 if $w^{(a,b,90)}$ satisfies φ, or -0.5 otherwise. One way to circumvent the problem is to manually scale input signals so that they do not dominate the computation. Instead we provide a general solution to this problem via the appropriate notion of *output robustness*.

We repeat the same experiment in which we declare *REQUEST* as an input and *RESPONSE* as an output signal, and compute the relative output robustness $\mu(\varphi, w^{(a,b,90)})$. Figure 9 shows the resulting heat map. We can see that the output robustness measures the quality of the system output, providing more interesting and intuitive information about the system performance.

5.2.2 Fault localization. In this section we validate the enhanced fault localization method on the FC model. For demonstration purposes we choose a simulation trace that violates the requirement φ and contrast the impact of the robustness measure on the fault localization procedure.

[4] Details regarding the requirements, physical meaning of signals, and units are suppressed for proprietary reasons.

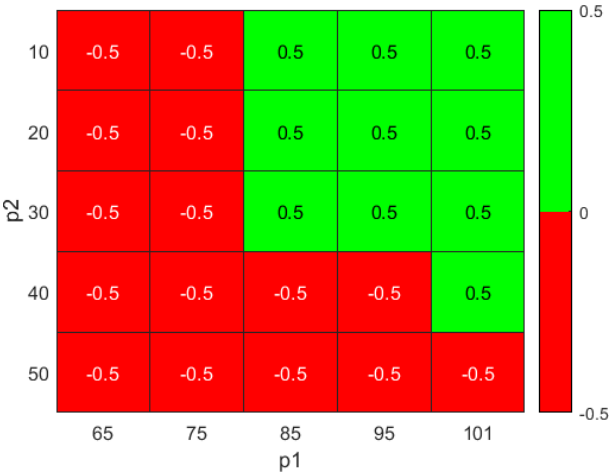

Figure 8: Heatmap for the Fuel Cell system, requirement R and $p_3 = 90$, using classical STL robustness computation.

Figure 9: Heatmap for the Fuel Cell system, requirement R and $p_3 = 90$, using IA-STL output robustness computation.

Figure 10 depicts the outcome of the fault localization when worst-case diagnostics is driven by classical STL robustness. The overall robustness value is dominated by the Boolean input, hence the fault localization procedure identifies points in the input as reasons for the worst-case robustness.

By contrast, combining fault localization with the IA-STL output robustness, as shown in Figure 11, allows our fault localization procedure to correctly identify the precise time point that is responsible for the worst-case deviation of the output from the reference.

5.2.3 Falsification. Finally, we illustrate the potential benefits of using the IA-STL output robustness computation to drive a falsification procedure. We compare falsification results performed with the traditional STL robustness and enhanced IA-STL output robustness computations. We fix an input trace for an FC system model and vary the system parameters p_1, p_2 and p_3 to attempt to falsify the requirement φ. For both cases, we use a search procedure based

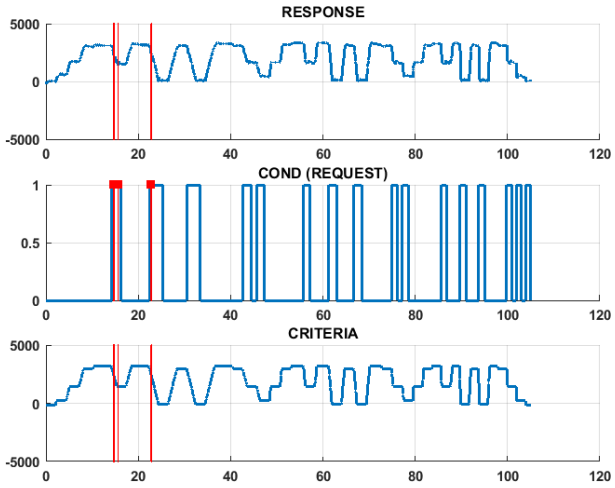

Figure 10: Fault localization, driven by traditional STL robustness.

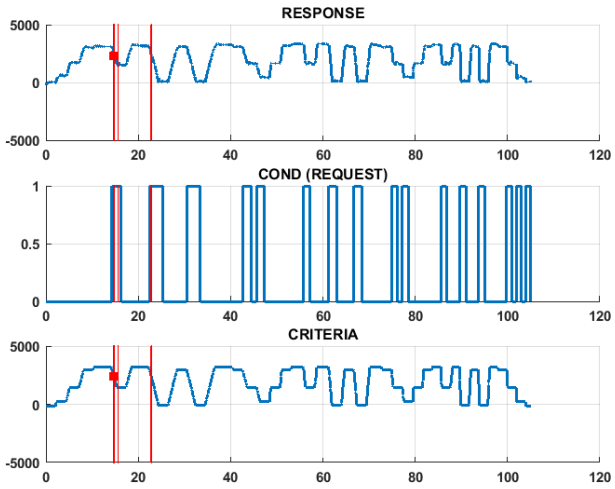

Figure 11: Fault localization, driven by the IA-STL output robustness.

on the MATLAB® local search optimizer `fmincon`. The search procedure as implemented in Breach uses a combination of the local search method provided by `fmincon` with random restarts when a local optimum is detected.

Figure 12-(left) illustrates the falsification results using traditional falsification robustness computations. The figure shows the evolution of the STL robustness across many iterations of the search procedure. We can observe that the solver either exhibits the hallmarks of random restarts (discontinuities) or is flat. Also, the majority of the flat regions are at the 0.5 value; this is due to the fact that the Boolean input dominates the robustness value when computed using the traditional method. This value does not convey information about the behavior of the system output.

By contrast, consider the results of the falsification procedure using the IA-STL output robustness computations, as shown in

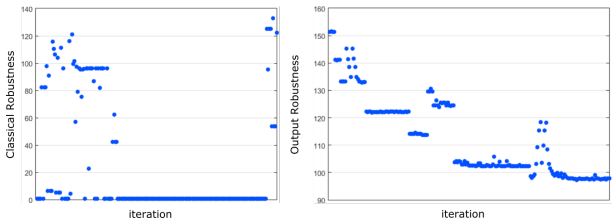

Figure 12: Falsifcation results, using classical STL (left) and output IA-STL (right) robustness.

Figure 12-(right). We can see in the figure that there appear to be fewer random restarts.

Despite the fact that we do not falsify the property using either traditional STL robustness or the enhanced version, the above example demonstrates that the STL output robustness computation provides more meaningful cost information that could potentially be exploited by an appropriate solver to better guide the search.

6 CONCLUSION AND FUTURE WORK

In this work, we enhanced signal temporal logic (STL) with support to define input-output interfaces. We adapted robustness computations to exploit this interface and consequently provide deeper insights into the system behaviors with respect to their specifications. We demonstrated the value of the enhanced robustness notions by using them to perform three analysis activities: robustness sensitivity analysis, fault localization, and search-based testing. We illustrated these analysis approaches on two examples, including an industrial automotive fuel cell system.

We plan to extend the presented work in several directions. The input/output signature is a natural pre-requisite for compositional reasoning about the system and its requirements. We will in particular study how we can leverage IA-STL to do compositional testing. We have seen that output robustness can be used to precisely identify portions of the input and output signals that explain the violation of a specification. We will study how this information could be used to explain the reason for the violation in terms of the system model.

ACKNOWLEDGMENTS

The authors would like to thank Arthur Wu and Jared Farnsworth from Toyota Motor North America for their help in understanding and using the hydrogen fuel cell model and for many helpful discussions. This research was supported in part by the Austrian Science Fund (FWF) under grants S11402-N23 (RiSE/SHiNE) and Z211-N23 (Wittgenstein Award).

REFERENCES

[1] J. Kapinski, J. V. Deshmukh, X. Jin, H. Ito, and K. Butts, "Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques," *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 45–64, Dec 2016.

[2] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems (FORMATS/FTRTFT)*, 2004, pp. 152–166.

[3] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications," in *Formal Approaches to Software Testing and Runtime Verification, First Combined International Workshops, FATES 2006 and RV 2006, Seattle, WA, USA, August 15-16, 2006, Revised Selected Papers*, 2006, pp. 178–192.

[4] ——, "Robustness of temporal logic specifications for continuous-time signals," *Theor. Comput. Sci.*, vol. 410, no. 42, pp. 4262–4291, 2009.

[5] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *Formal Modeling and Analysis of Timed Systems (FORMATS)*, 2010, pp. 92–106.

[6] A. Donzé, T. Ferrère, and O. Maler, "Efficient robust monitoring for STL," in *International Conference on Computer Aided Verification*. Springer, 2013, pp. 264–279.

[7] A. Donzé, "Breach, A toolbox for verification and parameter synthesis of hybrid systems," in *Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings*, 2010, pp. 167–170.

[8] Y. Annpureddy, C. Liu, G. E. Fainekos, and S. Sankaranarayanan, "S-taliro: A tool for temporal logic falsification for hybrid systems," in *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2011, pp. 254–257.

[9] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Falsification of LTL safety properties in hybrid systems," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2009, pp. 368–382.

[10] T. Nghiem, S. Sankaranarayanan, G. Fainekos, F. Ivancić, A. Gupta, and G. J. Pappas, "Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems," in *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*. ACM, 2010, pp. 211–220.

[11] W. Li, A. Forin, and S. A. Seshia, "Scalable specification mining for verification and diagnosis," in *Proceedings of the 47th design automation conference*. ACM, 2010, pp. 755–760.

[12] E. Bartocci, L. Bortolussi, and G. Sanguinetti, "Data-driven statistical learning of temporal logic properties," in *Formal Modeling and Analysis of Timed Systems (FORMATS)*, 2014, pp. 23–37.

[13] Z. Kong, A. Jones, and C. Belta, "Temporal logics for learning and detection of anomalous behavior," *IEEE Trans. Automat. Contr.*, vol. 62, no. 3, pp. 1210–1222, 2017.

[14] E. Asarin, A. Donzé, O. Maler, and D. Nickovic, "Parametric identification of temporal properties," in *Runtime Verification*, 2011, pp. 147–160.

[15] H. Yang, B. Hoxha, and G. Fainekos, "Querying parametric temporal logic properties on embedded systems," in *IFIP International Conference on Testing Software and Systems*. Springer, 2012, pp. 136–151.

[16] X. Jin, A. Donzé, J. V. Deshmukh, and S. A. Seshia, "Mining requirements from closed-loop control models," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 34, no. 11, pp. 1704–1717, 2015.

[17] A. Bakhirkin, T. Ferrère, and O. Maler, "Efficient parametric identification for STL," in *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week)*. ACM, 2018, pp. 177–186.

[18] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J. Raclet, P. Reinkemeier, A. L. Sangiovanni-Vincentelli, W. Damm, T. A. Henzinger, and K. G. Larsen, "Contracts for system design," *Foundations and Trends in Electronic Design Automation*, vol. 12, no. 2-3, pp. 124–400, 2018.

[19] E. Bartocci, T. Ferrère, N. Manjunath, and D. Nickovic, "Localizing faults in simulink/stateflow models with STL," in *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week), HSCC 2018, Porto, Portugal, April 11-13, 2018*, 2018, pp. 197–206.

[20] J. Kapinski, X. Jin, J. Deshmukh, A. Donze, T. Yamaguchi, H. Ito, T. Kaga, S. Kobuna, and S. Seshia, "ST-Lib: A library for specifying and classifying model behaviors," SAE Technical Paper, Tech. Rep., 2016.

[21] T. Akazaki, "Falsification of conditional safety properties for cyber-physical systems with gaussian process regression," in *Runtime Verification - 16th International Conference, RV*, 2016, pp. 439–446.

[22] A. Dokhanchi, S. Yaghoubi, B. Hoxha, and G. Fainekos, "Vacuity aware falsification for MTL request-response specifications," in *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, Aug 2017, pp. 1332–1337.

[23] I. Beer, S. Ben-David, C. Eisner, and Y. Rodeh, "Efficient detection of vacuity in ACTL formulaas," in *Computer Aided Verification, 9th International Conference, CAV*, 1997, pp. 279–290.

[24] O. Kupferman and M. Y. Vardi, "Vacuity detection in temporal model checking," in *Correct Hardware Design and Verification Methods (CHARME)*, 1999, pp. 82–96.

[25] T. Ball and O. Kupferman, "Vacuity in testing," in *Tests and Proofs, Second International Conference, TAP*, 2008, pp. 4–17.

[26] T. Ferrère, O. Maler, and D. Nickovic, "Trace diagnostics using temporal implicants," in *Automated Technology for Verification and Analysis*, 2015, pp. 241–258.

[27] X. Jin, J. V. Deshmukh, J. Kapinski, K. Ueda, and K. Butts, "Powertrain Control Verification Benchmark," in *Proc. of Hybrid Systems: Computation and Control*, 2014, pp. 253–262.

[28] A. Adimoolam, T. Dang, A. Donzé, J. Kapinski, and X. Jin, "Classification and coverage-based falsification for embedded control systems," in *Computer Aided Verification*, R. Majumdar and V. Kunčak, Eds. Cham: Springer International Publishing, 2017, pp. 483–503.