An Algebraic Framework for Runtime Verification

Stefan Jakšić*†, Ezio Bartocci†, Radu Grosu†, and Dejan Ničković*
*AIT Austrian Institute of Technology, Austria †Faculty of Informatics, TU Wien, Austria

***Abstract*—Runtime verification (RV) is a pragmatic and scalable, yet rigorous technique, to assess the correctness of complex systems, including cyber-physical systems (CPS). Modern RV tools also allow to measure the distance of a CPS behavior from a given formal requirement, thus, to quantify the robustness of a CPS with respect to perturbations caused by the physical environment.**

In this paper we propose Algebraic Runtime Verification (ARV), a general, semantic framework for correctness and robustness monitoring. ARV implements an abstract monitoring procedure, in which the specification language can be instantiated with various qualitative and quantitative semantics. This allows us to expose the core aspects of RV, by separating the monitoring algorithm from the concrete choice of the specification language and its semantics. We demonstrate the effectiveness of our framework on two examples from the automotive domain.

I. Introduction

Verification of realistic cyber-physical systems is still a challenge, as CPS exhibit both discrete and continuous dynamics, software (SW) and hardware (HW) components, complex communication between sub-systems, and sophisticated interactions with the physical environment. Runtime verification provides a pragmatic, yet rigorous solution, for assessing CPS correctness at runtime.

RV for CPS [10], centered around Signal Temporal Logic (STL) [38], has recently achieved noticeable success. STL extends Metric Temporal Logic (MTL) [37] with predicates over real-valued variables. The monitoring algorithms for STL were first developed in [38], [39] and implemented in a tool [43], giving thus STL a practical purpose in the analysis of realistic CPS.

In a CPS context, RV measures the satisfaction robustness of a CPS run with respect to a specification [1], [5], [23]. Most measures were associated to STL specifications [1], [5], [23]–[27], [30] and to its variants or extensions [3], [14]. The general approach was to first develop the measure, based on the STL syntax, and then use the measure for monitoring. This resulted in a plethora of measures, assessing space [26], [30], time [26], and averaging [3] robustness, respectively. A discrete-time, weighted, edit-distance measure was proposed in [36], and a space-robustness measure for Signal Regular Expressions (SRE) in [6].

The proliferation of measures reflects the needs of various applications, and the limitations of STL-based measure definitions. We propose Algebraic Runtime Verification (ARV), as a general, semantic approach to correctness and robustness monitoring of CPS. ARV simplifies and unifies qualitative and quantitative approaches to the RV of CPS. It exposes the core of the RV problem, and its underlying structure, allowing us to develop an abstract monitoring algorithm, that we instantiate with different specification languages, as well as qualitative and quantitative semantics. The use of automata, admits semantic and precise monitoring procedures, that are invariant to different syntactic representations of the same specification. It also enables flexible code generation that can directly translate the abstract monitoring procedure into real-time monitors, implemented in SW, embedded SW, or HW.

ARV starts from a regular specification formalism that admits an effective translation into symbolic automata. It then decorates the symbolic acceptor with weights and associates a semiring to the resulting symbolic weighted automaton (SWA). The weights in the SWA measure the distance at a given point in time between the current trace observation and the constraints induced by the specification. ARV defines the monitoring procedure over the SWA as a dynamic programming algorithm that computes the shortest path induced by an input trace. By instantiating the semiring, it provides various qualitative and quantitative semantics to the monitoring procedure without changing the underlying algorithms. We study the basic properties of our generic ARV framework and evaluate it with two case studies from the automotive domain.

The rest of the paper is organized as follows. In Section 2 we discuss related work. In Section 3 we introduce the theoretical background for formalizing ARV in Section 4. Section 5 evaluates our approach on two examples from the automotive domain, and illustrates the precision of ARV compared to tools that implement syntactic-based robustness degree. In Section 6 we draw conclusions and discuss numerous possibilities for future work. We prove soundness and completeness of ARV in Section 7.

II. Related Work

The theoretical and practical concerns regarding symbolic automata and transducers are studied in [19], [50]. Our ARV framework is based on the theory developed for symbolic automata.

An algebraic framework for the basic properties of the weighted automata is investigated in [41]. The shortest distance problem in weighted automata is studied in [40]. It generalizes the Bellman-Ford algorithm [12] to non-idempotent semirings. We are interested in a Hausdorff measure and hence restrict our attention to additively idempotent semirings. In contrast to fixed edge weights, we use SWA with weights which depend on current trace valuation.

Qualitative monitoring for temporal specifications has been a well-studied problem in the runtime verification community. In [11], the authors propose an automata-based approach to monitoring LTL and its real-time extension TLTL with the three-valued semantics. Another automata-based approach to online monitoring of LTL for Java software instrumented

This article was presented in the International Conference on Embedded Software 2018
and appears as part of the ESWEEK-TCAD special issue.

with AspectJ is presented in [49]. The authors in [8] present a rule-based software monitoring framework for LTL and metric LTL. In the context of hardware, monitors synthesis from temporal specifications such as Property Specification Language (PSL), has been extensively studied in the last years, resulting in tools FoCs [16] and MBAC [13]. Finkbeiner et al. in [32] present a procedure for synthesizing monitor circuits from LTL, while in [35] we provide a framework to generate hardware runtime monitors implementing the qualitative semantics of STL in field programmable gate arrays. Offline monitoring for STL with qualitative semantics was developed in [38], [39]. In contrast to these works, which aim at developing optimal monitoring procedures for specific settings, our primary focus in this paper is to develop a general framework that unifies RV of qualitative and quantitative semantics.

Quantitative semantics for temporal logics based on the uniform-norm were studied in [26], [30], [45]. Spatial robustness monitoring is implemented in S-TaLiRo [5] and Breach [24] tools. The spatial robustness was complemented with time robustness in [26] and with a combined time-space robustness based on (ϵ, τ)-similarity in [2]. In [3], the authors extend STL with *averaged* temporal operators. In contrast to our work, these approaches consider STL interpreted over continuous time. Continuous and discrete time interpretations of temporal logic specifications are complementary, and the relation between the two was studied in [29]. Determining robustness of hybrid systems using self-validated arithmetics is shown in [31]. The weighted Hamming and edit distances between behaviors are proposed in [48], where the authors use it to develop procedures for reasoning about the Lipshitz-robustness of Mealy machines and string transducers. The authors of [17] propose an online monitoring procedure where semantics describe the relation between input and output streams. In [9] the authors have introduced an algebraic approach to define the robustness for a spatio-temporal extension of STL. Although their framework enables also to plugin different (both multiplicative and additively) idempotent semirings generating different semantics, here we show that the use of additively idempotent semirings requires to take into account important considerations. Furthermore, the implemented monitoring procedure in [9] operates inductively on the structure of the formula, with the consequence to produce different results when we interpret quantitatively formulas that are logically equivalent.

Similarly to our work, [46] explores different interpretation of TL operators. However, these interpretations are applied directly on the syntax and semantics of the logic, aiming to demonstrate a relation between TL operators and convolution. We mention the work on quantitative languages [15] over infinite words, complementary to ours.

The problem of online robustness monitoring was studied more recently in [21], [22]. The authors of [22] propose a predictor-based online monitoring approach, in contrast to our black-box view of monitoring. In [21], the authors propose an interval-based approach of online evaluation that allows estimating the minimum and the maximum robustness with respect to both the observed prefix and unobserved trace suffix. Instead, ARV gives the distance of the trace prefix from the specification at every point in time.

III. Background

We first introduce the background needed to develop our algebraic runtime verification algorithm: semirings, metric spaces and distances, specification languages, symbolic automata and their weighted extension.

A. Semirings

Semirings are one of the most important algebraic structures, laying the foundation to both continuous and discrete mathematics. They help finding similarities between the two domains, even in places where these are not at all obvious.

Definition 1 (Semiring)**.** *A* semiring $\mathcal{S}$ *is a tuple* $(S, \oplus, \otimes, e_\oplus, e_\otimes)$*, where* S *is a set with two binary operations,* addition *(*$\oplus$*) and* multiplication *(*$\otimes$*), and two identity elements,* $e_\oplus$ *and* $e_\otimes$*, such that:*

- $(S, \oplus, e_\oplus)$ *is a commutative monoid with identity element* $e_\oplus$*;*
- $(S, \otimes, e_\otimes)$ *is a monoid with identity element* $e_\otimes$*;*
- $\otimes$ *distributes over* $\oplus$*; and*
- $e_\oplus$ *is an annihilator element for* $\otimes$*.*

We say that a semiring is *commutative* if the $\otimes$-multiplication operation is commutative. A semiring is said to be *additively (multiplicatively) idempotent* if for all $s \in S$, we have that $s \oplus s = s$ ($s \otimes s = s$). We say that a semiring is *idempotent* if it is both additively and multiplicatively idempotent. We say that a semiring is *bounded* if $e_\otimes$ is an annihilator element for $\oplus$. We note that a bounded semiring is also additively idempotent [40].

Example 1. *We depict in Table I several examples of semiring structures that we use in this paper. We note that the Boolean, MinMax and powerset semirings are both commutative and idempotent. The tropical semiring is commutative and additively idempotent. All four semirings are bounded. Probability semiring is neither multiplicatively not additively idempotent. Note that we use a non-standard definition of the Boolean and MinMax semirings, in which* $\oplus$ *corresponds to* $\wedge$ *and* $\min$*, while* $\otimes$ *corresponds to* $\vee$ *and* $\max$*.*

Semiring	S	$\oplus$	$\otimes$	$e_\oplus$	$e_\otimes$
Boolean	$\{0,1\}$	$\wedge$	$\vee$	1	0
MinMax	$\mathbb{R}_+ \cup \{\infty\}$	$\min$	$\max$	∞	0
Tropical	$\mathbb{R}_+ \cup \{\infty\}$	$\min$	$+$	∞	0
Probability	$\mathbb{R}_+ \cup \{\infty\}$	$+$	$\times$	0	1
Powerset	$\mathcal{P}(T)$	$\cap$	$\cup$	T	$\emptyset$

TABLE I
EXAMPLES OF SEMIRINGS.

In this paper, we restrict our study to additively idempotent semirings. In addition, we are motivated by CPS applications in which behaviors are sequences of real values. As a consequence, we consider semirings that are defined over sets that can be embedded in the set of reals. In particular, we focus on the Boolean, MinMax and tropical semirings from Table I in the remainder of the paper.

This article was presented in the International Conference on Embedded Software 2018 and appears as part of the ESWEEK-TCAD special issue.

Definition 2 (Natural order on S)**.** *Let $(S, \oplus, \otimes, e_\oplus, e_\otimes)$ be an additively idempotent semiring. We define the* natural order *on S as the relation $\sqsubseteq$: $(a \sqsubseteq b) \leftrightarrow (a \oplus b = a)$. If $(S, \oplus, \otimes, e_\oplus, e_\otimes)$ is also multiplicatively idempotent, we additionaly require: $(a \sqsubseteq b) \leftrightarrow (a \otimes b = b)$.*

Lemma 1 ([40])**.** *Let $(S, \oplus, \otimes, e_\oplus, e_\otimes)$ be an additively idempotent semiring. The natural order $\sqsubseteq$ on S defines a partial order.*

We now define the monotonicity of semirings, an important property that will allow us factoring and simplifying RV operations.

Definition 3 (Negative and monotonic semirings)**.** *Let $(S, \oplus, \otimes, e_\oplus, e_\otimes)$ be a semiring. We say that S is* negative *if $e_\otimes \sqsubseteq e_\oplus$. We say that S is* monotonic *if for all $a, b, c \in S$:*

1) $a \sqsubseteq b \rightarrow (a \oplus c) \sqsubseteq (b \oplus c)$
2) $a \sqsubseteq b \rightarrow (a \otimes c) \sqsubseteq (b \otimes c)$
3) $a \sqsubseteq b \rightarrow (c \otimes a) \sqsubseteq (c \otimes b)$

Lemma 2 ([40])**.** *Let $\mathcal{S}$ be an additively idempotent semiring $(S, \oplus, \otimes, e_\oplus, e_\otimes)$ equipped with the natural order $\sqsubseteq$ on S. Then S is both negative and monotonic.*

Lemma 3. *Let $\mathcal{S} = (S, \oplus, \otimes, e_\oplus, e_\otimes)$ be an additively idempotent, negative and monotonic semiring. Then, for all $a \in S$, $e_\otimes \sqsubseteq a \sqsubseteq e_\oplus$.*

Proof. *Consider an arbitrary $a \in S$. By Definition 3 and assumption that S is negative, we have that $e_\otimes \sqsubseteq e_\oplus$. By Definition 3 and assumption that S is monotonic, we have that $e_\otimes \otimes a \sqsubseteq e_\oplus \otimes a$ and $e_\otimes \oplus a \sqsubseteq e_\oplus \oplus a$. By Definition 1, $e_\otimes \sqsubseteq a$ and $a \sqsubseteq e_\oplus$, which concludes the proof.*

B. Metric Spaces and Distances

A metric space is a set $\mathcal{M}$ possessing a distance among its elements. The distance $d(m_1, m_2)$ between two elements $m_1, m_2 \in \mathcal{M}$ is a positive real value in $\mathbb{R}_+$.

Definition 4 (Metric space and distance)**.** *Given a set S, let $d : \mathcal{M} \times \mathcal{M} \rightarrow \mathbb{R}_+$ be a* distance*. Then $\mathcal{M}$ is a* metric space *with the* distance measure *d, if:*

1) *$d(m_1, m_2) \geq 0$ for all m_1, m_2 in $\mathcal{M}$;*
2) *$d(m_1, m_2) = 0$ if and only if $m_1 = m_2$;*
3) *$d(m_1, m_2) = d(m_2, m_1)$ for all m_1, m_2 in $\mathcal{M}$; and*
4) *$d(m_1, m_2) \leq d(m_1, m) + d(m, m_2)$ for all m, m_1, m_2 in $\mathcal{M}$.*

Since we reason about real-valued behaviors and the distances between them, we are interested in semirings defined over (subsets of) reals, see Example 1. Given $m \in \mathcal{M}$ and $M \subseteq \mathcal{M}$, we can lift the above definition to reason about the distance[1] between an element m of $\mathcal{M}$ and the subset M of $\mathcal{M}$ to define a Hausdorff-like measure. In this extension, we extend the set of reals with the infinity ∞ element. We use the $\oplus$-addition to combine individual distances between m and the elements in M and fix $e_\otimes$ to 0. We also need a special value when we compare m to an empty set and define $d(m, \emptyset) = e_\oplus$.

$$d(m, M) = \begin{cases} e_\oplus & \text{if } M \text{ is empty} \\ \oplus_{m' \in M} d(m, m') & \text{otherwise.} \end{cases}$$

We define the *robustness degree* $\rho(m, M)$ of m w.r.t. the set M as:

$$\rho(m, M) = \begin{cases} d(m, \mathcal{M} \backslash M) & \text{if } m \in M \\ -d(m, M) & \text{otherwise.} \end{cases}$$

C. Traces and Specification Languages

Let X denote a set of variables defined over a *domain* D. We denote by $v : X \rightarrow D$ the *valuation* function that maps a variable in X to a value in D. We denote by $\tau = v_1, \ldots, v_n$ a *trace* over X and by $\mathcal{T}(X)$ the set of all traces over X.

A specification φ over a set of variables X, regardless of the formalism used, defines a *language* $L(\varphi) \subseteq \mathcal{T}(X)$ that partitions the set of all traces over X.

Definition 5 (Trace-specification distance)**.** *Let v and v' be two valuations over $D^{|X|}$, τ and τ' two traces over X of size m and n and φ a specification over X. We then have:*

$$\begin{array}{lcl} d(v, v') & = & \otimes_{x \in X} d(v(x), v'(x)) \\ d(\tau, \tau') & = & \begin{cases} e_\oplus & \textit{,if } m \neq n \\ \otimes_{1 \leq i \leq m} d(v_i, v'_i) & \textit{,otherwise} \end{cases} \\ d(\tau, \varphi) & = & \oplus_{\tau' \models \varphi} d(\tau, \tau') \end{array}$$

In this paper, we consider specification languages defined over discrete time and real-valued variables that are *regular*. Examples of specification languages that fall into this category are Signal Temporal Logic (STL) and Signal Regular Expressions (SRE), both interpreted over discrete time.[2]. Due to the page limit, we recall only the syntax of STL and SRE at this point.

We consider STL with both *past* and *future* operators interpreted over digital signals of final length. We assume that D is a metric space equipped with a distance d. The syntax of a STL formula φ over X is defined by the grammar:

$$\varphi := x \sim u \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \mathcal{U}_I \varphi_2 \mid \varphi_1 \mathcal{S}_I \varphi_2$$

where $x \in X$, $\sim \in \{<, \leq\}$, $u \in \mathbb{D}$, I is of the form $[a, b]$ or $[a, \infty)$ such that $a, b \in \mathbb{N}$ and $0 \leq a \leq b$. The other standard (Boolean and temporal) operators are derived from the basic ones. For instance, *eventually* $\Diamond_I \varphi$ is defined as $\top \mathcal{U}_I \varphi$, while *always* $\Box_I \varphi$ corresponds to $\neg \Diamond_I \neg \varphi$.

The syntax of a SRE formula φ over X is defined by the grammar:

$$\begin{array}{lcl} b & := & x \sim u \mid \neg b \mid b_1 \vee b_2 \\ \varphi & := & \epsilon \mid b \mid \varphi_1 \cdot \varphi_2 \mid \varphi_1 \cup \varphi_2 \mid \varphi_1 \cap \varphi_2 \mid \varphi^* \mid \langle \varphi \rangle_I \end{array}$$

where $x \in X$, $\sim \in \{<, \leq\}$, $u \in \mathbb{D}$, I is of the form $[a, b]$ or $[a, \infty)$ such that $a, b \in \mathbb{N}$ and $0 \leq a \leq b$. Although we

[1] Since $d(m, M)$ is comparing an element to a set, strictly speaking it is not a distance.

[2] STL and TRE interpreted over discrete time are not more expressive than Linear Temporal Logic (LTL) and regular expressions (RE) augmented with predicates over real variables.

This article was presented in the International Conference on Embedded Software 2018
and appears as part of the ESWEEK-TCAD special issue.

interpret SRE over discrete time, we interpret its operators following the style of continuous time TRE. As a consequence, a signal segment that matches a predicate such as $x \leq 5$ means that it matches it for a strictly positive duration. The time duration operator $\langle\varphi\rangle_I$ is matched by a segment if it has a duration in I. STL and SRE semantics are given in [38] and [6].

Example 2. *Consider the requirement "There must be a point in time within the trace where (1) x is smaller or equal than 3, and (2) both x is smaller or equal than 5 and y is greater or equal than 6 for the duration of at least two time steps". We formalize the above requirement as the STL specification $\varphi_1 \equiv \Diamond(x \leq 3 \wedge \Box_{[0,1]}(x \leq 5 \wedge y \geq 6))$ and the SRE specification $\varphi_2 \equiv \top \cdot ((x \leq 3) \cdot \top) \cap \langle x \leq 5 \wedge y \geq 6 \rangle_{[1,1]}) \cdot \top$.*

D. Symbolic and Symbolic Weighted Automata

In this section, we first recall the definition of symbolic automata as the automata with transitions labeled by predicates over rich alphabet theories (a fragment of linear arithmetic over reals in this paper) and then define their weighted extension.

We start by defining a predicate. Let $D = \mathbb{R}$ be the domain of reals and X the set of variables defined over D. We now define predicates over variables in X.

Definition 6 (Predicate)**.** *We define the syntax of a* predicate *ψ over X with the following grammar: $\psi := \bot \mid \top \mid x \preceq k \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid \psi_1 \wedge \psi_2$, where $\preceq \in \{<, \leq\}$, $x \in X$ and $k \in D$.*

We denote by $\Psi(X)$ the set of all predicates over X. We say that a valuation v *models* the predicate ψ, denoted by $v \models \psi$, iff ψ evaluates to true under v.

We note that $x \preceq k$ plays the role of basic propositions. In our framework, the predicates come from specifications, hence we allow predicates in arbitrary form. In order to define the subsequent RV algorithms, we need to transform arbitrary predicates into *(minimal) disjunctive normal form (DNF)*.

Definition 7 (Predicate in Disjunctive Normal Form (DNF))**.** *A predicate in disjunctive normal form is generated by the grammar:*

$$\begin{array}{ll} \psi & := \psi^c \mid \psi^c \vee \psi \\ \psi^c & := \psi^l \mid \psi^l \wedge \psi^c \\ \psi^l & := \top \mid \bot \mid (x \preceq k) \mid \neg(x \preceq k) \end{array}$$

where $\preceq \in \{<, \leq\}$, $x \in X$ and $k \in D$. The predicate has the structure:

$$\psi = \bigvee_{i=1}^{h} \psi_i^c, \quad \psi_i^c = \bigwedge_{h=1}^{m(i)} \psi_{i,h}^l$$

where h is the number of clauses, each clause i is a conjunction of $m(i)$ literals and each literal can be either a basic proposition, its negation, true or false.

Definition 8 (Predicate in $\wedge$-minimal DNF)**.** *A predicate ψ is expressed in a $\wedge$-minimal DNF if it satisfies the following properties:*

$$\bigwedge_{i=1}^{h} \bigwedge_{s=1,r=1,s\neq r}^{m(i)} (\psi_{i,s}^l \rightarrow \psi_{i,r}^l) = \bot$$

Example 3. *The predicate $\psi_1 \equiv (x \leq 3 \wedge x \leq 5 \wedge y \leq 5) \vee (z > 0)$ is in DNF, while the predicate $\psi_2 \equiv (x \leq 3 \wedge y \leq 5) \vee (z > 0)$ is in $\wedge$-minimal DNF.*

We lift the definition of a distance between two valuations to the distance between a valuation and a predicate by $\oplus$-summing the distances between the valuation and the set of valuations defined by a predicate.

Definition 9 (Valuation-predicate distance)**.** *Given a valuation $v \in D^{|X|}$ and a predicate $\psi \in \Psi(X)$, we have that:*

$$d(v, \psi) = \oplus_{v' \models \psi} \otimes_{x \in X} d(v(x), v'(x)).$$

This completes our definitions for computing the distance between an observation and a specification at a single point in time. We now concentrate on the dynamic (temporal) aspect of the specification. We first define *symbolic* and *symbolic weighted automata*.

Definition 10 (Symbolic and Symbolic Weighted Automata)**.** *A* symbolic automaton *(SA) $\mathcal{A}$ is the tuple $\mathcal{W} = (X, Q, I, F, \Delta)$, where X is a finite set of variables defined over a domain D, Q is a finite set of* locations*, $I \subseteq Q$ is the set of* initial *states, $F \subseteq Q$ is the set of* final *states and $\Delta \subseteq Q \times \Psi(X) \times Q$ is the* transition relation*. A* symbolic weighted automaton *$\mathcal{W}$ is the pair $(\mathcal{A}, \lambda)$, where $\mathcal{A}$ is a symbolic automaton and $\lambda : \Delta \times D^{|X|} \rightarrow D$ is the* weight *function.*

In the following, we assume that the weights are elements of the semiring $(S, \oplus, \otimes, e_\oplus, e_\otimes)$. We use $\otimes$ to compute the weight of a path by $\otimes$-multiplying the weights of the transitions taken along that path. We use $\oplus$ to $\oplus$-sum the path weights induced by an input trace. We now formalize the above notions.

A *path* π in $\mathcal{A}$ is a finite alternating sequence of locations and transitions $\pi = q_0, \delta_1, q_1, \ldots, q_{n-1}, \delta_n, q_n$ such that $q_0 \in I$ and for all $1 \leq i \leq n$, $(q_{i-1}, \delta_i, q_i) \in \Delta$. We say that the path π is *accepting* if $q_n \in F$. We say that a trace $\tau = v_1, v_2, \ldots, v_n$ *induces* a path $\pi = q_0, \delta_1, q_1, \ldots, q_{n-1}, \delta_n, q_n$ in $\mathcal{A}$ if for all $1 \leq i \leq n$, $v_i \models \psi_i$, where $\delta_i = (q_{i-1}, \psi_i, q_i)$. We denote by $\Pi(\tau) = \{\pi \mid \pi \in F$ and τ induces π in $\mathcal{A}\}$ the set of all accepting paths in $\mathcal{A}$ induced by trace τ.

Definition 11 (Path and trace value)**.** *The* path value *$\mu(\tau, \pi)$ of a path π induced in $\mathcal{A}$ by a trace τ is defined as: $\mu(\tau, \pi) = \otimes_{1 \leq i \leq n} \lambda(\delta_i, v_i)$. The* trace value *$\alpha(\tau, \mathcal{W})$ of a trace τ in $\mathcal{W}$ is defined as: $\alpha(\tau, \mathcal{W}) = \oplus_{\pi \in \Pi(\tau)} \mu(\tau, \pi)$.*

IV. Algebraic Monitors for Correctness and Robustness

In this section, we develop ARV, a novel procedure for abstract computation of the *robustness degree* of a discrete

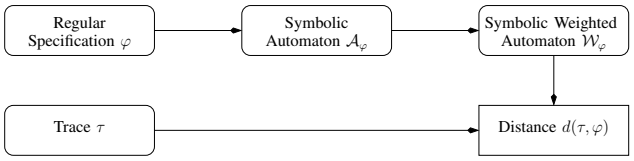

Fig. 1. Computation of $d(\tau, \varphi)$.

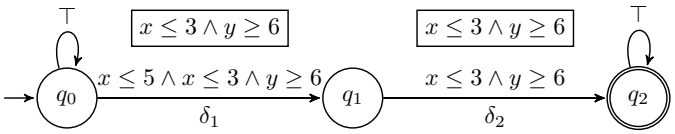

Fig. 2. SWA $\mathcal{W}$ that accepts the language of φ_1 and φ_2.

signal with respect to a specification φ. The proposed methods consists of several steps, illustrated in Figure 1. We first translate the specification φ into a symbolic automaton $\mathcal{A}_\varphi$ that accepts the same language as the specification. The automaton $\mathcal{A}_\varphi$ treats timing constraints from the formula in an enumerative fashion, but keeps symbolic guards on data variables. We then decorate $\mathcal{A}_\varphi$ with weights on transitions as explained in section IV-A, thus obtaining the *symbolic weighted automaton* $\mathcal{W}_\varphi$. We propose an abstract algorithm for computing the distance between a trace τ and a specification φ by reducing it to the problem of finding the shortest path in $\mathcal{W}_\varphi$ induced by τ. Computing the robustness degree between the trace τ and the specification φ follows from combining the computed distance from the specification φ and its negation $\neg\varphi$.

A. From Specification to SWA

We assume in this paper an effective translation T of a regular specification language φ to symbolic automata $\mathcal{A}_\varphi$ that accepts the language of φ, i.e. $L(\varphi) = L(\mathcal{A}_\varphi)$. For STL and SRE defined over discrete time, such translation is a moderate adaptation of standard methods that translate LTL or RE to automata, including on-the-fly tableau construction [33] and temporal testers [44]. For the sake of saving space, we do not elaborate further on the well-known translations from specifications to automata. During the translation, we decorate the transitions in the symbolic automaton with weight functions that measure the distance between observed valuations and the predicate ψ associated to the transition, i.e. we instantiate the weight function λ to $\lambda(\delta, v) = d(v, \psi)$, for all $\delta = (q, \psi, q') \in \Delta$.

Example 4. *We illustrate this step on specifications φ_1 and φ_2 from Example 2:* $\varphi_1 \equiv \Diamond(x \leq 5 \wedge \Box_{[0,1]}(x \leq 3 \wedge y > 6))$,
$\varphi_2 \equiv \top \cdot ((x \leq 5 \cdot \top) \cap \langle x \leq 3 \wedge y \geq 6 \rangle_{[1,1]}) \cdot \top$
In Figure 2 we depict the SWA for languages of φ_1 and φ_2. The $\wedge$-minimal predicates are shown in the boxes above the transitions.

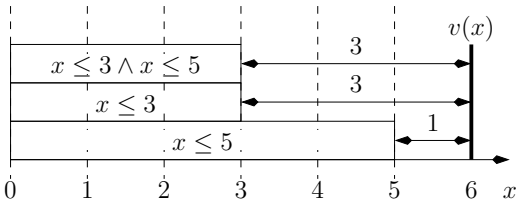

Fig. 3. Example of a distance between v and ψ.

B. Valuation-Predicate Distance Computation

We propose an effective procedure for computing the distance between a valuation v and a predicate ψ. The procedure is shown in Algorithm 1 and works as follows. The input to the procedure is a valuation v and a predicate ψ in DNF. The computation of the distance between v and ψ is done inductively in the structure of ψ. In the base case when ψ is an atomic predicate, the distance between v and ψ is $e_\otimes$ if v satisfies ψ, and it is equal to $d(v(x), k)$ otherwise. We compute logic operations $\vee$ and $\wedge$ by interpreting them as $\oplus$-addition and $\otimes$-multiplication.

The procedure presented in Algorithm 1 indeed computes the distance from Def. 9 for (1) idempotent semirings and (2) additively idempotent semirings with the predicate given in $\wedge$-minimal DNF. The transformation of arbitrary predicates to DNF is standard, while we present the $\wedge$-minimization of a predicate in DNF in Algorithm 2. In the following example, we give intuition why predicates must be in $\wedge$-minimal DNF if the semiring is only $\oplus$-idempotent.

Example 5. *Consider the term $x \leq 3 \wedge x \leq 5$ from the transition δ_1 in Figure 2 that defines the set of valuations $\{v(x) \mid v(x) \leq 3\}$, its semantically equivalent $\wedge$-minimal representation $x \leq 3$ and the valuation $v(x) = 6$. It is clear that $d(v, x \leq 3 \wedge x \leq 5) = d(v, x \leq 3) = 3$. Let us consider the tropical semiring and the computation of vpd. We illustrate the need for the $\wedge$-minimal predicate in Figure 3. We have that $\mathsf{vpd}(v, x \leq 3) = 3$, but $\mathsf{vpd}(v, x \leq 3 \wedge x \leq 5) = 1 \otimes 3 = 1 + 3 = 4$. Due to the non-minimality of the term and the additive nature of $+$, we incorrectly accumulate the distance from $v(x)$ to the atomic predicate $x \leq 5$.*

Algorithm 1 $\mathsf{vpd}(v, \psi)$

Require: Predicate $\psi \equiv \bigvee_i \bigwedge_j p_{ij}$
1: **if** (ψ is UNSAT) **then return** $e_\oplus$
2: **end if**
3: **if** ($\psi = x \preceq k$) or ($\psi = \neg(x \preceq k)$) **then**
4: **if** ($v(x) \models \psi$) **then return** $e_\otimes$
5: **else return** $d(v(x), k)$
6: **end if**
7: **else if** ($\psi = \psi_1 \vee \psi_2$) **then**
8: **return** $\mathsf{vpd}(v, \psi_1) \oplus \mathsf{vpd}(v, \psi_2)$
9: **else if** ($\psi = \psi_1 \wedge \psi_2$) **then**
10: **return** $\mathsf{vpd}(v, \psi_1) \otimes \mathsf{vpd}(v, \psi_2)$
11: **end if**

This article was presented in the International Conference on Embedded Software 2018
and appears as part of the ESWEEK-TCAD special issue.

Algorithm 2 $\wedge$-min(v, ψ)

Require: Predicate $\psi \equiv \bigvee_i \bigwedge_j p_{ij}$
Let $P_i = \{p_{i1}, \ldots, p_{ij}\}$
Let $\mathcal{P} = \{P_1, \ldots, P_i\}$
for all $P \in \mathcal{P}$ **do**
  for all $p_1, p_2 \in P$ s.t. $p_1 \neq p_2$ **do**
    if $p_1 \rightarrow p_2$ **then**
      $P \leftarrow P \backslash \{p_2\}$
    end if
  end for
end for
return $\bigvee_{P \in \mathcal{P}} \bigwedge_{p \in P} p$

The following theorem states that the procedure $\mathsf{vpd}(v, \psi)$ described in Algorithm 1 correctly computes the the distance $d(v, \psi)$ between an observation v and a predicate ψ for idempotent (Boolean, MinMax) semirings. In the case of semirings that are only additively idempotent, such as the tropical semiring, the predicate must be in addition translated with the application of Algorithm 2 to the $\wedge$-minimal form. We use this theorem to show that we properly generate weights on the transitions of the symbolic automaton.

Theorem 1. *Given a predicate ψ in DNF, a valuation v and the distance $d(v, \psi)$ defined over a bounded semiring S, we have that $\mathsf{vpd}(v, \psi) = d(v, \psi)$ if: (1) S is idempotent; or (2) ψ is in $\wedge$-minimal DNF.*

Proof. *(Sketch) The proof is inductive in the structure of the predicate. We first prove the theorem for an arbitrary term ψ^c and then prove it for general DNF predicate ψ. The proof of the base case follows directly from the definitions of the distance and Algorithm 1. The proof for a term ψ^c is more intricate when ψ^c contains two literals of the form $x \leq c_1$ and $x \leq c_2$. In case that $\otimes$ is idempotent, such a term does not pose the problem. However, when $\otimes$ is not idempotent, we need to assume that ψ^c is $\wedge$-minimal which ensures that ψ^c does not contain both $x \leq c_1$ and $x \leq c_2$. The proof for the disjunction of terms is straightforward from the definitions, due to the assumption that $\oplus$ is idempotent.*

C. Trace Value Computation

In this section, we present a dynamic programming procedure (see Algorithm 3) for computing the value of a trace $\tau = v_1, \ldots, v_n$ in a symbolic weighted automaton $\mathcal{W}$. We assume that the weights are defined over a semiring S. The algorithm first assigns to every state q the initial cost, depending whether q is initial or not. At every step $i \in [1, n]$ and for every state q of $\mathcal{W}$, the procedure computes the cost of reaching q with the i-prefix of τ. The procedure uses the $\otimes$-multiplication to aggregate the valuation-predicate distances collected along every path π induced by τ and thus compute the path weight and the $\oplus$-addition to combine the weights of all the accepting paths induced by τ.

We now state that Algorithm 3 correctly computes the value of τ in $\mathcal{W}$, which corresponds to the distance $d(\tau, \varphi)$. Then we build the monitor that measures the robustness degree between the trace τ and a specification φ by computing the value of τ in $\mathcal{W}_\varphi$ and $\mathcal{W}_{\neg\varphi}$. This procedure is summarized in Algorithm 4 that trivially implements the robustness measure ρ. We show that our abstract computation of ρ is sound and complete.

Algorithm 3 $\mathsf{val}(\tau, \mathcal{W})$

for all $q \in Q$ **do**
  $c(q, 0) \leftarrow (q \in I)\ ?\ e_\otimes\ :\ e_\oplus$
end for
for $i = 1$ *to* n **do**
  $c(q, i) \leftarrow \oplus_{(s,\psi,q) \in \Delta} (c(s, i-1) \otimes \mathsf{vpd}(v_i, \psi))$
end for
return $\oplus_{q \in F}\ c(q, n)$

Algorithm 4 $\mathsf{rob}(\tau, \varphi)$

$\mathcal{W}_\varphi \leftarrow \mathsf{T}(\varphi)$
$\mathcal{W}_{\neg\varphi} \leftarrow \mathsf{T}(\neg\varphi)$
$v_1 \leftarrow \mathsf{val}(\tau, \mathcal{W}_\varphi)$
$v_2 \leftarrow \mathsf{val}(\tau, \mathcal{W}_{\neg\varphi})$
if $v_1 = e_\otimes$ **then**
  return v_2
else return $-v_1$
end if

The following theorem states the main result of the paper - Algorithm 4 correctly computes the distance between a specification φ and a trace τ. We note that the theorem works for any $\mathcal{W}$ that has the same language as φ. In particular, the theorem implies that val is invariant to different syntactic, semantically equivalent, representations of the specification and of the associated automaton.

Theorem 2. *Given a specification φ, the automaton $\mathcal{A}_\varphi$ such that $L(\varphi) = L(\mathcal{A}_\varphi)$, its associated SWA $\mathcal{W}_\varphi$ defined over a semiring S and a trace τ, we have that $\mathsf{val}(\tau, \mathcal{W}_\varphi) = \alpha(\tau, \mathcal{W}_\varphi) = d(\tau, \varphi)$.*

Proof. *The proof follows from the monotonicity of the natural order in additively idempotent semirings. This property allow us to merge the values of all paths π induced by a prefix of size n of the trace τ and ending in a location q into a single representative value that we represent as the cost $c(q, n)$ of the location at time n and that is used to compute the value of all the extensions. (we recall that for all $a, b, c \in S$, if $a \sqsubseteq b$, then $a \otimes c \sqsubseteq b \otimes c$). Following the definition of the trace value, the cost $c(q, n+1)$ of q at time $n+1$ is then the $\oplus$-summation of the individual effects of taking all possible transitions (s, ψ, q) from s to q with the new valuation v_{n+1} (the cost $c(s, n)$ $\otimes$-multiplied by the predicate-value distance $\mathsf{vpd}(v_{n+1}, \psi)$).*

Theorem 3 establishes the relation between the (quantitative) robustness degree measurement and the (qualitative) satisfaction relation. The theorem states that the sign of $\rho(\tau, \varphi)'$ determines the satisfaction status of the formula. It also states that if τ satisfies φ, then any other trace τ' whose distance from τ is smaller than the robustness of τ from φ also satisfies φ.

Theorem 3. *Given traces τ and τ', a specification φ and distances $d(\tau, \tau')$, $d(\tau, \neg\varphi)$ defined over a bounded semiring S,*

$$\begin{aligned} \rho(\tau,\varphi) > 0 &\rightarrow \tau \models \varphi \\ \rho(\tau,\varphi) < 0 &\rightarrow \tau \not\models \varphi \\ \tau \models \varphi \text{ and } d(\tau,\tau') \sqsubset \rho(\tau,\varphi) &\rightarrow \tau' \models \varphi. \end{aligned}$$

Proof. *The proof for the first two implications is trivial from the definitions of $\rho(\tau,\varphi)$ and $d(\tau,\varphi)$. We prove the third implication by contradiction. Assume that $\tau \models \varphi$, $d(\tau,\tau') \sqsubset \rho(\tau,\varphi)$ and $\tau' \not\models \varphi$. Since by assumption $\tau \models \varphi$, we have that $\rho(\tau,\varphi) = d(\tau',\neg\varphi)$. Then, by definition of the distance, we have that $d(\tau',\neg\varphi) = e_\otimes$. By the additive idempotence of S and the definition of $d(\tau',\neg\varphi)$, there exists $\tau'' \models \neg\varphi$ such that $d(\tau',\tau'') = e_\otimes$, hence $\tau' = \tau''$. However, in that case we reach a contradiction with $d(\tau,\tau') = d(\tau,\neg\varphi)$.*

We first observe that if $\rho(\tau,\varphi) = 0$, then we do not know (only from that number) whether τ satisfies φ. We illustrate this observation with the formula $x > 0$ and the trace $\tau = 0$ of size one. It is clear that $\tau \not\models \varphi$ but the actual distance between the element in $\{v \mid v > 0\}$ that is the closest to 0 and 0 is infinitesimally close to 0. In order to have both directions of the implications in the soundness proof and to guarantee that the robustness degree is never equal to 0, we would need to introduce non-standard reals. Note that even with the current setting, we can easily say whether $\tau \models \varphi$, even when $\rho(\tau,\varphi) = 0$. Second, we do not need to explicitly compute the complement automaton $\mathcal{W}_{\neg\varphi}$ if $\mathcal{W}_\varphi$ is deterministic. In that case, it is sufficient to apply a slight modification of Algorithm 3 on $\mathcal{W}_\varphi$ only. The modification consists in reporting either the minimum value of an accepting or the minimum value of a non-accepting location, depending on whether the trace satisfies φ.

D. Instantiating Monitors

In Sections IV-B and IV-C, we presented an abstract monitoring procedure that measures a robustness degree of a trace τ with respect to a specification φ. We give concrete semantics to these monitors by instantiating the semiring and the distance function. Here we consider three instantiations of the procedure:

1) Boolean semiring with $d(a,b) = 1$ if $a \neq b$ and 0 otherwise
2) Minmax semiring with $d(a,b) = |a-b|$
3) Tropical semiring with $d(a,b) = |a-b|$

The instantiation 1 gives the monitors classical qualitative semantics, where the distance of τ from φ is 0 if τ is in the language of φ and 1 otherwise. The instantiation 2 gives the computation of the robustness degree based on the infinity norm, similarly as defined (see Section V-C) in [30]. The instantiation 3 gives the computation of the robustness degree based on the Hamming distance lifted to the sets.

Example 6. *We illustrate our monitoring procedure instantiated to different semantics in Table II. We choose the trace $\tau = (4,2)\cdot(5,3)\cdot(2,5)\cdot(3,5)$ that violates the specifications φ_1 and φ_2 from Example 2. We instantiate Algorithm 3 to the specific semirings and apply each instantiation to the τ and $\mathcal{W}$ from Figure 2. We mark in bold the accepting state and the trace value.*

semiring	state	init	$(4,2)$	$(5,3)$	$(2,5)$	$(3,5)$
Boolean	q_0	0	0	0	0	0
	q_1	1	1	1	1	1
	$\boldsymbol{q_2}$	1	1	1	1	**1**
MinMax	q_0	0	0	0	0	0
	q_1	∞	4	3	1	1
	$\boldsymbol{q_2}$	∞	∞	4	3	**1**
Tropical	q_0	0	0	0	0	0
	q_1	∞	5	5	2	1
	$\boldsymbol{q_2}$	∞	∞	10	7	**3**

TABLE II
VAL$(\tau,\mathcal{W})$ COMPUTED ON SWA FROM FIGURE 2 WITH DIFFERENT SEMIRING INSTANTIATIONS.

We note that by Theorem 2, our computation of robustness is *precise* with respect to the semantics of the specification regardless of the instantiated semiring. This is in contrast to the syntactic approaches to robustness [26], [30] that under-approximate the robustness value. In case we instantiate MinMax semiring in ARV, we do not demonstrate imprecision shown in Examples 17,18 from [30]. The comparison results from section V-A confirm this observation.

E. Implementing Monitors

The ARV framework that we develop in this paper admits an effective and straightforward implementation of both *offline* and *real-time* monitors. Algorithms 3 and 4 describe the offline monitoring procedure in which we assume the availability of the entire input trace τ. We use this assumption solely to simplify the presentation. This monitoring procedure is directly extended to its online variant by processing new inputs as they arrive and by updating the costs of the automaton locations in an on-the-fly fashion.

The size of the automaton $\mathcal{W}$ obtained from a specification φ is in $\mathcal{O}(2^{b|\varphi|})$, where b is the largest constant appearing in timed modalities $[a,b]$ of φ [4]. We note that this is the worst case upper bound. We conjecture that the discrete time adaptation of more recent translations from real-time temporal logics to timed automata [28], [42] yields automata with size that is exponential in the ratio between the lower bound a and the upper bound b. This significantly reduced the size of automata for many classes of specifications. For example, the automaton associated to $\Diamond_{[0,b]}p$ has the size in $\mathcal{O}(b)$. On the other hand, translating a punctual formula such as $\Diamond_{[b,b]}p$ without any assumption on the variability of p results indeed into an automaton that reaches the worst case size in $\mathcal{O}(2^b)$.

By definition of Algorithm 3, it is clear that the number of operations required to compute the trace value at every time step is linear in (1) the size of the automaton, (2) the maximum number of incoming transitions that any location in the automaton can have, and (3) the maximum size of the predicates appearing in the transitions of the automaton. More specifically, the number of operations at every time step remains constant with the length of the observed prefix. This results in a predictable monitoring procedure with real-time guarantees that is well-suited for an online implementation.

Algorithm 3 needs to store at every point in time $2n$ real values, where n denotes the number of locations in the automaton - the current and the previous cost for each location. In our theoretical setting, both inputs and costs range over

This article was presented in the International Conference on Embedded Software 2018
and appears as part of the ESWEEK-TCAD special issue.

reals. In any practical implementation, the observation of the monitored CPS during its operation is limited by the resolution of the measurement device, which typically quantizes the observed values into a finite set of integer values $[0, k]$. We assume that we use the binary representation of integers. It follows that every value stored in a monitor instantiated with the Boolean or the MinMax semiring is bounded by k and can be represented by $\log k$ bits. As a consequence, Algorithm 3 instantiated with qualitative or infinity-based norm quantitative semantics has memory requirements that are constant in the size of the input trace, another important property for the implementation of real-time monitors.

In the case of monitors instantiated with the tropical semiring, the situation is slightly more involved, since costs are accumulated over time. It follows that for such monitors, the memory requirements grows logaritmically with the length of the input trace. A real-time implementation of such monitors can address this logarithmic growth by using a large enough number of bits for integers that will guarantee correctness for sufficiently long input traces.

V. Examples

We implemented our approach (ARV) in a prototype tool in Java. We used an adaptation of the temporal testers approach [44] to translate STL specifications into symbolic automata. In order to determine satisfiability of SWA transition constraints, we used the Z3 solver [20]. We provide a preliminary evaluation of our framework as a proof of concept on two examples from the automotive domain: The Autonomous Vehicle (AV) Control Stack model [47] and the Automatic Transmission System model [34]. We also demonstrate on a concrete example the precision of ARV compared to relevant syntax-based tools developed by the RV community.

A. Autonomous Vehicle Control Stack

The first benchmark is a model of an autonomous vehicle control stack, which is used to solve the trajectory planning problem. The stack consists of three layers, starting from the top: Behavioral Planner (BP), Trajectory Planner (TP) and Trajectory Tracker (TT). The BP provides the coarse-grain trajectory way-points for the Autonomous Vehicle (AV), and supplies them to the underlying TP which calculates fine grain trajectory points, using cubic spline trajectory generation. The lowest layer is the TT which actuates the AV based on the trajectory points in order to steer it towards the requested path.

The benchmark supports three distinct layouts: a room with obstacles, a curved road and a roundabout. The obstacles and undesired areas are determined by assigning the specific cost. The autonomous vehicles can simulate four scenarios: parallel driving, paths crossing without collision and collision avoidance by either the first or the second vehicle. We ran the scenario of AVs performing parallel driving on roundabout layout and we obtained the traces for the speed v_x and acceleration a_x. We specified two requirements, which model normal operation (w.r.t. acceleration) and traffic rules (speed limit).

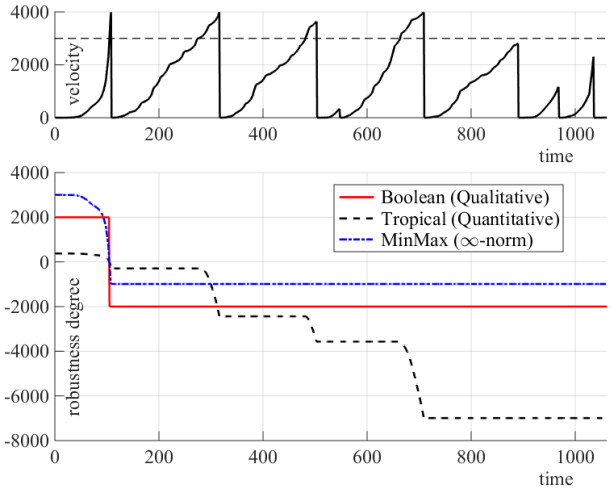

Fig. 4. Robustness degree $\rho(\tau_{[0,t]}, \varphi_1)$, where $\varphi_1 = \square(v_{ego} \leq v_{limit})$, based on three different semiring instantiations.

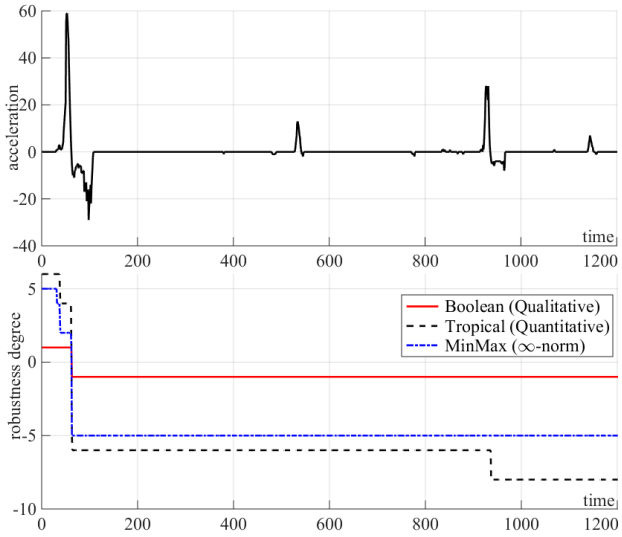

Fig. 5. Robustness $\rho(\tau_{[0,t]}, \varphi_2)$, where $\varphi_2 = \square((a_x \geq \theta) \rightarrow \square_{(0,\epsilon]} \neg(a_x \leq 0))$, based on different semiring instantiations.

In Fig. 4 we demonstrate various robustness degrees for the following requirement: "The ego vehicle travels at a velocity less than or equal to the speed limit". We formalize the requirement in STL: $\varphi_1 = \square(v_{ego} \leq v_{limit})$. In Fig. 5 we monitor the requirement: "If the autonomous vehicle started to accelerate, then it will not start decelerating in the very near future". We formalize the requirement in STL: $\varphi_2 = \square((a_x \geq \theta) \rightarrow \square_{(0,\epsilon]} \neg(a_x \leq 0))$. We select $\theta = 5$ and $\epsilon = 3$. For the clarity of Fig. 4, values are scaled.

B. Automatic Transmission System

We first consider the Automatic Transmission deterministic model [34] as our system-under-test (SUT). It is a model of a transmission controller that exhibits both continuous and discrete behavior. The system has two inputs – the throttle u_t and the break u_b. The break allows the user to model variable load on the engine. The system has two continuous-

This article was presented in the International Conference on Embedded Software 2018
and appears as part of the ESWEEK-TCAD special issue.

time variables – speed of the engine ω (RPM), speed of the vehicle v (mph) and active gear g_i.

The system is initialized with zero vehicle and engine speed. It follows that the output trajectories depend only on the input signals u_t and u_b, which can take any value between 0 and 100 at any point in time. The Simulink model contains 69 blocks including 2 integrators, 3 look-up tables, 2 two-dimensional look-up tables and a Stateflow chart with 2 concurrently executing finite state machines with 4 and 3 states, respectively. The benchmark [34] defines 8 STL formalized requirements that the system must satisfy.

We select the following requirement: "The engine and the vehicle speed never reach ω_{max} and v_{max}". We use STL to formalize this requirement as follows: $\varphi_3 = \Box((\omega \leq \omega_{max}) \wedge (v \leq v_{max}))$.

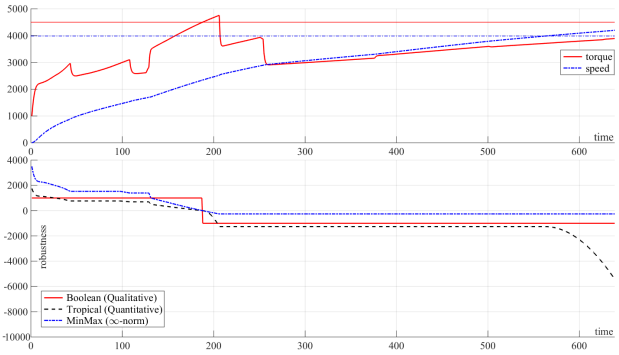

Fig. 6. Robustness degree $\rho(\tau_{[0,t]}, \varphi_3)$, where $\varphi_3 = \Box((\omega \leq \omega_{max}) \wedge (v \leq v_{max}))$, based on different semiring instantiations.

We translate φ_3 into a SWA and instantiate it with three semirings – Boolean, MinMax and Tropical, thus obtaining monitors for qualitative, ∞-norm and Hamming distance based quantitative semantics. We note that the qualitative semantics yields the binary verdicts. In contrast, the quantitative semantics based on the ∞-norm is obtained by instantiating MinMax idempotent semiring. Therefore the robustness degree based of this semantics relies on maximum pointwise distance, without accumulating it over time. For this reason we find it useful for performing worst-case analysis. Finally, the quantitative semantics based on Tropical semiring accumulates the pointwise distances over the entire trace due to non-idempotent $\otimes$ (addition). Thus, the robustness degree in each time instance depends on the robustness of the entire trace prefix, ensuring that no information on robustness is lost over time. We can finally observe that all the quantitative semantics are consistent with qualitative semantics, as stated in Theorem 3. This is observable in Fig 6, before time reaches 200.

C. Comparison with S-TaLiRo and Breach

We now compare our approach to the widely used S-TaLiRo [5] and Breach [24] tools. Both S-TaLiRo and Breach implement robustness monitoring algorithms, measured using infinity norm. The algorithms implemented in these tools work inductively on the structure of the formula without passing via automata. We note that in contrast to our setting, these two tools work with continuous time. In order to enable the comparison, we instantiate our monitors to the MinMax semiring and we simulate the AV Control Stack model with fixed sampling, thus ensuring that the discrete vs. continuous discrepancy does not affect the results.

	STLR	Breach	ARV
$\psi_1 = a \geq -30 \wedge a \leq 30$	30	30	30
$\psi_2 = (a \geq -30 \wedge a < 0) \vee (a \geq 0 \wedge a \leq 30)$	0	10^{-13}	30
$\psi_3 = \Diamond(a \geq -10)$	69	69	69
$\psi_4 = \Diamond((a \geq -10 \wedge a \leq 60) \vee (a \geq 55))$	35	35	69
$\psi_5 = \Box(a \geq 5 \wedge a < 5)$	-64	-59	$-\infty$
$\psi_6 = \neg(\Diamond\psi_1 \vee \Diamond(a < -30 \vee a > 30))$	-30	-30	$-\infty$

TABLE III
PRECISION COMPARISON: SYNTACTIC VS. SEMANTIC TOOLS.

We first observe that S-TaLiRo and Breach use a different notion of robustness from ARV: the robustness is defined inductively in both time and the structure of the specification, and assigns a robustness degree at every point in time to each sub-formula. In contrast, we adopt a more global and semantic definition in which robustness is the distance between the observed trace and the boundary of the set of traces that satisfy/violate the specification.

In contrast to ARV, S-TaLiRo and Breach support a fixed specification language (STL) with specific quantitative semantics. For instance, adding support for regular expressions is not a trivial task. It requires developing new inductive robustness definition for regular expression operators and devising completely new algorithms for computing the robustness degree [7]. In contrast, adding regular expressions to ARV only requires an effective procedure that translates the expressions to symbolic automata. In addition, ARV allows plugging-in accumulative semantics without changing the underlying algorithms. We do not see an easy way to adapt the existing algorithms in S-TaLiRo and Breach to accommodate for accumulative quantitative semantics.

We now compare the robustness values obtained by S-TaLiRo, Breach and ARV for semantically equivalent specifications as well as for specifications that are contradictions. In ψ_1 and ψ_2 from Table III we test the sensitivity of the robustness degree algorithm w.r.t. to the minimal set representation. The formula ψ_1 defines the acceptable range $[-30, 30]$ of values for a, while ψ_2 is a semantically equivalent formula that represents the same set as a disjoint union $[-30, 0) \cup [0, 30]$. Similarly, ψ_3 and ψ_4 represent two semantically equivalent temporal formulas. Finally, both ψ_5 and ψ_6 represent formulas that are unsatisfiable.

We can observe that our approach produces results that are invariant to the syntactic representation of the formula. In particular, our monitoring algorithm is able to detect unsatisfiable formulas. In contrast, neither S-TaLiRo nor Breach can detect unsatisfiable specifications. We can also observe that in some cases, these two tools are also sensitive to the syntactic representation of the specification. This is visible in the computation of the robustness for ψ_1 and ψ_2, where S-TaLiRo computes the inconclusive result 0 for a specification that is satisfied by the trace, while Breach correctly finds that the formula is satisfied, but assigns it a very low robustness degree. These results directly follow from the different ways to define robust semantics in the two approaches.

This article was presented in the International Conference on Embedded Software 2018
and appears as part of the ESWEEK-TCAD special issue.

The flexibility of our approach comes at a price. In contrast to S-TaLiRo and Breach that admit monitoring procedures that are linear in length of the input trace and exponential in the size of the formula [25], ARV requires translating specifications to symbolic automata, and in case of STL it results in a monitoring algorithm that is linear in the length of the trace and exponential in the size of the specification, but also in the largest constant appearing in the formula.

VI. Conclusions and Future Work

We presented ARV, a generic algebraic approach to monitor correctness and robustness of CPS applications. We demonstrated the flexibility of ARV w.r.t. specification languages and semantics. We showed that ARV, which relies on the use of automata, enables a precise robustness measurement of a trace with respect to a specification. The definition of the abstract RV algorithm over the automaton structure facilitates code generation of real-time monitors for various platforms, including SW and FPGAs.

We believe that our approach may open many new research and development avenues. Currently we have an enumerative approach to real-time. We will investigate the effect of symbolic representation of time to our automata-based approach.

We have seen that the precision of our semantic approach comes at a price – an exponential blow up in the size of the specification, but also in the largest constant appearing in the formula. In this paper, we sketched the worst-case complexity of our translation. We believe that for many interesting and relevant specifications the effective translation will yield monitors that are much smaller than the worst case. This requires further theoretical and experimental investigations. On the theoretical side, we will study the tight bounds for an optimal translation from STL to symbolic automata with enumerative time. On the experimental side, we will optimize our prototype implementation. To achieve this goal, we will explore different optimization strategies, including the use of the algorithms implemented in the symbolic automata library[3] such as its minimization procedure [18]. We will integrate our robustness monitoring approach to the existing falsification testing frameworks and quantify the improvements due to the preciseness of our algorithms. We will implement several code generators (interpreted Java, Simulink S-functions, embedded C, FPGA, etc.) and investigate the reuse of specifications and monitors across stages in the development cycle.

In this paper, we restricted ourselves to Hausdorff-like measures, and additive idempotent semirings. We would like to study in the future, the extension of our framework to non-idempotent $\oplus$-addition, and its application to RV: e.g. to a probabilistic semiring. We would also like to investigate if we can use our approach to support other types of semantics. We will study whether we can generalize ARV to enable measuring weighted edit distance measure between a trace and a specification as proposed in [36].

Acknowledgment

This research is partially supported by research projects IOSENSE, AutoDrive, CPS/IoT project (HRSM), the Austrian National Research Network (nr. S 11405-N23 and S 11412-N23) SHiNE funded by the Austrian Science Fund (FWF) and the EU ICT COST Action IC1402 on Runtime Verification beyond Monitoring (ARVI). IOSENSE project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 692480. AutoDrive project has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737469. These Joint Undertakings receive support from the European Union's Horizon 2020 research and innovation programme and national authorities. The CPS/IoT project receives support from the Austrian government through the Federal Ministry of Science, Research and Economy (BMWFW) in the funding program Hochschulraum-Strukturmittel (HRSM) 2016.

[3]https://github.com/lorisdanto/symbolicautomata

References

[1] Houssam Abbas, Bardh Hoxha, Georgios E. Fainekos, Jyotirmoy V. Deshmukh, James Kapinski, and Koichi Ueda. Conformance testing as falsification for cyber-physical systems. *CoRR*, abs/1401.5200, 2014.

[2] Houssam Abbas, Hans D. Mittelmann, and Georgios E. Fainekos. Formal property verification in a conformance testing framework. In *Proc. of MEMOCODE 2014: the Twelfth ACM/IEEE International Conference on Formal Methods and Models for Codesign*, pages 155–164. IEEE, 2014.

[3] Takumi Akazaki and Ichiro Hasuo. Time robustness in MTL and expressivity in hybrid system falsification. In *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part II*, pages 356–374, 2015.

[4] Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. *Inf. Comput.*, 104(1):35–77, 1993.

[5] Yashwanth Annpureddy, Che Liu, Georgios E. Fainekos, and Sriram Sankaranarayanan. S-TaLiRo: A tool for temporal logic falsification for hybrid systems. In *Proc. of TACAS 2011: the 17th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 6605 of *LNCS*, pages 254–257. Springer, 2011.

[6] Alexey Bakhirkin, Thomas Ferrère, Oded Maler, and Dogan Ulus. On the quantitative semantics of regular expressions over real-valued signals. In *Formal Modeling and Analysis of Timed Systems - 15th International Conference, FORMATS 2017, Berlin, Germany, September 5-7, 2017, Proceedings*, pages 189–206, 2017.

[7] Alexey Bakhirkin, Thomas Ferrère, Oded Maler, and Dogan Ulus. On the quantitative semantics of regular expressions over real-valued signals. In *Formal Modeling and Analysis of Timed Systems - 15th International Conference, FORMATS 2017, Berlin, Germany, September 5-7, 2017, Proceedings*, pages 189–206, 2017.

[8] Howard Barringer, Allen Goldberg, Klaus Havelund, and Koushik Sen. Program monitoring with LTL in EAGLE. In *18th International Parallel and Distributed Processing Symposium (IPDPS 2004), CD-ROM / Abstracts Proceedings, 26-30 April 2004, Santa Fe, New Mexico, USA*, 2004.

[9] Ezio Bartocci, Luca Bortolussi, Michele Loreti, and Laura Nenzi. Monitoring mobile and spatially distributed cyber-physical systems. In *Proc.of MEMOCODE 2017: the 15th ACM-IEEE International Conference on Formal Methods and Models for System Design*, pages 146–155. ACM, 2017.

[10] Ezio Bartocci, Jyotirmoy Deshmukh, Alexandre Donzé, Georgios Fainekos, Oded Maler, Dejan Nickovic, and Sriram Sankaranarayanan. Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications. In *Lectures on Runtime Verification - Introductory and Advanced Topics*, volume 10457 of *LNCS*, pages 128–168. Springer, 2018.

[11] Andreas Bauer, Martin Leucker, and Christian Schallhart. Runtime verification for LTL and TLTL. *ACM Trans. Softw. Eng. Methodol.*, 20(4):14:1–14:64, 2011.

[12] Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.

[13] M. Boulé and Z. Zilic. Incorporating efficient assertion checkers into hardware emulation. In *Proc. of ICCD*, pages 221–228. IEEE Computer Society Press, 2005.

[14] Lubos Brim, Tomas Vejpustek, David Safránek, and Jana Fabriková. Robustness analysis for value-freezing signal temporal logic. In *Proceedings Second International Workshop on Hybrid Systems and Biology, HSB 2013, Taormina, Italy, 2nd September 2013.*, pages 20–36, 2013.

[15] Krishnendu Chatterjee, Laurent Doyen, and Thomas A Henzinger. Quantitative languages. In *International Workshop on Computer Science Logic*, pages 385–400. Springer, 2008.

[16] A. Dahan, D. Geist, L. Gluhovsky, D. Pidan, G. Shapir, Y. Wolfsthal, L. Benalycherif, R. Kamidem, and Y. Lahbib. Combining system level modeling with assertion based verification. In *Proc. of ISQED 2005: Sixth International Symposium on Quality of Electronic Design*, pages 310–315. IEEE, 2005.

[17] Ben D'Angelo, Sriram Sankaranarayanan, César Sánchez, Will Robinson, Bernd Finkbeiner, Henny B. Sipma, Sandeep Mehrotra, and Zohar Manna. LOLA: runtime monitoring of synchronous systems. In *12th International Symposium on Temporal Representation and Reasoning (TIME 2005), 23-25 June 2005, Burlington, Vermont, USA*, pages 166–174, 2005.

[18] Loris D'Antoni and Margus Veanes. Minimization of symbolic automata. In *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 541–554, 2014.

[19] Loris D'Antoni and Margus Veanes. Extended symbolic finite automata and transducers. *Formal Methods in System Design*, 47(1):93–119, 2015.

[20] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, 2008.

[21] Jyotirmoy V. Deshmukh, Alexandre Donzé, Shromona Ghosh, Xiaoqing Jin, Garvit Juniwal, and Sanjit A. Seshia. Robust online monitoring of signal temporal logic. *Formal Methods in System Design*, 51(1):5–30, 2017.

[22] Adel Dokhanchi, Bardh Hoxha, and Georgios E. Fainekos. On-line monitoring for temporal logic robustness. In *Proc. RV 2014: the 5th International Conference on Runtime Verification*, volume 8734 of *Lecture Notes in Computer Science*, pages 231–246. Springer, 2014.

[23] Adel Dokhanchi, Aditya Zutshi, Rahul T. Sriniva, Sriram Sankaranarayanan, and Georgios E. Fainekos. Requirements driven falsification with coverage metrics. In *2015 International Conference on Embedded Software, EMSOFT 2015, Amsterdam, Netherlands, October 4-9, 2015*, pages 31–40, 2015.

[24] Alexandre 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*, pages 167–170, 2010.

[25] Alexandre Donzé, Thomas Ferrère, and Oded Maler. Efficient robust monitoring for STL. In *Computer Aided Verification (CAV)*, pages 264–279, 2013.

[26] Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *Formal Modeling and Analysis of Timed Systems (FORMATS)*, pages 92–106, 2010.

[27] Tommaso Dreossi, Thao Dang, Alexandre Donzé, James Kapinski, Xiaoqing Jin, and Jyotirmoy V. Deshmukh. Efficient guiding strategies for testing of temporal properties of hybrid systems. In *NASA Formal Methods - 7th International Symposium, NFM 2015, Pasadena, CA, USA, April 27-29, 2015, Proceedings*, pages 127–142, 2015.

[28] Deepak D'Souza and R Matteplackel. A clock-optimal hierarchical monitoring automaton construction for MITL. Technical report, 2013.

[29] Georgios E. Fainekos and George J. Pappas. Robust sampling for MITL specifications. In *Formal Modeling and Analysis of Timed Systems, 5th International Conference, FORMATS 2007, Salzburg, Austria, October 3-5, 2007, Proceedings*, pages 147–162, 2007.

[30] Georgios E. Fainekos and George J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theor. Comput. Sci.*, 410(42):4262–4291, 2009.

[31] Georgios E. Fainekos, Sriram Sankaranarayanan, Franjo Ivancic, and Aarti Gupta. Robustness of model-based simulations. In *Proc. of RTSS 2009: the 30th IEEE Real-Time Systems Symposium*, pages 345–354. IEEE Computer Society, 2009.

[32] B. Finkbeiner and L. Kuhtz. Monitor circuits for ltl with bounded and unbounded future. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5779 LNCS:60–75, 2009.

[33] Rob Gerth, Doron Peled, Moshe Y Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Protocol Specification, Testing and Verification XV*, pages 3–18. Springer, 1995.

[34] Bardh Hoxha, Houssam Abbas, and Georgios E. Fainekos. Benchmarks for temporal logic requirements for automotive systems. In *1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems, ARCH@CPSWeek 2014, Berlin, Germany, April 14, 2014 / ARCH@CPSWeek 2015, Seattle, WA, USA, April 13, 2015.*, pages 25–30, 2014.

[35] Stefan Jaksic, Ezio Bartocci, Radu Grosu, Reinhard Kloibhofer, Thang Nguyen, and Dejan Nickovic. From signal temporal logic to FPGA monitors. In *Proc. of MEMOCODE 2015: the ACM/IEEE International Conference on Formal Methods and Models for Codesign*, pages 218–227, 2015.

[36] Stefan Jaksic, Ezio Bartocci, Radu Grosu, and Dejan Nickovic. Quantitative monitoring of STL with edit distance. In *Proc. of RV 2016: the 16th International Conference on Runtime Verification*, volume 10012 of *LNCS*, pages 201–218. Springer, 2016.

[37] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[38] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Joint International Conferences on Formal Modelling and Analysis of Timed Systems, FORMATS 2004 and Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 152–166, 2004.

[39] Oded Maler and Dejan Nickovic. Monitoring properties of analog and mixed-signal circuits. *STTT*, 15(3):247–268, 2013.

[40] Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.

[41] Mehryar Mohri. Edit-distance of weighted automata: General definitions and algorithms. *Int. J. Found. Comput. Sci.*, 14(6):957–982, 2003.

[42] Dejan Nickovic. *Checking timed and hybrid properties: Theory and applications*. PhD thesis, Université Joseph Fourier, Grenoble, France, 2008.

[43] Dejan Nickovic and Oded Maler. AMT: A property-based monitoring tool for analog systems. In *Formal Modeling and Analysis of Timed Systems, 5th International Conference, FORMATS 2007, Salzburg, Austria, October 3-5, 2007, Proceedings*, pages 304–319, 2007.

[44] Amir Pnueli and Aleksandr Zaks. On the merits of temporal testers. In *25 Years of Model Checking - History, Achievements, Perspectives*, pages 172–195, 2008.

[45] Aurélien Rizk, Grégory Batt, François Fages, and Sylvain Soliman. On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In *Computational Methods in Systems Biology, 6th International Conference, CMSB 2008, Rostock, Germany, October 12-15, 2008. Proceedings*, pages 251–268, 2008.

[46] Alena Rodionova, Ezio Bartocci, Dejan Nickovic, and Radu Grosu. Temporal logic as filtering. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 11–20. ACM, 2016.

[47] Alena Rodionova, Matthew O'Kelly, Houssam Abbas, Vincent Pacelli, and Rahul Mangharam. An autonomous vehicle control stack. In *ARCH17. 4th International Workshop on Applied Verification of Continuous and Hybrid Systems, collocated with Cyber-Physical Systems Week (CPSWeek) on April 17, 2017 in Pittsburgh, PA, USA*, pages 44–51, 2017.

[48] Roopsha Samanta, Jyotirmoy V. Deshmukh, and Swarat Chaudhuri. Robustness analysis of string transducers. In *Proc. of ATVA 2013: the 11th International Symposium on Automated Technology for Verification and Analysis*, volume 8172 of *LNCS*, pages 427–441. Springer, 2013.

[49] Volker Stolz and Eric Bodden. Temporal assertions using aspectj. *Electr. Notes Theor. Comput. Sci.*, 144(4):109–124, 2006.

[50] Margus Veanes, Pieter Hooimeijer, Benjamin Livshits, David Molnar, and Nikolaj Bjørner. Symbolic finite state transducers: algorithms and applications. In *Proceedings of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012, Philadelphia, Pennsylvania, USA, January 22-28, 2012*, pages 137–150, 2012.

This article was presented in the International Conference on Embedded Software 2018
and appears as part of the ESWEEK-TCAD special issue.