


RobTL: Robustness Temporal Logic for CPS

Valentina Castiglioni ✉ 

Eindhoven University of Technology, The Netherlands

Michele Loreti ✉ 

University of Camerino, Italy

Simone Tini ✉ 

University of Insubria, Italy

Abstract

We propose *Robustness Temporal Logic (RobTL)*, a novel temporal logic for the specification and analysis of distances between the behaviours of Cyber-Physical Systems (CPS) over a finite time horizon. RobTL specifications allow us to *measure* the differences in the behaviours of systems with respect to various objectives and temporal constraints, and to study how those differences *evolve in time*. Specifically, the unique features of RobTL allow us to specify *robustness properties* of CPS against uncertainty and perturbations. As an example, we use RobTL to analyse the robustness of an engine system that is subject to attacks aimed at inflicting overstress of equipment.

2012 ACM Subject Classification Theory of computation → Verification by model checking; Theory of computation → Modal and temporal logics

Keywords and phrases Cyber-physical systems, robustness, temporal logic, uncertainty

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2024.15

Supplementary Material *Software*: <https://github.com/quasylab/jspear/tree/working> [9]
archived at `swh:1:dir:ddf418d5a080b8e83323a1b2c38d9f7065e2554`

Funding *Simone Tini*: This study received funding from the European Union – Next-GenerationEU – National Recovery and Resilience Plan (NRRP) – MISSION 4 COMPONENT 2, INVESTMENT N. 1.1, CALL PRIN 2022 D.D. 104 02-02-2022 – MEDICA Project, CUP N. J53D23007180006.

This publication is part of the project *NODES* which has received funding from the MUR – M4C2 1.5 of PNRR with grant agreement no. ECS00000036.

1 Introduction

When systems are subject to *uncertainty* and *perturbations*, like Cyber-Physical Systems (CPS) [35] in which software components, or *agents*, must interact with an unpredictable *environment*, it is crucial to provide some guarantees on their *robustness*. This is the ability of a system to function correctly even in presence of uncontrollable events affecting its behaviour, as, e.g., unexpected physical phenomena, failures, or cyber-physical attacks.

In the literature, we can find a wealth of proposals of robustness properties, that differ in the underlying model (including how uncertainty is modelled), in the formalisation, or in whether they are designed to analyse a specific feature of the behaviour of systems. We refer to [24, 38, 40, 43] for an overview of these notions. Although it seems natural to us that different application contexts call for different formalisations of robustness, the downside of this variety is the lack of a general tool for the verification of robustness properties.

In this paper we provide a formal framework for the verification of robustness properties of CPS. In this setting, robustness is usually formalised as a *measure* of the capability of agents to tolerate perturbations in the environmental conditions and still fulfil their tasks. This boils down to *quantifying the differences* between the behaviour of the system with its behaviour under the effect of perturbations, possibly at *different moments in time*. Intuitively, the system is robust if whenever the two behaviours are initially at a bounded distance, then



© Valentina Castiglioni, Michele Loreti, and Simone Tini;
licensed under Creative Commons License CC-BY 4.0

35th International Conference on Concurrency Theory (CONCUR 2024).

Editors: Rupak Majumdar and Alexandra Silva; Article No. 15; pp. 15:1–15:23



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

their distance after a given amount of time should always be smaller than a given threshold. This means that whenever we require a CPS to be robust against perturbations, we are actually specifying a *property on the evolution in time of distances between behaviours*.

Hence, a formal framework for the specification of similar properties should include:

- A model for the specification of the behaviour of CPS.
- A mechanism for the specification of the effects of perturbations on their behaviour.
- A mechanism to define distances between the behaviours of CPS.
- A temporal logic for the specification of properties on the evolution of those distances.

We adopt the model from the literature: the *evolution sequence model* introduced in [10,11]. The two mechanisms and the logic are introduced in this paper.

The evolution sequence model. The evolution sequence model follows a discrete-time, data-driven approach: the behaviour of the system is modelled in terms of the modifications that the interaction of the agents with the environment induces on a set of application-relevant data, called *data state*. Due to the unpredictability of the environment and potential approximations in the specification of agents, those modifications are modelled as continuous *distributions* on the attainable data states. The *evolution sequence* of a system is then defined as the sequence of the distributions over data states that are obtained at each time step.

The reason why we chose this model over classical and more established ones, like Labelled Markov Chains and Stochastic Hybrid Systems [7,27], is purely technical. The most prominent consequence of the design choices in this model is that the behaviour of the system is not given by a set of traces/trajectories, but by the combination of their effects. In other words, an evolution sequence is the discrete-time version of the cylinder of all possible trajectories of the system. This means that a property of an evolution sequence takes into account the outcomes of all possible observations, at a given time step, on the system. This is fundamental in the verification of robustness, since even the slightest modification induced by uncertainty on behaviour is taken into account.

Robustness Temporal Logic. We introduce *Robustness Temporal Logic (RobTL)* that allows us to compare distances between nominal and perturbed evolution sequences over a finite time horizon, by also providing the means to specify the perturbations and the distances. Specifically, RobTL offers:

- A class of *distance expressions* for the definition of arbitrary distances between evolution sequences. This freedom allows us to compare systems with respect to different aspects of behaviour in time, as well as to combine distances having different formulations.
- A class of *perturbations* for the definition of the effects of unpredictable events on the behaviour of the system.
- Atomic propositions $\Delta(\text{exp}, \mathbf{p}) \bowtie \eta$ to evaluate, at a given time step, the *distance*, specified by a *distance expression* exp , between a given evolution sequence and its perturbed version, obtained by some *perturbation* \mathbf{p} , and to compare it with the threshold η .
- Classical Boolean and temporal operators for the analysis of the evolution of the specified distances over a finite time horizon.

We provide a *statistical model checking algorithm* for the verification of RobTL specifications. As our algorithms are based on statistical inference, we need to account for the statistical error when checking formulae. Hence, we also propose a three-valued semantics for RobTL, in which the truth value *unknown* suggests that the parameters in the property need some tuning, or that a larger number of samples is needed to obtain a precise evaluation of the

distances. To showcase its features, we apply our framework to the analysis of a case-study from Industrial Control Systems: an engine system that is subject to cyber-physical attacks aimed at inflicting overstress of equipment [25].

All the algorithms and examples have been implemented in the tool STARK [12,14].

Why a new logic? RobTL is the only existing temporal logic expressing properties of distances between systems behaviours. Usually, even in logic equipped with a real-valued semantics, the behaviour of a *single given system* is compared to the desired property. Conversely, in RobTL the behaviours of *two systems* are taken into account. More precisely, we distinguish three approaches to the specification of properties in the quantitative setting:

- Specification of properties over a *single trajectory* of the system. This is the classic approach of PCTL, probabilistic LTL, and their variants [4,30,39].
- Specification of properties across the trajectories of the system. This is the *hyper-property* [15] approach of, e.g., HyperPCTL [2] or HPSTL [3], where one can express quantitative dependencies, in the form of bounds on probabilistic weights, between *different independent trajectories* of the system.
- Specification of properties based on the comparison of *all possible trajectories* of *two different systems*. This is the approach of RobTL, where one system is the perturbed version of the other. This feature also distinguishes our approach to robustness from classical ones, like those in [18,22]. Our properties are based on the comparison of the evolution sequences of two different systems, whereas [18,22] compare a single trajectory of a single system with the set of the behaviours that satisfy a given property, which is specified by means of a formula expressed in a suitable temporal logic, like, e.g., STL.

2 The Evolution Sequence Model

We recall the main elements of the evolution sequence model [11]. Systems consist of a set of *agents* and an *environment*, whose interaction produces changes on a *data space* \mathcal{D} , containing the values assumed by *variables*, from a *finite* set Var , representing:

- (i) physical quantities,
- (ii) sensors,
- (iii) actuators, and
- (iv) internal variables of agents.

For each $x \in \text{Var}$, the domain $\mathcal{D}_x \subseteq \mathbb{R}$ is either *finite*, or a *compact* subset of \mathbb{R} (and thus Polish), and equipped with the Borel σ -algebra \mathcal{B}_x . Then $\mathcal{D} = \prod_{x \in \text{Var}} \mathcal{D}_x$ and we equip it with the product σ -algebra $\mathcal{B}_{\mathcal{D}} = \prod_{x \in \text{Var}} \mathcal{B}_x$ [6]. Let $\Pi(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$ be the set of distributions over $(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$.

We call *data state* the current state of the data space, and represent it by a mapping $\mathbf{d}: \text{Var} \rightarrow \mathbb{R}$, with $\mathbf{d}(x) \in \mathcal{D}_x$ for all $x \in \text{Var}$. At each step, the agents and the environment induce some changes on the data state, providing a new data state at the next step. Those modifications are also subject to the presence of uncertainties, meaning that it is not always possible to determine exactly the values assumed by the data at the next step. Hence, we model the changes induced at each step as a distribution on the attainable data states. The behaviour of the system is then expressed by its *evolution sequence*, i.e., the sequence of distributions over the data states obtained at each step.

In this paper we do not focus on how evolution sequences are generated. In [11, Prop. 3.15] it was proved that the function defining the combined behaviour of the agents and the environment, specified according to the framework, is a *Markov kernel*. Hence, we simply assume a Markov kernel $\text{step}: \mathcal{D} \rightarrow \Pi(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$ governing the evolution of the system, and define the evolution sequence as the *Markov process* generated by step (Definition 1): $\text{step}(\mathbf{d})(\mathbb{D})$

expresses the probability to reach a data state in \mathbb{D} from \mathbf{d} in one computation step. Indeed, each system is characterised by a particular function **step** starting from an *initial distribution* over \mathcal{D} . For instance, for the engine system of our case study (presented below), the initial distribution is a Dirac distribution over a chosen data state, and function **step** is obtained by combining the effects of the agents in Figure 1c and the environment in Figure 1d.

► **Definition 1.** Let $\text{step}: \mathcal{D} \rightarrow \Pi(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$ be the Markov kernel generating the behaviour of a system \mathbf{s} having μ as initial distribution. The evolution sequence of \mathbf{s} is a countable sequence $\mathcal{S}_{\mu} = \mathcal{S}_{\mu}^0, \mathcal{S}_{\mu}^1, \dots$ of distributions in $\Pi(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$ such that, for all $\mathbb{D} \in \mathcal{B}_{\mathcal{D}}$:

$$\mathcal{S}_{\mu}^0(\mathbb{D}) = \mu(\mathbb{D}) \quad \text{and} \quad \mathcal{S}_{\mu}^{i+1}(\mathbb{D}) = \int_{\mathcal{D}} \text{step}(\mathbf{d})(\mathbb{D}) d\mathcal{S}_{\mu}^i(\mathbf{d}).$$

► **Remark 2.** We shall write $\mathcal{S}, \mathcal{S}_1$ in place of, respectively, $\mathcal{S}_{\mu}, \mathcal{S}_{\mu_1}$, whenever the formalisation of the initial distributions μ, μ_1 does not play a direct role in the discussion.

Case study: the engine system. As a running example, we consider a refrigerated engine system [31], sketched in Figure 1. There is one agent with three tasks:

- (i) regulate the speed,
- (ii) maintain the temperature within a specific range by means of a cooling system, and
- (iii) detect anomalies.

The first two tasks are on charge of a controller, the other is took over by an intrusion detection system, henceforth IDS. Figure 1a shows that these two components use **channels** to exchange information and to communicate with other agents. The variables used in the system, and their role, are listed in Figure 1b. Variable *stress* quantifies the level of *equipment stress*, which increases when the temperature stays too often above the threshold 100: the higher the stress, the higher the probability of a wreckage. The agent and the environment acting on these data have been specified in STARK, and are reported in Figure 1c and 1d, respectively. At each scan cycle the controller sets **internal variables** and **actuators** according to the values received from **sensors**, the IDS raises a warning if the status of **sensors** and **actuators** is unexpected, and the environment models the probabilistic evolution of the temperature. Notice that the controller must use channel *ch_temp* to receive data from sensor *temp*. Even though the use of channels is a common feature in CPS, it exposes them to attacks, as we will discuss in Example 11. We assume that the engine can cooperate with other engines (e.g., in an aircraft with a *left* and a *right* engine), by receiving values on channel *ch_in* and sending values on *ch_out*. If the engine is required to work at slow speed, it asks to other engines to proceed at full speed to compensate the lack of performance.

3 Robustness Temporal Logic

Robustness Temporal Logic (RobTL) allows us to express temporal properties of distances over systems behaviour, and, thus, to specify and verify robustness properties of CPS against uncertainty and perturbations. RobTL uses atomic propositions of the form $\Delta(\text{exp}, \mathbf{p}) \bowtie \eta$ to evaluate, at a given time step, the *distance*, specified by an *expression* **exp**, between a given evolution sequence and its perturbed version, obtained by some *perturbation* **p**, and to compare it with the threshold η . Atomic propositions are then combined with classic Boolean and temporal operators, in order to extend and compare these evaluations over the chosen time horizon. Hence, RobTL formulae are defined over three main components:

1. A language **DistExp** to specify *distances*.
2. A language **Pert** to specify *perturbations*.
3. Classic Boolean and temporal operators to specify requirements on the *evolution of distances in time*.

Then, we introduce the language `DistExp` whose operators allow us to combine various instances of this ground distance and define more complex distances over evolution sequences, while possibly taking into account *different objectives* of the system and *temporal constraints*.

Ground distance on distributions. In our setting, as in most application contexts, the objectives of the system can be expressed in a purely data-driven fashion: at any step, any difference between the desired value of some parameters of interest and the data actually obtained can be interpreted as a flaw in the behaviour of the system. Hence, following [10], to capture a particular objective, we use *penalty functions* $\rho: \mathcal{D} \times \mathbb{N} \rightarrow [0, 1]$ assigning to each pair \mathbf{d} , τ a *penalty* in $[0, 1]$, expressing how far the values of the parameters related to the considered task in \mathbf{d} are from their desired ones at step τ . Hence, $\rho(\mathbf{d}, \tau) = 0$ if \mathbf{d} respects all the parameters at step τ . For brevity, we let $\rho_\tau(\mathbf{d}) = \rho(\mathbf{d}, \tau)$. We remark that since we are in a the discrete-time setting, we can safely identify time steps with natural numbers.

► **Example 3.** Consider the engine system from our case study. We define the penalty functions ρ^w , ρ^t , and ρ^s , for all time steps τ , by:

$$\rho_\tau^w(\mathbf{d}) = \begin{cases} 1 & \text{if } \mathbf{d}(ch_wrn) = \text{hot,} \\ 0 & \text{if } \mathbf{d}(ch_wrn) = \text{ok} \end{cases} \quad \begin{cases} \rho_\tau^t(\mathbf{d}) = |\mathbf{d}(ch_temp) - \mathbf{d}(temp)|/150 \\ \rho_\tau^s(\mathbf{d}) = \mathbf{d}(stress) \end{cases}$$

They express, respectively, how far the level of alert raised by the IDS, the value carried by channel ch_temp , and the level of stress are from their desired value. These coincide with the value `ok`, the value of sensor $temp$, and zero, respectively.

Penalty functions can be used also to express *false negatives* and *false positives*, representing, respectively, the average *effectiveness*, and the average *precision* of the IDS to signal through channel ch_wrn that the engine system is under stress. We use two new variables fn and fp to quantify false negatives and false positives depending on $stress$ and ch_wrn . Both variables are initialised to 0 and updated as follows:

$$fn(\tau+1) = \frac{\tau * fn(\tau) + \max(0, stress(\tau) - \gamma)}{\tau + 1} \quad fp(\tau+1) = \frac{\tau * fp(\tau) + \max(0, \gamma - stress(\tau))}{\tau + 1}$$

where γ is 0 if $ch_wrn(\tau)$ is `ok`, and γ is 1 if $ch_wrn(\tau)$ is `hot`. Then, penalties $\rho_\tau^{fn}(\mathbf{d}) = \mathbf{d}(fn)$ and $\rho_\tau^{fp}(\mathbf{d}) = \mathbf{d}(fp)$ express how far fn and fp are from their ideal value 0.

We can then make use of penalty functions to define a *distance on data states*:

► **Definition 4.** Let ρ be a penalty function, and $\tau \in \mathbb{N}$. The metric on data states in \mathcal{D} , $\mathbf{m}_\tau^\rho: \mathcal{D} \times \mathcal{D} \rightarrow [0, 1]$, is defined, for all $\mathbf{d}_1, \mathbf{d}_2 \in \mathcal{D}$, by $\mathbf{m}_\tau^\rho(\mathbf{d}_1, \mathbf{d}_2) = \max\{\rho_\tau(\mathbf{d}_2) - \rho_\tau(\mathbf{d}_1), 0\}$.

Note that $\mathbf{m}_\tau^\rho(\mathbf{d}_1, \mathbf{d}_2)$ is a hemimetric, i.e., a pseudometric that is not required to be symmetric, expressing how much \mathbf{d}_2 is *worse* than \mathbf{d}_1 according to ρ_τ .

Finally, we need to lift the hemimetric \mathbf{m}_τ^ρ to a hemimetric over $\Pi(\mathcal{D}, \mathcal{B}_\mathcal{D})$. In the literature, we can find a wealth of notions of function lifting doing so (see [34] for a survey). Among those, the *Wasserstein lifting* [47] has the following advantages:

- (i) it preserves the properties of the ground metric, and
- (ii) one can apply statistical inference to obtain good approximations of it, whose exact computation is tractable [11, 44, 46].

► **Definition 5.** Let ρ be a penalty function, and $\tau \in \mathbb{N}$. For any $\mu, \nu \in \Pi(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$, the Wasserstein lifting of \mathbf{m}_{τ}^{ρ} to a distance between μ and ν is defined by:

$$\mathbf{W}(\mathbf{m}_{\tau}^{\rho})(\mu, \nu) = \inf_{\mathfrak{w} \in \mathfrak{W}(\mu, \nu)} \int_{\mathcal{D} \times \mathcal{D}} \mathbf{m}_{\tau}^{\rho}(\mathbf{d}, \mathbf{d}') d\mathfrak{w}(\mathbf{d}, \mathbf{d}')$$

where $\mathfrak{W}(\mu, \nu)$ is the set of the couplings of μ and ν , namely the set of joint distributions \mathfrak{w} over the product space $(\mathcal{D} \times \mathcal{D}, \mathcal{B}_{\mathcal{D} \times \mathcal{D}})$ having μ and ν as left and right marginal, respectively, i.e., $\mathfrak{w}(\mathbb{D} \times \mathcal{D}) = \mu(\mathbb{D})$ and $\mathfrak{w}(\mathcal{D} \times \mathbb{D}) = \nu(\mathbb{D})$, for all $\mathbb{D} \in \mathcal{B}_{\mathcal{D}}$. (See [11, 23] and Appendix A for an in depth discussion on the definition of the Wasserstein lifting over hemimetrics.)

► **Proposition 6** ([11]). For each penalty function ρ , and time step $\tau \in \mathbb{N}$, function $\mathbf{W}(\mathbf{m}_{\tau}^{\rho})$ is a 1-bounded hemimetric on $\Pi(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$.

The language DistExp. We can now proceed to define distances that take into account several time steps in the evolution of systems.

► **Definition 7.** Expressions in **DistExp** are defined as follows:

$$\begin{aligned} \text{exp} ::= & \langle^{\rho} \mid \rangle^{\rho} \mid \mathbf{F}^I \text{exp} \mid \mathbf{G}^I \text{exp} \mid \text{exp} \mathbf{U}^I \text{exp} \mid \\ & \min(\text{exp}, \text{exp}) \mid \max(\text{exp}, \text{exp}) \mid \sum_{k \in K} w_k \cdot \text{exp}_k \mid \sigma(\text{exp}, \bowtie \zeta) \end{aligned}$$

where ρ ranges over penalty functions, I is an interval, K is a finite set of indexes, $w_k \in (0, 1]$ for each $k \in K$, $\sum_{k \in K} w_k = 1$, $\bowtie \in \{<, \leq, \geq, >\}$ and $\zeta \in [0, 1]$.

Atomic expressions \langle^{ρ} and \rangle^{ρ} are used to evaluate the ground distance with respect to the penalty function ρ . We have two distinct atomic expressions because the ground distance is a hemimetric: the direction of the arrowhead in \langle^{ρ} and \rangle^{ρ} identifies which argument is considered as the *first* one in the evaluation (cf. Definition 8 below). Moreover, having penalty functions as parameters allows us to study the differences in the behaviour of systems with respect to different data and objectives in time. Then, we provide three *temporal expression* operators, namely \mathbf{F}^I , \mathbf{G}^I and \mathbf{U}^I , allowing for the evaluation of minimal and maximal distances over a given time interval I . The *comparison operator* $\sigma(\text{exp}, \bowtie \zeta)$ returns a value in $\{0, 1\}$ used to establish whether the evaluation of **exp** is in relation \bowtie with the threshold ζ .

Distance expressions are evaluated over two evolution sequences and a time τ , representing the time step at which (or starting from which) the differences are computed.

► **Definition 8.** Let $\mathcal{S}_1, \mathcal{S}_2$ be to evolution sequences, and τ be a time step. The evaluation of distance expressions in the triple $\mathcal{S}_1, \mathcal{S}_2, \tau$ is the function $\llbracket \cdot \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^{\tau} : \text{DistExp} \rightarrow [0, 1]$ defined inductively on the structure of expressions as follows:

- $\llbracket \langle^{\rho} \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^{\tau} = \mathbf{W}(\mathbf{m}_{\tau}^{\rho})(\mathcal{S}_1^{\tau}, \mathcal{S}_2^{\tau});$
- $\llbracket \rangle^{\rho} \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^{\tau} = \mathbf{W}(\mathbf{m}_{\tau}^{\rho})(\mathcal{S}_2^{\tau}, \mathcal{S}_1^{\tau});$
- $\llbracket \mathbf{F}^I \text{exp} \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^{\tau} = \min_{t \in I + \tau} \llbracket \text{exp} \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^t;$
- $\llbracket \mathbf{G}^I \text{exp} \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^{\tau} = \max_{t \in I + \tau} \llbracket \text{exp} \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^t;$
- $\llbracket \text{exp}_1 \mathbf{U}^I \text{exp}_2 \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^{\tau} = \min_{t \in I + \tau} \max \left\{ \llbracket \text{exp}_2 \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^t, \max_{t' \in I + \tau, t' < t} \llbracket \text{exp}_1 \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^{t'} \right\};$
- $\llbracket \min(\text{exp}_1, \text{exp}_2) \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^{\tau} = \min \{ \llbracket \text{exp}_1 \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^{\tau}, \llbracket \text{exp}_2 \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^{\tau} \};$
- $\llbracket \max(\text{exp}_1, \text{exp}_2) \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^{\tau} = \max \{ \llbracket \text{exp}_1 \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^{\tau}, \llbracket \text{exp}_2 \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^{\tau} \};$
- $\llbracket \sum_{k \in K} w_k \text{exp}_k \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^{\tau} = \sum_{k \in K} w_k \cdot \llbracket \text{exp}_k \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^{\tau};$
- $\llbracket \sigma(\text{exp}, \bowtie \zeta) \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^{\tau} = \begin{cases} 0 & \text{if } \llbracket \text{exp} \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^{\tau} \bowtie \zeta, \\ 1 & \text{otherwise.} \end{cases}$

The evaluation bases on the two atomic expressions. We use $<^\rho$ to measure the distance between the distributions reached by \mathcal{S}_1 and \mathcal{S}_2 at time τ (\mathcal{S}_τ^1 and \mathcal{S}_τ^2) with respect to the penalty function ρ , i.e., $\mathbf{W}(\mathbf{m}_\tau^\rho)(\mathcal{S}_\tau^1, \mathcal{S}_\tau^2)$. Conversely, $>^\rho$ measures the distance $\mathbf{W}(\mathbf{m}_\tau^\rho)(\mathcal{S}_\tau^2, \mathcal{S}_\tau^1)$. Operators F^I , G^I , and U^I can be thought of as the quantitative versions of the corresponding bounded temporal operators, respectively, *eventually*, *always*, and *until*. Their semantics follows by associating existential quantification with minima, and universal quantification with maxima. Hence, the evaluation of $F^I \mathbf{exp}$ is obtained as the minimum value of the distance \mathbf{exp} over the time interval I . Dually, $G^I \mathbf{exp}$ gives us the maximum value of \mathbf{exp} over I . Then, the evaluation of $\mathbf{exp}_1 U^I \mathbf{exp}_2$ follows from that of bounded until (see Definition 14), accordingly. The expression $\sigma(\mathbf{exp}, \bowtie \zeta)$ evaluates to 0 if the evaluation of \mathbf{exp} is $\bowtie \zeta$; otherwise it evaluates to 1. Informally, the comparison operator σ can be combined with temporal expression operators to check if several constraints of the form $\bowtie \zeta_i$ are satisfied over a time interval under a single application of a perturbation function (see Example 16 below).

3.2 Perturbations

A *perturbation* is the effect of unpredictable events on the current state of the system. Hence, we model it as a function that maps a data state into a distribution over data states. To account for possibly repeated, or different effects in time of a single perturbation, we make the definition of perturbation function also time-dependent: a perturbation function \mathbf{p} is a list of mappings in which the i -th element describes the effects of \mathbf{p} at time i .

► **Definition 9.** A perturbation function is a mapping $\mathbf{p}: \mathcal{D} \times \mathbb{N} \rightarrow \Pi(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$ such that, for each $\tau \in \mathbb{N}$, the mapping $\mathbf{d} \mapsto \mathbf{p}(\mathbf{d}, \tau)(\mathbb{D})$ is $\mathcal{B}_{\mathcal{D}}$ -measurable for all $\mathbb{D} \in \mathcal{B}_{\mathcal{D}}$.

To describe the perturbed behaviour of a system, we need to account for the effects of a function \mathbf{p} on the evolution sequence. Hence, we combine \mathbf{p} with the Markov kernel **step**:

► **Definition 10.** Given an evolution sequence \mathcal{S}_μ generated by kernel **step**, and a perturbation function \mathbf{p} , the perturbation of \mathcal{S}_μ via \mathbf{p} is the evolution sequence $\mathcal{S}_\mu^{\mathbf{p}}$ obtained by:

$$\mathcal{S}_\mu^{\mathbf{p},0}(\mathbb{D}) = \int_{\mathcal{D}} \mathbf{p}(\mathbf{d}, 0)(\mathbb{D}) d\mu(\mathbf{d}), \quad \mathcal{S}_\mu^{\mathbf{p},i+1}(\mathbb{D}) = \int_{\mathcal{D}} \int_{\mathcal{D}} \mathbf{p}(\mathbf{d}', i+1)(\mathbb{D}) d\text{step}(\mathbf{d})(\mathbf{d}') d\mathcal{S}_\mu^{\mathbf{p},i}(\mathbf{d}).$$

Specifying perturbations. We specify a perturbation function \mathbf{p} via a (syntactic) perturbation \mathbf{p} in the language **Pert**:

$$\mathbf{p} ::= \text{nil} \quad | \quad \mathbf{f}@_\tau \quad | \quad \mathbf{p}_1; \mathbf{p}_2 \quad | \quad \mathbf{p}^n$$

where \mathbf{p} ranges over **Pert**, n and τ are finite natural numbers, and:

- **nil** is the perturbation with *no effects*, i.e., at each time step it behaves like the identity function $\text{id}: \mathcal{D} \rightarrow \Pi(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$ such that $\text{id}(\mathbf{d}) = \delta_{\mathbf{d}}$ for all $\mathbf{d} \in \mathcal{D}$;
- **f@ τ** is an *atomic perturbation*, i.e., a function $\mathbf{f}: \mathcal{D} \rightarrow \Pi(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$ such that the mapping $\mathbf{d} \mapsto \mathbf{f}(\mathbf{d})(\mathbb{D})$ is $\mathcal{B}_{\mathcal{D}}$ -measurable for all $\mathbb{D} \in \mathcal{B}_{\mathcal{D}}$, and that is applied precisely after τ time steps from the current instant;
- **p₁; p₂** is a *sequential perturbation*, i.e., perturbation \mathbf{p}_2 is applied at the time step subsequent to the (final) application of \mathbf{p}_1 ;
- **pⁿ** is an *iterated perturbation*, i.e., perturbation \mathbf{p} is applied for a total of n times.

Despite its simplicity, this language allows us to define some non-trivial perturbation functions that we can use to test systems behaviour. (Unspecified perturbations behave like **id**.)

► **Example 11.** In [31] several cyber-physical attacks tampering with sensors or actuators of the engine system aiming to inflict *overstress of equipment* [25] were described. Here we show how those attacks can be modelled by employing our perturbations.

Consider sensor *temp*. There is an attack that aims at delaying the cooling phase, forcing the system to work for several instants at high temperatures and accumulate stress. It tricks the controller by adding a negative offset $o \in \mathbb{R}^{\leq 0}$ to the value carried by the insecure channel *ch_temp*. If $[100, 200]$ is the attack window, the attack is modelled by perturbation $\mathbf{p}_{temp,o} = (\mathbf{id}@0)^{100}; (\mathbf{f}_{temp,o}@0)^{100}$, where $\mathbf{f}_{temp,o}(\mathbf{d}) = \delta_{\mathbf{d}'}$ with $\mathbf{d}'(ch_temp) = \mathbf{d}(ch_temp) + \mathbf{rnd} * o$, with \mathbf{rnd} uniformly distributed in $[0, 1]$, and $\mathbf{d}'(x) = \mathbf{d}(x)$ for all other variables in Figure 1b.

Consider now actuator *cool*. There is an attack aims at forcing the system to reach quickly high temperatures after the start of a cooling phase. It switches off the cooling system as soon as the temperatures goes below $99.8 - t$ degrees, for some $t \in \mathbb{R}^{\geq 0}$. This attack, is *stealth*, meaning that the IDS does not detect it. If $[0, 100]$ is the attack window, the attack is modelled by perturbation $\mathbf{p}_{cool,t}$ defined by $\mathbf{p}_{cool,t} = (\mathbf{f}_{cool,t}@0)^{100}$, where $\mathbf{f}_{cool,t}(\mathbf{d}) = \delta_{\mathbf{d}}$, if $\mathbf{d}(temp) \geq 99.8 - t$, and $\mathbf{f}_{cool,t}(\mathbf{d}) = \delta_{\mathbf{d}'}$, otherwise, with $\mathbf{d}''(cool) = \mathbf{off}$, and $\mathbf{d}''(x) = \mathbf{d}(x)$ for all other variables in Figure 1b.

Each $\mathbf{p} \in \mathbf{Pert}$ denotes a perturbation function as in Definition 9. To obtain it, we exploit function $\mathbf{effect}(\mathbf{p})$, describing the effects of \mathbf{p} at the current step, and function $\mathbf{next}(\mathbf{p})$, identifying the perturbation to be applied at next step. Both functions are defined inductively on the structure of perturbations.

$$\begin{aligned} \mathbf{effect}(\mathbf{nil}) &= \mathbf{id} & \mathbf{effect}(\mathbf{f}@_\tau) &= \begin{cases} \mathbf{id} & \text{if } \tau > 0, \\ \mathbf{f} & \text{if } \tau = 0 \end{cases} \\ \mathbf{effect}(\mathbf{p}^n) &= \mathbf{effect}(\mathbf{p}) & \mathbf{effect}(\mathbf{p}_1; \mathbf{p}_2) &= \mathbf{effect}(\mathbf{p}_1) \\ \mathbf{next}(\mathbf{nil}) &= \mathbf{nil} & \mathbf{next}(\mathbf{f}@_\tau) &= \begin{cases} \mathbf{f}@(\tau - 1) & \text{if } \tau > 0, \\ \mathbf{nil} & \text{otherwise} \end{cases} \\ \mathbf{next}(\mathbf{p}^n) &= \begin{cases} \mathbf{next}(\mathbf{p}); \mathbf{p}^{n-1} & \text{if } n > 0, \\ \mathbf{nil} & \text{otherwise} \end{cases} & \mathbf{next}(\mathbf{p}_1; \mathbf{p}_2) &= \begin{cases} \mathbf{next}(\mathbf{p}_1); \mathbf{p}_2 & \text{if } \mathbf{next}(\mathbf{p}_1) \neq \mathbf{nil}, \\ \mathbf{p}_2 & \text{otherwise.} \end{cases} \end{aligned}$$

We define the semantics of perturbations as the mapping $\langle\langle \cdot \rangle\rangle : \mathbf{Pert} \rightarrow (\mathcal{D} \times \mathbb{N} \rightarrow \Pi(\mathcal{D}, \mathcal{B}_{\mathcal{D}}))$ such that, for all $\mathbf{d} \in \mathcal{D}$ and $i \in \mathbb{N}$, $\langle\langle \mathbf{p} \rangle\rangle(\mathbf{d}, i) = \mathbf{effect}(\mathbf{next}^i(\mathbf{p}))(\mathbf{d})$, where $\mathbf{next}^0(\mathbf{p}) = \mathbf{p}$ and $\mathbf{next}^i(\mathbf{p}) = \mathbf{next}(\mathbf{next}^{i-1}(\mathbf{p}))$, for all $i > 0$.

► **Proposition 12.** *For each $\mathbf{p} \in \mathbf{Pert}$, $\langle\langle \mathbf{p} \rangle\rangle$ is a well defined perturbation function.*

Proof. Since, by definition, each \mathbf{f} occurring in atomic perturbations is such that $\mathbf{d} \mapsto \mathbf{f}(\mathbf{d})(\mathbb{D})$ is $\mathcal{B}_{\mathcal{D}}$ -measurable for all $\mathbb{D} \in \mathcal{B}_{\mathcal{D}}$, and the same property trivially holds for the identity function \mathbf{id} , it is immediate to conclude that $\langle\langle \mathbf{p} \rangle\rangle$ satisfies Definition 9 for each $\mathbf{p} \in \mathbf{Pert}$. ◀

3.3 RobTL Formulae

We use RobTL formulae to specify and analyse distances between nominal and perturbed evolution sequences over a finite time horizon \mathfrak{h} . By combining atomic propositions with temporal operators, we can apply (possibly) different distance expressions and perturbations at different steps, thus allowing for an analysis of behaviour in complex scenarios.

15:10 RobTL: Robustness Temporal Logic for CPS

► **Definition 13.** *RobTL consists in the set of formulae L defined by:*

$$\varphi ::= \top \mid \Delta(\text{exp}, \mathbf{p}) \bowtie \eta \mid \neg \varphi \mid \varphi \wedge \varphi \mid \varphi \mathcal{U}^I \varphi$$

where φ ranges over L , exp ranges over distance expressions in DistExp , \mathbf{p} ranges over perturbations in Pert , $\bowtie \in \{<, \leq, \geq, >\}$, $\eta \in [0, 1]$, and $I \subseteq [0, h]$ is a bounded time interval.

Formulae are evaluated in an evolution sequence and a time step.

► **Definition 14.** *Let \mathcal{S} be an evolution sequence, and τ a time step. The satisfaction relation \models is defined inductively on the structure of formulae as:*

- $\mathcal{S}, \tau \models \top$ for all \mathcal{S}, τ ;
- $\mathcal{S}, \tau \models \Delta(\text{exp}, \mathbf{p}) \bowtie \eta$ iff $[\text{exp}]_{\mathcal{S}, \mathcal{S}_{|\langle\langle \mathbf{p} \rangle\rangle, \tau}^\tau} \bowtie \eta$, with evolution sequence $\mathcal{S}_{|\langle\langle \mathbf{p} \rangle\rangle, \tau}$ defined as:

$$(\mathcal{S}_{|\langle\langle \mathbf{p} \rangle\rangle, \tau})^t = \begin{cases} \mathcal{S}^t & \text{if } t < \tau, \\ \mathcal{S}_{\mathcal{S}^\tau}^{\langle\langle \mathbf{p} \rangle\rangle, t-\tau} & \text{if } t \geq \tau; \end{cases}$$

- $\mathcal{S}, \tau \models \neg \varphi$ iff $\mathcal{S}, \tau \not\models \varphi$;
- $\mathcal{S}, \tau \models \varphi_1 \wedge \varphi_2$ iff $\mathcal{S}, \tau \models \varphi_1$ and $\mathcal{S}, \tau \models \varphi_2$;
- $\mathcal{S}, \tau \models \varphi_1 \mathcal{U}^I \varphi_2$ iff there is a $\tau' \in I + \tau$ such that $\mathcal{S}, \tau' \models \varphi_2$ and for all $\tau'' \in I + \tau$, $\tau'' < \tau'$ it holds that $\mathcal{S}, \tau'' \models \varphi_1$, where, for $I = [a, b]$, we let $I + \tau = [\min\{a + \tau, h\}, \min\{b + \tau, h\}]$.

Let us focus on atomic propositions. The evolution sequence \mathcal{S} at time τ satisfies the formula $\Delta(\text{exp}, \mathbf{p}) \bowtie \eta$ if the distance defined by exp between \mathcal{S} and $\mathcal{S}_{|\langle\langle \mathbf{p} \rangle\rangle, \tau}$ is $\bowtie \eta$, where $\mathcal{S}_{|\langle\langle \mathbf{p} \rangle\rangle, \tau}$ is the evolution sequence obtained by applying the perturbation \mathbf{p} to \mathcal{S} at time τ . For the first $\tau - 1$ steps $\mathcal{S}_{|\langle\langle \mathbf{p} \rangle\rangle, \tau}$ is identical to \mathcal{S} . At time τ the perturbation \mathbf{p} is applied, and the distributions in $\mathcal{S}_{|\langle\langle \mathbf{p} \rangle\rangle, \tau}$ are thus given by the perturbation via $\langle\langle \mathbf{p} \rangle\rangle$ of the evolution sequence having \mathcal{S}^τ as initial distribution (Definition 10). It is worth noticing that by combining atomic propositions with temporal operators we can apply (possibly) different perturbations at different time steps, thus allowing for the analysis of systems behaviour in complex scenarios.

Naturally, other operators can be defined as macros in our logic:

$$\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2) \quad \varphi_1 \implies \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2 \quad \diamond^I \varphi \equiv \top \mathcal{U}^I \varphi \quad \square^I \varphi \equiv \neg \diamond^I \neg \varphi.$$

We now provide some examples of robustness properties that can be expressed in RobTL.

► **Example 15.** By using the penalty functions in Example 3 and the perturbations from Example 11, we can build a formula φ_1 expressing that the attack on the insecure channel ch_temp is successful. This happens if, whenever the difference observed along the attack window $I = [100, 200]$ between the physical value of temperature and that read by the controller is in the interval $[\eta_1, \eta_2]$, for suitable η_1 and η_2 , then the level of the alarm raised by the IDS remains below a given stealthiness threshold η_3 , and the level of system stress overcomes a danger threshold η_4 within the time interval $J = [100, 210]$:

$$\begin{aligned} \varphi_1 &= \diamond^{[0, h]}(\varphi'_1 \implies \varphi''_1) \\ \varphi'_1 &= \Delta(\mathbf{F}^I \langle^{\rho^t}, \mathbf{p}_{temp, o}\rangle \geq \eta_1 \wedge \Delta(\mathbf{G}^I \langle^{\rho^t}, \mathbf{p}_{temp, o}\rangle \leq \eta_2) \\ \varphi''_1 &= \Delta(\mathbf{G}^J \langle^{\rho^w}, \mathbf{p}_{temp, o}\rangle \leq \eta_3 \wedge \Delta(\mathbf{G}^J \langle^{\rho^s}, \mathbf{p}_{temp, o}\rangle \geq \eta_4). \end{aligned}$$

► **Example 16.** Consider the attack on actuator *cool* described in Example 11. We give a formula φ_2 expressing that such an attack fails within 210 units of time. This happens if the level of the alarm raised by the IDS goes above a given threshold ζ_2 at some instant $\tau' \in [0, 210]$, i.e., the attack is detected, while the level of stress remains below an acceptable threshold ζ_1 :

$$\varphi_2 = \Delta \left(\sigma(\langle \rho^s, < \zeta_1 \rangle) \mathcal{U}^{[0,210]} \sigma(\langle \rho^w, > \zeta_2 \rangle, \mathbf{p}_{cool,t}) \right) < 1.$$

Notice that in the formula φ_2 the perturbation $\mathbf{p}_{cool,t}$ is applied only once, at time 0, and by means of the comparison and until operators on expressions we can evaluate all the distances along the considered interval between the original evolution sequence and its perturbation via $\mathbf{p}_{cool,t}$. Conversely, in the formula φ_3 below, the time step at which a perturbation is applied is determined by the bounded until operator:

$$\varphi_3 = \varphi_2 \mathcal{U}^{[\tau_1, \tau_2]} \Delta(\langle \rho^{fn}, \mathbf{p}_{cool,t} \rangle \leq \eta_3).$$

This formula is satisfied if there is a $\tilde{\tau} \in [\tau_1, \tau_2]$ s.t.:

1. the attack on actuator *cool* is detected regardless of the time step in $[\tau_1, \tilde{\tau})$ at which $\mathbf{p}_{cool,t}$ is applied, and
2. the IDS is effective, up to tolerance η_3 , against an application of $\mathbf{p}_{cool,t}$ at time $\tilde{\tau}$.

The effectiveness is measured in terms of the penalty function ρ^{fn} on false negatives presented in Example 3.

4 RobTL model checker

A RobTL model checker has been implemented as part of the *Software Tool for the Analysis of Robustness in the unKnown environment* (STARK) [12, 14], available at <https://github.com/quasylab/jspear/tree/working>, and the related, detailed, documentation can be found at <https://github.com/quasylab/jspear/wiki>.

It consists of the following four procedures, based on *statistical techniques* and *simulation*:

- (i) Simulation of the evolution sequence of system, assuming an initial distribution μ .
- (ii) Simulation of the effects of a perturbation on a given evolution sequence.
- (iii) Syntax driven estimation of the evaluation of distance expressions.
- (iv) Satisfaction of a given RobTL formula by a given evolution sequence.

These four procedures are presented with more details below. Due to space constraints, since all the algorithms are implemented in STARK, we only report them in Appendix B. We also remark that the simulation procedure in (i) and that for the estimation of the Wasserstein distance that is part of (iii) were already discussed at length in [11]. We briefly report them here for the sake of readability. We discuss the time complexity of the model checking algorithm in Section 4.1. Since the procedures outlined above are based on statistical inference, we need to take into account the statistical error when checking the satisfaction of formulae. Hence, in Section 4.2 we discuss a classical algorithm for the evaluation of confidence intervals in the evaluation of distances. Then, we propose a three-valued semantics for RobTL specifications, in which the truth value *unknown* is added to true and false.

Throughout the section, we also provide some examples of an application of the algorithms to the analysis of the engine system. The interested reader can find the script used to generate the plots and results as the `Main.java` file at <https://github.com/quasylab/jspear/blob/working/examples/engine/>.

Simulation of evolution sequences. Given a distribution μ and $N, k \in \mathbb{N}$, we use function SIM (Algorithm 1) to obtain an *empirical evolution sequence* of the form E_0, \dots, E_k of size N and length k , starting from μ . Each E_i is a tuple $\mathbf{d}_i^1, \dots, \mathbf{d}_i^N$ of data states that are used to estimate the probability distribution reached at step i . E_0 is a sample of size N of μ obtained by means of a function SAMPLEDISTR. Then, E_{i+1} is obtained by simulating one computation step from each element in E_i via a function SIMSTEP that mimics the behaviour of step (Section 2): for any \mathbf{d} and $\mathbb{D} \in \mathcal{B}_{\mathcal{D}}$ it holds that $\Pr\{\text{SIMSTEP}(\mathbf{d}) \in \mathbb{D}\} = \text{step}(\mathbf{d})(\mathbb{D})$. For any $i \in [0, k]$, we let $\hat{\mathcal{S}}_{E_0}^{i,N}$ be the distribution such that $\hat{\mathcal{S}}_{E_0}^{i,N}(\mathbb{D}) = |E_i \cap \mathbb{D}|/N$ for any $\mathbb{D} \in \mathcal{B}_{\mathcal{D}}$. By applying the weak law of large numbers to the i.i.d. samples, we get $\lim_{N \rightarrow \infty} \hat{\mathcal{S}}_{E_0}^{i,N} = \mathcal{S}_{\mu}^i$. Henceforth, we identify $\hat{\mathcal{S}}_{E_0}^N$ with the estimated evolution sequence of size N , $\bar{E} = E_0, \dots, E_k$.

Applying perturbations to evolution sequences. We use function SIMPER (Algorithm 2) to simulate the effect of a perturbation \mathbf{p} on an estimated evolution sequence \bar{E} of size N . This function takes two integers as parameters: τ and ℓ . τ is the time step at which \mathbf{p} is applied. ℓ is the number of additional samples that we generate to evaluate the effect of \mathbf{p} on each data state, to guarantee statistical relevance of the collected data. Given E , we denote by $\ell \cdot E$ the sample set obtained from E by replicating each of its elements ℓ times. Function SIMPER is similar to SIM. They differ in constructing the tuple E_{i+1} , as in SIMPER we need first to sample the effect of \mathbf{p} . This is done by function SAMPLE($f(\mathbf{d})$) whose definition is standard and therefore omitted. According to Section 3.2, function f in SAMPLE($f(\mathbf{d})$) is effect(\mathbf{p}), and the perturbation used at next time step is next(\mathbf{p}).

Evaluation of distance expressions. Distance expressions are estimated following a syntax driven algorithm (Algorithm 4). To deal with atomic expressions \langle^{ρ} and \rangle^{ρ} , we rely on an existing approach [46] to estimate the Wasserstein distance between two distributions $\mu, \nu \in \Pi(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$. We consider N independent samples $\{\mathbf{d}_1^1, \dots, \mathbf{d}_1^N\}$ taken from μ , and ℓN independent samples $\{\mathbf{d}_2^1, \dots, \mathbf{d}_2^{\ell N}\}$ taken from ν , for some integers N, ℓ . Then, we exploit the penalty function ρ_i to map each sampled data state onto \mathbb{R} , so that, to capture the minimisation over the couplings, it is enough to consider the reordered sequences of values $\{\omega_j = \rho_i(\mathbf{d}_1^j) \mid \omega_j \leq \omega_{j+1}\}$ and $\{\nu_h = \rho_i(\mathbf{d}_2^h) \mid \nu_h \leq \nu_{h+1}\}$. Then $\mathbf{W}(m_{\rho,i})(\nu, \mu) = \frac{1}{\ell N} \sum_{h=1}^{\ell N} \max\{\nu_h - \omega_{\lceil \frac{h}{M} \rceil}, 0\}$ [10, 11].

Function WASS (Algorithm 3) implements the procedure outlined above. Parameter $op \in \{<, >\}$ of WASS allows us to choose which between \langle^{ρ} and \rangle^{ρ} we want to approximate: if op is $<$, we approximate $\mathbf{W}(m_{\rho,i})(\mu, \nu)$; if op is $>$, we approximate $\mathbf{W}(m_{\rho,i})(\nu, \mu)$. Since penalty functions allow us to evaluate \mathbf{W} on \mathbb{R} , rather than on \mathbb{R}^n , the complexity of the outlined procedure is $\mathcal{O}(\ell N \log(\ell N))$ [46], due to the sorting of $\{\nu_h \mid h \in [1, \dots, \ell N]\}$. We refer to [44, Corollary 3.5, Equation (3.10)] for an estimation of the approximation error on the evaluation of the Wasserstein distance over $N, \ell N$ samples.

Using the estimation of the atomic expressions as base case, we define function EVALEXP (Algorithm 4) recursively on the syntax of **exp**, following Definition 8.

Checking formulae satisfaction. Function SAT (Algorithm 5) allows us to verify whether a given evolution sequence satisfies a given RobTL formula at a given time step. It takes five parameters: the initial distribution μ , the time step τ , the formula φ , the two integers N and ℓ identifying the number of samples. Function SAT consists of three steps. Firstly, we compute, using structural induction, the *time horizon* k of φ to identify the number of steps needed to evaluate it. Then, function SIM is used to simulate the evolution sequence from μ . Finally, φ is evaluated over \bar{E} and τ by calling function EVAL (Algorithm 6), that yields the Boolean evaluation of φ computed recursively on its structure following Definition 14.

(a) Difference with respect to $temp$.(b) Difference with respect to $stress$.

■ **Figure 2** Differences with respect to the values of $temp$ and $stress$, under $\mathbf{p}_{temp,o}$ for $o \in \{-2, -1.5, -1\}$.

(a) \exp_1 .(b) \exp_2 .

■ **Figure 3** Evaluation of \exp_1 and \exp_2 over the time interval $[0, 50]$.

► **Example 17.** Consider the attack on the sensor $temp$, modelled by perturbation $\mathbf{p} = \mathbf{p}_{temp,o}$ (Example 11), and let 0 be the current step. To give an idea of the impact of \mathbf{p} on the behaviour \mathcal{S} of the engine, in Figure 2a we report the pointwise evaluation of the distance $\langle \rho \rangle$, where $\rho(\mathbf{d}) = \mathbf{d}(temp)/150$ for all $\mathbf{d} \in \mathcal{D}$, over the time window $[90, 300]$, between the temperature in \mathcal{S} and that in three perturbations of it, obtained by three variations of \mathbf{p} with $o \in \{-2, -1.5, -1\}$. In all cases, the difference is greater in $[100, 200]$, i.e., while $\mathbf{f}_{temp,o}$ is active, and the smaller differences detected after 200 steps are due to the delays induced by the perturbations in the regular behaviour. Clearly, the larger the offset interval, the greater the difference. This is even more evident in Figure 2b, depicting the pointwise evaluations of the distances $\langle \rho^s \rangle$ between \mathcal{S} and its three perturbed versions, for the penalty function ρ^s defined as $\rho^s(\mathbf{d}) = \mathbf{d}(stress)$ in Example 3.

Let us now fix $o = -1.5$. Consider expressions $\exp_1 = \mathbf{G}^J \langle \rho^w \rangle$ and $\exp_2 = \mathbf{G}^J \langle \rho^s \rangle$, where $J = [100, 210]$ and both penalty functions ρ^w and ρ^s are defined in Example 3. In Figure 3 we report the variation of the evaluation of the two expressions over \mathcal{S} and its 51 perturbations via \mathbf{p} , each obtained by applying \mathbf{p} at a different $\tau' \in [0, 50]$. We associate the coordinate $x = \tau'$ with $\llbracket \exp_1 \rrbracket_{\mathcal{S}, \mathcal{S}_1 \langle \langle \mathbf{p} \rangle \rangle, \tau'}$ in Figure 3a, and with $\llbracket \exp_2 \rrbracket_{\mathcal{S}, \mathcal{S}_1 \langle \langle \mathbf{p} \rangle \rangle, \tau'}$ in Figure 3b. The two plots show that by applying \mathbf{p} at different time steps, we get different effects on system behaviour, with variations of the order of 10^{-3} . We run several experiments to infer for which stealthiness threshold η_3 and danger threshold η_4 , the formula φ_1 in Example 15 is satisfied. We concluded that for $\eta_3 \geq 0.06$ and $\eta_4 \leq 0.45$ the attack is successful (Example 20).

4.1 Complexity

We can assume that the evaluation of $\text{SIMSTEP}(\mathbf{d})$ needs a number of steps that is linear with the number of variables in \mathbf{d} . The same applies for the application of a perturbation or penalty function to a data state \mathbf{d} . Under these assumptions it is not hard to see that

to evaluate $\text{SIM}(\mu, N, k)$ we need $\mathcal{O}(kN \cdot |\text{Var}|)$ steps, while $\mathcal{O}(k\ell N \cdot |\text{Var}|)$ steps are needed to evaluate $\text{SIMPER}(\{E_1, \dots, E_k\}, \mathbf{p}, \tau, \ell)$. Moreover, $\mathcal{O}(|E_2| \log |E_2|)$ steps are needed for $\text{WASS}(E_1, E_2, \text{op}, \rho)$. This is dominated by the number of steps needed to order the sequences ω_i and ν_h (lines 4 and 5). For the sake of simplicity, the algorithms in Algorithm 4 and Algorithm 6 are presented following a *forward* approach where to compute the value at time i , all the values in an interval $[i + a, i + b]$ could be considered. This means that to compute $\text{EVAL}(\overline{E}, \tau, \varphi, \ell)$ (resp. $\text{EVAL}(\overline{E}, \overline{E}', \tau, \text{exp})$), we need a number of steps that, in the worst case, are linear with the length of \overline{E} and exponential with the size of φ (resp. exp). However, if a *backward* approach is used as in [17], the same functions can be computed with a number of steps that is linear with both the length of \overline{E} and the size of φ (resp. exp).

4.2 Statistical error

We provide an algorithm for the evaluation of a *confidence interval* CI on the estimation of the value of a distance expression exp . This means that, given exp , a nominal evolution sequence \mathcal{S} , a perturbation \mathbf{p} , two time steps τ and τ' , and a coverage probability α , the probability that the real value $\llbracket \text{exp} \rrbracket_{\mathcal{S}, \mathcal{S}_{1 \llcorner (\mathbf{p})}, \tau}^{\tau'}$ of the distance is in CI is at least α .

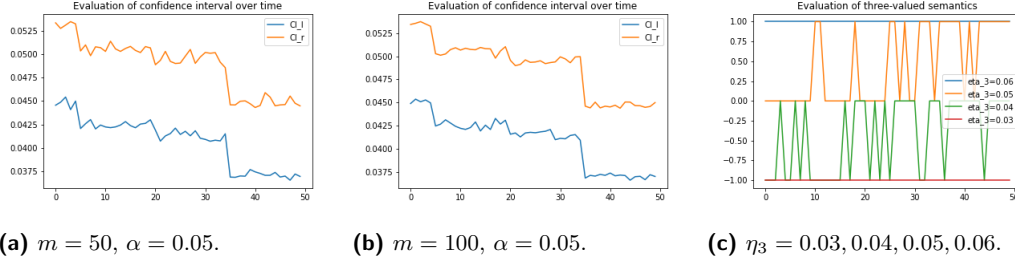
We start by evaluating the confidence intervals on $\mathbf{W}(\mathbf{m}_\tau^\ell)(\mu, \nu)$, obtained by applying the *empirical bootstrap* method [19, 20]:

1. Generate m bootstrap samples for μ and ν : these are obtained by drawing with replacement a sample of size N from the elements of the original sampling of μ , and one of size ℓN from those for ν . Let μ_1, \dots, μ_m and ν_1, \dots, ν_m the resulting bootstrap distributions.
2. Apply the procedure WASS m -times to evaluate the Wasserstein distances between the bootstrap distributions. Let W_1, \dots, W_m be the resulting bootstrap distances.
3. Evaluate the mean of the bootstrap distances $\overline{W} = \sum_{i=1}^m W_i / m$.
4. Evaluate the standard error $SE = (\sum_{i=1}^m (W_i - \overline{W})^2 / (m - 1))^{1/2}$.
5. Let $\text{CI} = \overline{W} \pm z_{1-\frac{\alpha}{2}} SE$, with $z_{1-\frac{\alpha}{2}}$ the $1 - \frac{\alpha}{2}$ quantile of the standard normal distribution.

► **Remark 18.** In [45] the *bias-corrected, accelerated percentile intervals* (BC_α) is used. We chose to use the empirical bootstrap method to find a balance between accuracy and computational complexity. Empirical bootstrap can be subject to bias in the samples, and more accurate techniques, like BC_α , were proposed [16]. However, to reach the desired accuracy with the BC_α method, it is necessary to use $m \geq \mathcal{O}(1000)$ bootstrap samples. Given the cost $\mathcal{O}(\ell N \log(\ell N))$ of a single evaluation of \mathbf{W} , and considering that in our formulae this distance is evaluated thousands of times, this approach would be computationally unfeasible. In our examples, $m \leq 100$ is sufficient to obtain reasonable confidence intervals (Example 19).

The evaluation of the confidence interval for the Wasserstein distance is then extended to distance expressions: once we have determined the bounds of the confidence intervals of the sub-expressions occurring in exp , the interval of exp is obtained by applying the function defining the evaluation of exp to them. For instance, if $\text{exp} = \max(\text{exp}_1, \text{exp}_2)$, $\text{CI}_{\text{exp}_1} = (l_1, r_1)$, and $\text{CI}_{\text{exp}_2} = (l_2, r_2)$, then $\text{CI}_{\text{exp}} = (\max\{l_1, l_2\}, \max\{r_1, r_2\})$.

► **Example 19.** In Figure 4 we report the 95% confidence intervals for $\llbracket \text{exp}_1 \rrbracket_{\mathcal{S}, \mathcal{S}_{1 \llcorner (\mathbf{p})}, 0}^{\tau'}$, where $\tau' \in [0, 50]$, with exp_1 and \mathbf{p} as in Example 17. The intervals in Figure 4a have been obtained by means of $m = 50$ bootstrap samplings, whereas for those in Figure 4b we used $m = 100$. In the former case, the maximal width of the interval is $9.39 \cdot 10^{-3}$, with an average width of $8.07 \cdot 10^{-3}$; in the latter case, those number become, respectively, $9.03 \cdot 10^{-3}$ and $7.93 \cdot 10^{-3}$.



■ **Figure 4** Confidence intervals of exp_1 , and three-valued evaluation of φ_{η_3} , over $[0, 50]$.

A three-valued semantics for RobTL. Given the presence of errors in the evaluation of expressions we extend our model checking algorithm with the possibility to assign a three-valued semantics to formulae. Alongside the classical Boolean evaluations true (\top) and false (\perp), a RobTL formula can assume the value *unknown* (Ψ). Intuitively, unknown is generated by the comparison between the distance and the chosen threshold in atomic propositions: if the threshold η does not lie in the confidence interval of the evaluation of the distance, then the formula will evaluate to \top or \perp according to the validity of the relation $\bowtie \eta$. Conversely, if η belongs to the confidence interval, then the atomic proposition evaluates to Ψ , since the validity of the relation $\bowtie \eta$ may depend on the particular samples obtained in the simulation.

Starting from atomic propositions, the three-valued semantics is extended to the Boolean operators via truth tables in the standard way [29, 48]. Then, we assign a three-valued semantics to RobTL formulae via the satisfaction function $\Omega_S: L \times [0, \mathfrak{h}] \rightarrow \{\top, \Psi, \perp\}$, defined inductively on the structure of RobTL formulae, starting from atomic propositions as follows:

$$\begin{aligned} \Omega_S(\top, \tau) &= \top \\ \Omega_S(\Delta(\text{exp}, \mathbf{p}) \bowtie \eta, \tau) &= \begin{cases} \Psi & \text{if } \eta \in \text{CI}_{\text{exp}} \\ \mathcal{S}, \tau \models \Delta(\text{exp}, \mathbf{p}) \bowtie \eta & \text{otherwise.} \end{cases} \\ \Omega_S(\neg \varphi, \tau) &= \neg \Omega_S(\varphi, \tau) \\ \Omega_S(\varphi_1 \wedge \varphi_2, \tau) &= \Omega_S(\varphi_1, \tau) \wedge \Omega_S(\varphi_2, \tau) \\ \Omega_S(\varphi_1 \mathcal{U}^I \varphi_2, \tau) &= \bigvee_{\tau' \in I} \left(\Omega_S(\varphi_2, \tau') \wedge \bigwedge_{\tau'' \in I, \tau'' < \tau'} \Omega_S(\varphi_1, \tau'') \right). \end{aligned}$$

The algorithm for the evaluation of function Ω_S is obtained in a straightforward manner from the Boolean evaluation (Algorithm 6).

► **Example 20.** Consider the formula $\varphi_{\eta_3} = \Delta(\text{exp}_1, \mathbf{p}) \leq \eta_3$ for exp_1 and \mathbf{p} as in Example 17. In Figure 4c we report the variation of the evaluation of $\Omega_S(\varphi_{\eta_3}, \tau')$ with respect to $\tau' \in [0, 50]$ and $\eta_3 \in \{0.03, 0.04, 0.05, 0.06\}$, where we let $\top \mapsto 1$, $\Psi \mapsto 0$, and $\perp \mapsto -1$. The plot confirms the validity of the empirical tuning of parameter η_3 that we carried out in Example 17.

5 Concluding remarks

The term robustness is used in several contexts, from control theory [50] to biology [28], and not always with the same meaning. Since our objective was to provide a formal tool for the verification of general robustness properties, we limit ourselves to recall that, in the context of CPS, we can distinguish five categories of robustness [24]:

- (i) input/output robustness;
- (ii) robustness with respect to system parameters;
- (iii) robustness in real-time system implementation;
- (iv) robustness due to unpredictable environment;
- (v) robustness to faults.

Our framework is designed for properties of type (iv), and we plan to extend it to the others.

[49] presents a PCTL statistical model checker based on stratified sampling. This allows for the generation of negatively correlated samples, thus considerably reducing the number of samples needed to obtain confident verdicts, provided the PCTL formulae are of a particular form. While direct comparison of the two algorithms would not be meaningful given the disparity in the logics, we will study the use of stratified sampling in our model checker.

In [1] the model of discrete time stochastic hybrid systems is introduced and used to formalise finite-horizon probabilistic reachability problems. Specifically, maximal probabilities of reaching (and maintaining) a safe set of states are considered. There are two main differences in between this model and the evolution sequence model that we would like to highlight:

- The purely data-driven characterisation of systems behaviour of the evolution sequence model has two crucial consequences. The first consequence is that the specification of the behaviour of the agent and that of the environment are independent, and there is no need to specify a system's state space, as opposed to what happens with the model in the proposed paper. The second consequence is that the behaviour of the system is not given by a set of traces/trajectories, but by the combination of their effects.
- The paper [1] only presents the analysis of probabilistic reachability properties, based on the evaluation of the desired safety property on each single trace of the system and the consequent computation of the total probability of executing those traces. In this paper, we introduce a logic that allows us to specify robustness properties based on the evaluation of distances between the behaviour of two different systems, the nominal and the perturbed one.

We plan to apply our framework to the analysis of biological systems. Some quantitative temporal logics have already been proposed in that setting [21,36,37] to capture some notions of robustness in system biology [5,28,32,41,42]. We are confident that the use of RobTL and evolution sequences can lead to new results, as shown in the preliminary work [13]. Moreover, we will apply our work to Medical CPS. In this context, statistical inference and learning methods have been combined in the synthesis of controllers, in order to deal with uncertainties [33]. The idea is then to use our tool to test the obtained controllers and verify their robustness against uncertainties.

Finally, we plan to apply our work to the evaluation of the effectiveness of digital twins [26]. To this end, we will enrich STARK with a special construct, similar to perturbations, that will allow us to model the communications, and their effects, between the digital and the real-world (perturbed) twin in a concise, clean, fashion. A preliminary result in this direction can be found in [8].

References

- 1 Alessandro Abate, Maria Prandini, John Lygeros, and Shankar Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Autom.*, 44(11):2724–2734, 2008. doi:10.1016/J.AUTOMATICA.2008.03.027.
- 2 Erika Ábrahám and Borzoo Bonakdarpour. HyperPCTL: A temporal logic for probabilistic hyperproperties. In *Proceedings of QEST 2018*, volume 11024 of *Lecture Notes in Computer Science*, pages 20–35. Springer, 2018. doi:10.1007/978-3-319-99154-2_2.

- 3 Shiraj Arora, René Rydhof Hansen, Kim Guldstrand Larsen, Axel Legay, and Danny Bøgsted Poulsen. Statistical model checking for probabilistic hyperproperties of real-valued signals. In *Proceedings of SPIN 2022*, volume 13255 of *Lecture Notes in Computer Science*, pages 61–78. Springer, 2022. doi:10.1007/978-3-031-15077-7_4.
- 4 Christel Baier. Probabilistic model checking. In Javier Esparza, Orna Grumberg, and Salomon Sickert, editors, *Dependable Software Systems Engineering*, volume 45 of *NATO Science for Peace and Security Series - D: Information and Communication Security*, pages 1–23. IOS Press, 2016. doi:10.3233/978-1-61499-627-9-1.
- 5 Naama Barkai and Stanislas Leibler. Robustness in simple biochemical networks. *Nature*, 387:913–917, 1997. doi:10.1038/43199.
- 6 Vladimir I. Bogachev. *Measure Theory, vol. 2.*. Measure Theory. Springer-Verlag, Berlin/Heidelberg, 2007. doi:10.1007/978-3-540-34514-5.
- 7 Christos G. Cassandras, John Lygeros, and (eds.). *Stochastic Hybrid Systems*. Number 24 in Control Engineering. CRC Press, Boca Raton, 1st edition, 2007. doi:10.1201/9781315221625.
- 8 Valentina Castiglioni, Ruggero Lanotte, Michele Loreti, and Simone Tini. Evaluating the effectiveness of digital twins through statistical model checking with feedback and perturbations. In *Proceedings of FMICS 2024*, Lecture Notes in Computer Science. Springer, 2024.
- 9 Valentina Castiglioni, Michele Loreti, and Simone Tini. STARK: A Software Tool for the Analysis of Robusness in the unKnown environment. Software, swbId: swb:1:dir:ddfb418d5a080b8e83323a1b2c38d9f7065e2554 (visited on 2024-08-21). URL: <https://github.com/quasylab/jspear/tree/working>.
- 10 Valentina Castiglioni, Michele Loreti, and Simone Tini. How adaptive and reliable is your program? In *Proceedings of FORTE 2021*, volume 12719 of *LNCS*, pages 60–79. Springer, 2021. doi:10.1007/978-3-030-78089-0_4.
- 11 Valentina Castiglioni, Michele Loreti, and Simone Tini. A framework to measure the robustness of programs in the unpredictable environment. *Log. Methods Comput. Sci.*, 19(3), 2023. doi:10.46298/LMCS-19(3:2)2023.
- 12 Valentina Castiglioni, Michele Loreti, and Simone Tini. STARK: A Software Tool for the Analysis of Robustness in the unKnown environment. In *Proceedings of COORDINATION 2023*, volume 13908 of *Lecture Notes in Computer Science*, pages 115–132. Springer, 2023. doi:10.1007/978-3-031-35361-1_6.
- 13 Valentina Castiglioni, Michele Loreti, and Simone Tini. Bio-STARK: A tool for the time-point robustness analysis of biological systems. In *Proceedings of CMSB 2024*, Lecture Notes in Computer Science. Springer, 2024. To appear.
- 14 Valentina Castiglioni, Michele Loreti, and Simone Tini. STARK: A tool for the analysis of CPSs robustness. *Science of Computer Programming*, 236:103134, 2024. doi:10.1016/j.scico.2024.103134.
- 15 Michael R. Clarkson and Fred B. Schneider. Hyperproperties. *J. Comput. Secur.*, 18(6):1157–1210, 2010. doi:10.3233/JCS-2009-0393.
- 16 Thomas J. DiCiccio and Bradley Efron. Bootstrap confidence intervals. *Statistical Science*, 11(3):189–228, 1996. doi:10.1214/ss/1032280214.
- 17 Alexandre Donzé, Thomas Ferrère, and Oded Maler. Efficient robust monitoring for STL. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification*, pages 264–279, Berlin, Heidelberg, 2013. Springer.
- 18 Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *Proceedings of FORMATS 2010*, volume 6246 of *LNCS*, pages 92–106. Springer, 2010. doi:10.1007/978-3-642-15297-9_9.
- 19 Bradley Efron. Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. doi:10.1214/aos/1176344552.
- 20 Bradley Efron. Nonparametric standard errors and confidence intervals. *Canadian Journal of Statistics*, 9(2):139–158, 1981. doi:10.2307/3314608.

- 21 François Fages and Aurélien Rizk. On temporal logic constraint solving for analyzing numerical data time series. *Theor. Comput. Sci.*, 408(1):55–65, 2008. doi:10.1016/j.tcs.2008.07.004.
- 22 Georgios E. Fainekos and George J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theor. Comput. Sci.*, 410(42):4262–4291, 2009. doi:10.1016/j.tcs.2009.06.021.
- 23 Olivier P. Faugeras and Ludeger Rüschemdorf. Risk excess measures induced by hemi-metrics. *Probability, Uncertainty and Quantitative Risk*, 3:6, 2018. doi:10.1186/s41546-018-0032-0.
- 24 Martin Fränzle, James Kapinski, and Pavithra Prabhakar. Robustness in cyber-physical systems. *Dagstuhl Reports*, 6(9):29–45, 2016.
- 25 Dieter Gollmann, Pavel Gurikov, Alexander Isakov, Marina Krotofil, Jason Larsen, and Alexander Winnicki. Cyber-Physical Systems Security: Experimental Analysis of a Vinyl Acetate Monomer Plant. In *Proceedings of CPSS 2015*, pages 1–12. ACM, 2015.
- 26 Michael Grieves and John Vickers. *Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems*, pages 85–113. Springer, 2017. doi:10.1007/978-3-319-38756-7_4.
- 27 Jianghai Hu, John Lygeros, and Shankar Sastry. Towards a theory of stochastic hybrid systems. In *Proceedings of HSCC 2000*, volume 1790 of *LNCS*, pages 160–173, 2000. doi:10.1007/3-540-46430-1_16.
- 28 Hiroaki Kitano. Towards a theory of biological robustness. *Molecular Systems Biology*, 3(1):137, 2007. doi:10.1038/msb4100179.
- 29 Stephen Cole Kleene. *Introduction to Metamathematics*. Princeton, NJ, USA: North Holland, 1952. doi:10.2307/2268620.
- 30 Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Stochastic model checking. In *Proceedings of SFM 2007*, volume 4486 of *LNCS*, pages 220–270. Springer, 2007. doi:10.1007/978-3-540-72522-0_6.
- 31 Ruggero Lanotte, Massimo Merro, Andrei Munteanu, and Simone Tini. Formal impact metrics for cyber-physical attacks. In *Proceedings of CSF 2021*, pages 1–16. IEEE, 2021. doi:10.1109/CSF51468.2021.00040.
- 32 Lucia Nasti, Roberta Gori, and Paolo Milazzo. Formalizing a notion of concentration robustness for biochemical networks. In *Proceedings of STAF 2018*, volume 11176 of *LNCS*, pages 81–97. Springer, 2018. doi:10.1007/978-3-030-04771-9_8.
- 33 Nicola Paoletti, Kin Sum Liu, Hongkai Chen, Scott A. Smolka, and Shan Lin. Data-driven robust control for a closed-loop artificial pancreas. *IEEE ACM Trans. Comput. Biol. Bioinform.*, 17(6):1981–1993, 2020. doi:10.1109/TCBB.2019.2912609.
- 34 Svetlozar T. Rachev, Lev B. Klebanov, Stoyan V. Stoyanov, and Frank J. Fabozzi. *The Methods of Distances in the Theory of Probability and Statistics*. Springer, 2013.
- 35 Ragnathan Rajkumar, Insup Lee, Lui Sha, and John A. Stankovic. Cyber-physical systems: the next computing revolution. In *Proceedings of DAC 2010*, pages 731–736. ACM, 2010. doi:10.1145/1837274.1837461.
- 36 Aurélien Rizk, Grégory Batt, François Fages, and Sylvain Soliman. A general computational method for robustness analysis with applications to synthetic gene networks. *Bioinform.*, 25(12), 2009. doi:10.1093/bioinformatics/btp200.
- 37 Aurélien Rizk, Grégory Batt, François Fages, and Sylvain Soliman. Continuous valuations of temporal logic specifications with applications to parameter optimization and robustness measures. *Theor. Comput. Sci.*, 412(26):2827–2839, 2011. doi:10.1016/j.tcs.2010.05.008.
- 38 Matthias Rungger and Paulo Tabuada. A notion of robustness for cyber-physical systems. *IEEE Trans. Autom. Control.*, 61(8):2108–2123, 2016.
- 39 Koushik Sen, Mahesh Viswanathan, and Gul Agha. On statistical model checking of stochastic systems. In *Proceedings of CAV 2005*, volume 3576 of *LNCS*, pages 266–280. Springer, 2005. doi:10.1007/11513988_26.
- 40 Ali Shahrokni and Robert Feldt. A systematic review of software robustness. *Information and Software Technology*, 55(1):1–17, 2013. doi:10.1016/j.infsof.2012.06.002.

- 41 Guy Shinar and Martin Feinberg. Structural sources of robustness in biochemical reaction networks. *Science*, 327(5971):1389–1391, 2010. doi:10.1126/science.1183372.
- 42 Guy Shinar and Martin Feinberg. Design principles for robust biochemical reaction networks: what works, what cannot work, and what might almost work. *Mathe. Biosci*, 231(1):39–48, 2011. doi:10.1016/j.mbs.
- 43 Eduardo D. Sontag. *Input to State Stability: Basic Concepts and Results*, pages 163–220. Springer, 2008. doi:10.1007/978-3-540-77653-6_3.
- 44 Bharath K. Sriperumbudur, Kenji Fukumizu, Arthur Gretton, Bernard Schölkopf, and Gert R. G. Lanckriet. On the empirical estimation of integral probability metrics. *Electronic Journal of Statistics*, 6:1550–1599, 2021. doi:10.1214/12-EJS722.
- 45 David Thorsley and Eric Klavins. Model reduction of stochastic processes using Wasserstein pseudometrics. In *2008 American Control Conference*, pages 1374–1381. IEEE, 2008. doi:10.1109/ACC.2008.4586684.
- 46 David Thorsley and Eric Klavins. Approximating stochastic biochemical processes with Wasserstein pseudometrics. *IET Syst. Biol.*, 4(3):193–211, 2010. doi:10.1049/iet-syb.2009.0039.
- 47 Leonid N. Vaserstein. Markovian processes on countable space product describing large systems of automata. *Probl. Peredachi Inf.*, 5(3):64–72, 1969.
- 48 Ludovica Luisa Vissat, Michele Loreti, Laura Nenzi, Jane Hillston, and Glenn Marion. Analysis of spatio-temporal properties of stochastic systems using TSTL. *ACM Trans. Model. Comput. Simul.*, 29(4):20:1–20:24, 2019. doi:10.1145/3326168.
- 49 Yu Wang, Nima Roohi, Matthew West, Mahesh Viswanathan, and Geir E. Dullerud. Statistical verification of PCTL using antithetic and stratified samples. *Formal Methods Syst. Des.*, 54(2):145–163, 2019. doi:10.1007/s10703-019-00339-8.
- 50 Kemin Zhou and John C. Doyle. *Essentials of Robust Control*. Prentice-Hall, 1997.

A The Wasserstein hemimetric

Given a (pseudo-, hemi-)metric space (Ω, m) , the (pseudo-, hemi-)metric m induces a natural topology over Ω , namely the topology generated by the open ε -balls, for $\varepsilon > 0$, $B_m(\omega, \varepsilon) = \{\omega' \in \Omega \mid m(\omega, \omega') < \varepsilon\}$. We can then naturally obtain the Borel σ -algebra over Ω from this topology.

In this paper we are interested in defining a *hemimetric on distributions*. To this end we will make use of the Wasserstein lifting [47] whose definition is based on the following notions and results. Given a set Ω and a topology T on Ω , the topological space (Ω, T) is said to be *completely metrisable* if there exists at least one metric m on Ω such that (Ω, m) is a complete metric space and m induces the topology T . A *Polish space* is a separable completely metrisable topological space. In particular, we recall that:

- (i) \mathbb{R} is a Polish space; and
- (ii) every closed subset of a Polish space is in turn a Polish space.

Moreover, for any $n \in \mathbb{N}$, if $\Omega_1, \dots, \Omega_n$ are Polish spaces, then the Borel σ -algebra on their product coincides with the product σ -algebra generated by their Borel σ -algebras, namely

$$\mathcal{B}\left(\prod_{i=1}^n \Omega_i\right) = \prod_{i=1}^n \mathcal{B}(\Omega_i).$$

(This is proved, e.g., in [6] as Lemma 6.4.2 whose hypothesis are satisfied by Polish spaces since they are second countable.) These properties of Polish spaces are interesting for us since they guarantee that all the distributions we consider in this paper are Radon measures

and, thus, the Wasserstein lifting is well-defined on them. For this reason, we also directly present the Wasserstein hemimetric by considering only distributions on Borel sets.

► **Definition 21** (Wasserstein hemimetric). *Consider a Polish space Ω and let m be a hemimetric on Ω . For any two distributions μ and ν on $(\Omega, \mathcal{B}(\Omega))$, the Wasserstein lifting of m to a distance between μ and ν is defined by*

$$\mathbf{W}(m)(\mu, \nu) = \inf_{\mathfrak{w} \in \mathfrak{W}(\mu, \nu)} \int_{\Omega \times \Omega} m(\omega, \omega') d\mathfrak{w}(\omega, \omega')$$

where $\mathfrak{W}(\mu, \nu)$ is the set of the couplings of μ and ν , namely the set of joint distributions \mathfrak{w} over the product space $(\Omega \times \Omega, \mathcal{B}(\Omega \times \Omega))$ having μ and ν as left and right marginal, respectively, namely $\mathfrak{w}(\mathbb{A} \times \Omega) = \mu(\mathbb{A})$ and $\mathfrak{w}(\Omega \times \mathbb{A}) = \nu(\mathbb{A})$, for all $\mathbb{A} \in \mathcal{B}(\Omega)$.

Despite the original version of the Wasserstein distance being defined on a metric on Ω , the Wasserstein hemimetric given above is well-defined. We refer the interested reader to [23] and the references therein for a formal proof of this fact. In particular, the Wasserstein hemimetric is given in [23] as Definition 7 (considering the compound risk excess metric defined in Equation (31) of that paper), and Proposition 4 in [23] guarantees that it is indeed a well-defined hemimetric on $\Pi(\Omega, \mathcal{B}(\Omega))$. Moreover, Proposition 6 in [23] guarantees that the same result holds for the hemimetric $m(x, y) = \max\{y - x, 0\}$.

B The algorithms

In this section we report the algorithms described in Section 4.

■ **Algorithm 1** Simulation of a evolution sequence.

```

1: function SIM( $\mu, N, k$ )
2:    $i \leftarrow 0$ 
3:    $E_0 \leftarrow \text{SAMPLEDISTR}(\mu, N)$ 
4:   while  $i < k$  do
5:      $E_{i+1} \leftarrow \emptyset$ 
6:     for  $\mathbf{d} \in E_i$  do
7:        $E_{i+1} \leftarrow \text{SIMSTEP}(\mathbf{d}), E_{i+1}$ 
8:     end for
9:      $i \leftarrow i + 1$ 
10:  end while
11:  return  $E_0, \dots, E_k$ 
12: end function

```

■ **Algorithm 2** Computation of the effect of a perturbation.

```

1: function SIMPER( $\overline{E}, p, \tau, \ell$ )
2:    $\forall i < \tau. E'_i \leftarrow E_i$ 
3:    $E'_\tau \leftarrow \ell \cdot E_\tau$ 
4:    $i \leftarrow \tau$ 
5:   while  $i < k$  do
6:      $f \leftarrow \text{effect}(p)$ 
7:      $p \leftarrow \text{next}(p)$ 
8:     for  $d \in E'_i$  do
9:        $d' \leftarrow \text{SAMPLE}(f(d))$ 
10:       $E'_{i+1} \leftarrow E'_{i+1}, \text{SIMSTEP}(d')$ 
11:    end for
12:     $i \leftarrow i + 1$ 
13:  end while
14:  return  $E'_0, \dots, E'_k$ 
15: end function

```

■ **Algorithm 3** Evaluation of the Wasserstein distance.

```

1: function WASS( $E_1, E_2, op, \rho$ )
2:    $(d_1^1, \dots, d_1^N) \leftarrow E_1$ 
3:    $(d_2^1, \dots, d_2^{\ell N}) \leftarrow E_2$ 
4:    $\forall j : (1 \leq j \leq N) : \omega_j \leftarrow \rho(d_1^j)$ 
5:    $\forall h : (1 \leq h \leq \ell N) : \nu_h \leftarrow \rho(d_2^h)$ 
6:   re index  $\{\omega_j\}$  s.t.  $\omega_j \leq \omega_{j+1}$ 
7:   re index  $\{\nu_h\}$  s.t.  $\nu_h \leq \nu_{h+1}$ 
8:   if  $op = <$  then
9:     return  $\frac{1}{\ell N} \sum_{h=1}^{\ell N} \max\{\nu_h - \omega_{\lceil \frac{h}{\ell} \rceil}, 0\}$ 
10:  else
11:    return  $\frac{1}{\ell N} \sum_{h=1}^{\ell N} \max\{\omega_{\lceil \frac{h}{\ell} \rceil} - \nu_h, 0\}$ 
12:  end if
13: end function

```

■ **Algorithm 4** Evaluation of distance expressions.

```

1: function EVAL Expr( $\overline{E}, \overline{E}', \tau, \text{exp}$ )
2:   match exp
3:     with  $<^\rho$  :
4:       return WASS( $E_\tau, E'_\tau, <, \rho$ )
5:     with  $>^\rho$  :
6:       return WASS( $E_\tau, E'_\tau, >, \rho$ )
7:     with  $F^I$  exp :
8:       return  $\min_{i \in \tau+I} \{\text{EVAL Expr}(\overline{E}, \overline{E}', i, \text{exp})\}$ 
9:     with  $G^I$  exp :
10:      return  $\max_{i \in \tau+I} \{\text{EVAL Expr}(\overline{E}, \overline{E}', i, \text{exp})\}$ 
11:     with  $\text{exp}_1 \cup^{[\tau_1, \tau_2]} \text{exp}_2$  :
12:        $\forall i \in [\tau + \tau_1, \tau + \tau_2] d_i^2 \leftarrow \text{EVAL Expr}(\overline{E}, \overline{E}', i, \text{exp}_2)$ 
13:        $\forall j \in [\tau + \tau_1, \tau + \tau_2] d_j^1 \leftarrow \text{EVAL Expr}(\overline{E}, \overline{E}', j, \text{exp}_1)$ 
14:       return  $\min_{\tau+\tau_1 \leq i \leq \tau+\tau_2} \{\max\{d_i^2, \max_{0 \leq j < i} \{d_j^1\}\}\}$ 
15:     with  $\min(\text{exp}_1, \text{exp}_2)$  :
16:        $v_1 \leftarrow \text{EVAL Expr}(\overline{E}, \overline{E}', \tau, \text{exp}_1)$ 
17:        $v_2 \leftarrow \text{EVAL Expr}(\overline{E}, \overline{E}', \tau, \text{exp}_2)$ 
18:       return  $\min\{v_1, v_2\}$ 
19:     with  $\max(\text{exp}_1, \text{exp}_2)$  :
20:        $v_1 \leftarrow \text{EVAL Expr}(\overline{E}, \overline{E}', \tau, \text{exp}_1)$ 
21:        $v_2 \leftarrow \text{EVAL Expr}(\overline{E}, \overline{E}', \tau, \text{exp}_2)$ 
22:       return  $\max\{v_1, v_2\}$ 
23:     with  $\sum_{i \in K} w_i \cdot \text{exp}_i$  :
24:        $v_i \leftarrow \text{EVAL Expr}(\overline{E}, \overline{E}', \tau, \text{exp}_i)$ 
25:       return  $\sum_{i \in K} w_i \cdot v_i$ 
26:     with  $\sigma(\text{exp}, \bowtie \zeta)$  :
27:        $v \leftarrow \text{EVAL Expr}(\overline{E}, \overline{E}', \tau, \text{exp})$ 
28:       if  $v \bowtie \zeta$  then
29:         return 0
30:       else
31:         return 1
32:       end if
33: end function

```

■ **Algorithm 5** Checking the satisfaction of a formula.

```

1: function SAT( $\mu, \tau, \varphi, N, \ell$ )
2:    $k \leftarrow \text{HORIZON}(\varphi)$ 
3:    $\overline{E} \leftarrow \text{SIM}(\mu, N, k)$ 
4:   return EVAL( $\overline{E}, \tau, \varphi, \ell$ )
5: end function

```

■ **Algorithm 6** Evaluation of RobTL formulae.

```

1: function EVAL( $\overline{E}, \tau, \varphi, \ell$ )
2:   match  $\varphi$ 
3:     with  $\varphi = \top$  :
4:       return true
5:     with  $\varphi = \Delta(\mathbf{exp}, \mathbf{p}) \bowtie \eta$  :
6:        $\overline{E}' \leftarrow \text{SIMPER}(\overline{E}, \mathbf{p}, \tau, \ell)$ 
7:        $v \leftarrow \text{EVALEXPR}(\overline{E}, \overline{E}', \tau, \mathbf{exp})$ 
8:       return  $v \bowtie \eta$ 
9:     with  $\varphi = \neg\varphi_1$  :
10:      return  $\neg\text{EVAL}(\overline{E}, \tau, \varphi_1, \ell)$ 
11:    with  $\varphi_1 \wedge \varphi_2$  :
12:      return  $\text{EVAL}(\overline{E}, \tau, \varphi_1, \ell) \wedge \text{EVAL}(\overline{E}, \tau, \varphi_2, \ell)$ 
13:    with  $\varphi_1 \mathcal{U}^{[\tau_1, \tau_2]} \varphi_2$  :
14:       $j \leftarrow \tau + \tau_1$ 
15:       $i \leftarrow j - 1$ 
16:       $\text{res} \leftarrow \textit{false}$ 
17:       $\text{res}' \leftarrow \textit{true}$ 
18:      while  $j \leq \tau + \tau_2 \wedge \neg\text{res} \wedge \text{res}'$  do
19:         $\text{res} \leftarrow \text{EVAL}(\overline{E}, j, \varphi_2, \ell)$ 
20:         $i \leftarrow i + 1$ 
21:         $\text{res}' \leftarrow \text{EVAL}(\overline{E}, i, \varphi_1, \ell)$ 
22:         $j \leftarrow j + 1$ 
23:      end while
24:      return  $\text{res}$ 
25: end function

```
