DisTL: A Temporal Logic for the Analysis of the Expected Behaviour of Cyber-Physical Systems

Valentina Castiglioni[1], Michele Loreti[2] and Simone Tini[3,*]

[1]Reykjavik University, Reykjavik, Iceland
[2]University of Camerino, Camerino, Italy
[3]University of Insubria, Como, Italy

Abstract

The behaviour of systems characterised by a closed interaction of software components with the environment is inevitably subject to uncertainties. We propose a general framework for the specification and verification of requirements on the behaviour of these systems. We introduce the *Distribution Temporal Logic* (*DisTL*), a novel temporal logic allowing us to specify properties on the *expected* behaviour of systems, and to include the presence of uncertainties in the specification. We equip DisTL with a robustness semantics and we prove it sound and complete w.r.t. the semantics induced by the *evolution metric*, i.e., a hemimetric expressing how well a system is fulfilling its tasks with respect to another one. Finally, we give a *statistical model checking algorithm* for DisTL specifications, and we apply our framework to a simple unmanned ground vehicle scenario.

Keywords

Cyber-physical systems, Uncertainty, Evolution sequence, Temporal logic, Statistical model checking

1. Introduction

We have recently proposed a formal framework to model and analyse the behaviour of systems that are subject to *uncertainty*. The most prominent example is that of cyber-physical systems [1] (CPSs), in which software components, or *agents*, must interact with a highly changing and, even, unpredictable environment. To reason on these systems, we have introduced, in [2, 3], the *evolution sequence model*: the behaviour of the system is modelled in terms of the modifications that the interaction of the agents with the environment induce on a set of application-relevant data, called *data state*. As those modifications are subject to uncertainty, induced by the environment and system's approximations, we model them as probability measures, henceforth simply called distributions, on the attainable data states. Hence, the evolution sequence of a system is the sequence of distributions over the data states obtained at each step, and can also be seen as the discrete-time version of the cylinder of all possible trajectories of the system, that takes into account the effects of uncertainty at each step. We have also provided the notion of *evolution metric* between evolution sequences, which allows us to quantify the *behavioural distance* [4, 5, 6] between systems.

ICTCS 2023: 24th Italian Conference on Theoretical Computer Science, September 13–15, Palermo, Italy
*Corresponding author.
valentinac@ru.is (V. Castiglioni); michele.loreti@unicam.it (M. Loreti); simone.tini@uninsubria.it (S. Tini)
0000-0002-8112-6523 (V. Castiglioni); 0000-0003-3061-863X (M. Loreti); 0000-0002-3991-5123 (S. Tini)

Then, we have introduced, in [7], a novel temporal logic, called Robustness Temporal Logic (RobTL), allowing us to specify temporal requirements on the evolution of distances between the nominal behaviour of a system and its *perturbed* version. In particular, we can use RobTL formulae to specify robustness properties [8, 9, 10, 11] against uncertainties of the agents in the system. This is made possible by using atomic propositions of the form $\Delta(\mathtt{exp}, \mathtt{p}) \bowtie \eta$, to compare a threshold η with the distance, specified by an expression exp, between a given evolution sequence and its perturbed version, obtained by applying a perturbation specified by p, starting from a given time step. Then, we combine atomic propositions with classic Boolean and temporal operators, in order to extend these evaluations to the entire evolution sequences.

The expressive power of RobTL comes at a price: besides the behaviour of the agents and the environment, we must be able to specify the perturbation that affects the system, in order to measure its robustness. While our tool Stark, the *Software Tool for the Analysis of Robustness in the unKnown environment* [12] available at https://github.com/quasylab/jspear, offers a domain specific language allowing us to do so, it might be the case that we do not know which data are manipulated by a perturbation, nor when and how such manipulations occur. Indeed, it might also be the case that we do not have access to a full specification of the system, but only to a collection of observations on data, following the deployment of the agents in the real world. Although we can still use Stark to perform some comparisons between the *observed* evolution sequence, obtained from the collected observations, and an *ideal* evolution sequence, in this case we cannot use RobTL to specify robustness properties of the system. Our aim, with this paper, is to provide an alternative approach to the study of systems robustness, in order to fill this gap.

Our Contribution: The Distribution Temporal Logic Let us consider a simple scenario: an unmanned ground vehicle is proceeding on a straight path towards a toll booth (henceforth, the objective), where it has to stop to allow the passenger to retrieve the entry ticket to the motorway. With classic probabilistic temporal logics (like, e.g., PCTL [13], CSL [14, 15], probabilistic variants of LTL [16], and probabilistic variants [17, 18] of MTL [19] and STL [20]), we can verify whether the vehicle is going to stop within a certain distance from the objective, and/or whether the probability to do so is above/below a desired threshold. We remark that this is achieved by reasoning in a trace-by-trace fashion: first we check the property over each trajectory of the system, and then we sum up the probability weights of those satisfying it.

Here we want to perform a different analysis: due to the presence of uncertainties, it would be preposterous to require the vehicle to stop precisely at the objective. Our aim is to express that the vehicle is *expected* to stop there, and to allow a certain *variance* on its actual final position, to take uncertainties into account, while guaranteeing that hazardous behaviours are avoided. Technically speaking, we want to specify that, when considering *all possible system behaviours*, the final (i.e., stationary) position of the vehicle agrees with a *desired distribution*, like, e.g., a Gaussian centred over the objective.

To this end, we introduce the *Distribution Temporal Logic* (*DisTL*), a novel temporal logic allowing us to express requirements on the *expected* behaviour of the system in the presence of uncertainties and perturbations, by using distributions over data states as atomic propositions. We equip DisTL with a real-valued semantics expressing the *robustness* of the satisfaction of

DisTL specifications. The *robustness* of a system $\mathbf{s}$ with respect to a formula φ is expressed as a real number $[\![\varphi]\!]_{\mathbf{s}} \in [-1, 1]$: if it is positive, $\mathbf{s}$ satisfies φ. In detail, $[\![\varphi]\!]_{\mathbf{s}}$ describes how much the behaviour of $\mathbf{s}$ has to be modified in order to violate (or satisfy) φ. We can then interpret $[\![\varphi]\!]_{\mathbf{s}}$ as an indicator of *how well* $\mathbf{s}$ behaves with respect to the requirement φ. To formalise "*how well*", we use the evolution metric of [2, 3], a (time-dependent) hemimetric on the evolution sequences of systems based on a *hemimetric on data states* and the *Wasserstein metric* [21]. The reason to opt for a hemimetric, instead of a more standard (pseudo)metric, is that it allows us to compare the *relative behaviour* of two systems and thus to express whether one system is *better than* the other. We use the evolution metric to define the robustness of systems with respect to DisTL formulae. As atomic propositions are distributions over data states, by means of the evolution metric we can compare them to the distributions in the evolution sequences of systems. In this way, we obtain useful information on the differences in the behaviour of two systems from the comparison of their robustness. In particular, we prove the robustness to be *sound* and *complete* with respect to our metric semantics: whenever the robustness of $\mathbf{s}_1$ with respect to a formula φ is greater than the distance between $\mathbf{s}_1$ and $\mathbf{s}_2$, then we can conclude that the robustness of $\mathbf{s}_2$ with respect to φ is positive. We have also implemented in STARK a *statistical algorithm* for the evaluation of systems robustness with respect to DisTL specifications.

In order to show how our techniques can be applied, we consider the unmanned ground vehicle scenario described above as a case study.

2. The Evolution Sequence Model

Evolution Sequences We consider systems consisting of a set of *agents* and an *environment*, whose interaction produces changes on a shared *data space* $\mathcal{D}$, containing the values assumed by *variables*, representing: (i) physical quantities, (ii) sensors, (iii) actuators, and (iv) internal variables of the agents. Technically, we assume a *finite* set of *variables* Var such that for each $x \in \mathrm{Var}$ the domain $\mathcal{D}_x \subseteq \mathbb{R}$ is either *finite*, or a *compact* subset of $\mathbb{R}$. Notice that, in particular, this means that $\mathcal{D}_x$ is a Polish space. Moreover, as a σ-algebra over $\mathcal{D}_x$ we assume the Borel σ-algebra, denoted $\mathcal{B}_x$. The data space $\mathcal{D}$ over Var is then defined as the Cartesian product over the variables domains $\mathcal{D} = \bigtimes_{x \in \mathrm{Var}} \mathcal{D}_x$, and it is equipped with the product σ-algebra $\mathcal{B}_{\mathcal{D}} = \bigotimes_{x \in \mathrm{Var}} \mathcal{B}_x$ [22].

We call *data state* the current state of the data space, and represent it by a mapping $\mathbf{d} \colon \mathrm{Var} \to \mathbb{R}$, with $\mathbf{d}(x) \in \mathcal{D}_x$ for all $x \in \mathrm{Var}$. 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 data at the next step. Hence, following [2], 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 other words, the evolution sequence is the discrete-time version of the cylinder of *all possible trajectories* of the system. In this paper, we do not focus on how evolution sequences are generated: we simply assume a Markov kernel governing the evolution of the system, and the evolution sequence is the *Markov process* generated by it.

Definition 1 (Evolution sequence, [2])**.** Given a data space $\mathcal{D}$, let $\Delta(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$ be the set of

distributions over the space $(\mathcal{D}, \mathcal{B}_\mathcal{D})$. Let $\mathsf{step}\colon \mathcal{D} \to \Delta(\mathcal{D}, \mathcal{B}_\mathcal{D})$ be the Markov kernel generating the behaviour of a system $\mathbf{s}$ having μ as initial distribution. Then, the *evolution sequence* of $\mathbf{s}$ is a countable sequence $\mathcal{S}_\mu = \mathcal{S}^0_\mu, \mathcal{S}^1_\mu, \ldots$ of distributions in $\Delta(\mathcal{D}, \mathcal{B}_\mathcal{D})$ such that, for all $\mathbb{D} \in \mathcal{B}_\mathcal{D}$:

$$\mathcal{S}^0_\mu(\mathbb{D}) = \mu(\mathbb{D})$$
$$\mathcal{S}^{i+1}_\mu(\mathbb{D}) = \int_\mathcal{D} \mathsf{step}(\mathbf{d})(\mathbb{D})\, \eth \mathcal{S}^i_\mu(\mathbf{d}).$$

We denote by $\mathbf{S}_\mathcal{D}$ the set of all possible evolution sequences over $\mathcal{D}$.

A Distance on Evolution Sequences We introduce a distance measuring the differences in the behaviour of systems that will be used to define the robustness of DisTL specifications. The idea is first to introduce a *distance on distributions over data states* measuring their differences with respect to a *given target*, and then to extend it to the evolution sequences. Following [2], to capture the tasks of the system, we use *penalty functions* $\rho\colon \mathcal{D} \times \mathbb{N} \to [0, 1]$ i.e., functions that assign to each data state $\mathbf{d}$ and time step 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 time τ. For brevity, we denote by ρ_τ the mapping corresponding to the τ-th element in the list ρ, i.e., $\rho_\tau(\mathbf{d}) = \rho(\mathbf{d}, \tau)$.

We have implemented a simple language, PF, to model penalties in STARK:

Definition 2 (Penalties)**.** Penalties in PF are defined as follows:

$$\mathtt{pf} \;::=\; \mathbf{0} \;\mid\; \mathfrak{f}@\tau \;\mid\; \mathtt{pf}_1 \,;\, \mathtt{pf}_2 \;\mid\; \mathtt{pf}^n$$

where pf ranges over PF, n and τ are finite natural numbers, and:

- $\mathbf{0}$ is the *null penalty*, i.e., at each time step it assigns penalty 0 to any data state;
- $\mathfrak{f}@\tau$ is an *atomic penalty*, i.e., a function $\mathfrak{f}\colon \mathcal{D} \to [0, 1]$ that is applied after τ time steps from the current instant;
- $\mathtt{pf}_1 \,;\, \mathtt{pf}_2$ is a *sequential penalty*, i.e., penalty $\mathtt{pf}_2$ is applied at the time step subsequent to the (final) application of $\mathtt{pf}_1$;
- $\mathtt{pf}^n$ is an *iterated penalty*, i.e., penalty pf is applied for a total of n times.

This simple language allows us to define some non-trivial penalties that we can use to analyse systems behaviour, like in the following example designed for our case study:

***Example* 1.** As the task of the vehicle is to stop at the objective, it is natural to use the normalisation of its distance from it, stored in variable p_dist, as a penalty. However, we notice that since the distance changes in time, in order to have a meaningful penalty value we also need to change the normalisation factor. In fact, if, for instance, we normalise only with respect to the initial distance, the penalty is fated to decrease in time, and thus we may loose important information on the behaviour of the vehicle when it gets closer to the objective. Therefore,

assuming that the vehicle is at an initial distance from the objective of 10000 m, we consider the following penalty:

$$\rho_{\text{pos}} = (\mathfrak{f}_{10000}@0)^{100}; (\mathfrak{f}_{7000}@0)^{100}; (\mathfrak{f}_{2500}@0)^{75}; (\mathfrak{f}_{10}@0)^{76} \quad \mathfrak{f}_x(\mathbf{d}) = \frac{\mathbf{d}(\mathsf{p\_dist})}{x}.$$

Each $\mathtt{pf} \in \mathtt{PF}$ denotes a penalty function of type $\mathcal{D} \times \mathbb{N} \to [0,1]$. To obtain it, we employ two auxiliary functions: effect and next. Both functions are defined inductively on the structure of penalties (where, as no confusion arises from the context, we use $\mathbf{0}$ to denote both, the null penalty, and the function mapping each data state into 0). Function $\mathsf{effect}(\mathtt{pf})$ describes the effect of $\mathtt{pf}$ at the current step:

$$\begin{aligned}
&\mathsf{effect}(\mathbf{0}) = \mathbf{0}\\
&\mathsf{effect}(\mathfrak{f}@\tau) = \begin{cases} \mathbf{0} & \text{if } \tau > 0,\\ \mathfrak{f} & \text{if } \tau = 0 \end{cases}\\
&\mathsf{effect}(\mathtt{pf}^n) = \mathsf{effect}(\mathtt{pf})\\
&\mathsf{effect}(\mathtt{pf}_1; \mathtt{pf}_2) = \mathsf{effect}(\mathtt{pf}_1)
\end{aligned}$$

Function $\mathsf{next}(\mathtt{pf})$ identifies the penalty that will be applied at the next step:

$$\begin{aligned}
&\mathsf{next}(\mathbf{0}) = \mathbf{0}\\
&\mathsf{next}(\mathfrak{f}@\tau) = \begin{cases} \mathfrak{f}@(\tau-1) & \text{if } \tau > 0,\\ \mathbf{0} & \text{otherwise} \end{cases}\\
&\mathsf{next}(\mathtt{pf}^n) = \begin{cases} \mathsf{next}(\mathtt{pf}); \mathtt{pf}^{n-1} & \text{if } n > 0,\\ \mathbf{0} & \text{otherwise} \end{cases}\\
&\mathsf{next}(\mathtt{pf}_1; \mathtt{pf}_2) = \begin{cases} \mathsf{next}(\mathtt{pf}_1); \mathtt{pf}_2 & \text{if } \mathsf{next}(\mathtt{pf}_1) \neq \mathbf{0},\\ \mathtt{pf}_2 & \text{otherwise.} \end{cases}
\end{aligned}$$

We can now define the semantics of penalties as the mapping $\langle\cdot\rangle \colon \mathtt{PF} \to (\mathcal{D} \times \mathbb{N} \to [0,1])$ such that, for all $\mathbf{d} \in \mathcal{D}$ and $i \in \mathbb{N}$:

$$\langle \mathtt{pf} \rangle(\mathbf{d}, i) = \mathsf{effect}(\mathsf{next}^i(\mathtt{pf}))(\mathbf{d}),$$

where $\mathsf{next}^0(\mathtt{pf}) = \mathtt{pf}$ and $\mathsf{next}^i(\mathtt{pf}) = \mathsf{next}(\mathsf{next}^{i-1}(\mathtt{pf}))$, for all $i > 0$.

The next proposition follows directly from the definition of the mapping $\langle\cdot\rangle$.

Proposition 1. *For each* $\mathtt{pf} \in \mathtt{PF}$*, the mapping* $\langle \mathtt{pf} \rangle$ *is a penalty function.*

Then we use penalty functions to obtain a *distance on data states*:

Definition 3 (Metric on data states)**.** Let $\rho \colon \mathcal{D} \times \mathbb{N} \to [0,1]$ be a penalty function on $\mathcal{D}$, and let $\tau \in \mathbb{N}$ be a time step. The *metric on data states* in $\mathcal{D}$, $\mathfrak{m}^\rho_\tau \colon \mathcal{D} \times \mathcal{D} \to [0,1]$, is defined, for all $\mathbf{d}_1, \mathbf{d}_2 \in \mathcal{D}$, by

$$\mathfrak{m}^\rho_\tau(\mathbf{d}_1, \mathbf{d}_2) = \max\{\rho_\tau(\mathbf{d}_2) - \rho_\tau(\mathbf{d}_1), 0\}.$$

Note that $\mathfrak{m}_\tau^\rho(\mathbf{d}_1, \mathbf{d}_2)$ is a hemimetric expressing how much $\mathbf{d}_2$ is *worse* than $\mathbf{d}_1$ according to ρ_τ. Then, we need to lift the hemimetric $\mathfrak{m}_\tau^\rho$ to a hemimetric over $\Delta(\mathcal{D}, \mathcal{B}_\mathcal{D})$. To this end, we make use of the *Wasserstein lifting* [21]: for any two distributions μ, ν on $(\mathcal{D}, \mathcal{B}_\mathcal{D})$, the Wasserstein lifting of $\mathfrak{m}_\tau^\rho$ to a distance between μ and ν is defined by

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

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 [3, 23] for a discussion on the definition of the Wasserstein lifting over hemimetrics.)

The evolution hemimetric of [2] is then obtained as the *infinity norm* of the tuple of the Wasserstein distances between the distributions in the evolution sequences. Since in most applications the changes on data induced by the system can be appreciated only along wider time intervals than a computation step by the agents, we consider a *discrete, finite* set OT of time steps at which the modifications on data give us useful information on the evolution of the system.

Definition 4 (Evolution metric)**.** Assume a finite set OT of time steps, a penalty function ρ and the metrics on data states $\mathfrak{m}_\tau^\rho$. Then, the *evolution metric* over ρ and OT, is the mapping $\mathfrak{Em}_{\mathrm{OT}}^\rho \colon \mathbf{S}_\mathcal{D} \times \mathbf{S}_\mathcal{D} \to [0, 1]$ defined, for all $\mathcal{S}_1, \mathcal{S}_2$, by

$$\mathfrak{Em}_{\mathrm{OT}}^\rho(\mathcal{S}_1, \mathcal{S}_2) = \max_{\tau \in \mathrm{OT}} \mathbf{W}(\mathfrak{m}_\tau^\rho)(\mathcal{S}_{1,\tau}, \mathcal{S}_{2,\tau}).$$

We remark that since we are interested in verifying requirements over a finite horizon (and we use only *bounded* temporal operators in the logic), the choice of having OT finite is not too restrictive.

Case Study: Unmanned Ground Vehicle As a running example, we consider the unmanned ground vehicle scenario described in the Introduction. Since our objective is merely to showcase the features of the new logic, we work under the following simplifying assumptions: 1. All the objects on the scene, including the vehicle, are one-dimensional; 2. When we start the simulation, the (program controlling the) vehicle already knows its distance from the point at which it has to stop (i.e., its distance from the button on the toll booth, henceforth referred to as *the objective*); 3. The acceleration of the vehicle can assume two values: a positive one A $\mathrm{m/s^2}$, and a negative one -B $\mathrm{m/s^2}$ (with B> 0); 4. The vehicle is equipped with a speed sensor, s_speed, that is subject to uncertainty (related to instrument accuracy); 5. The speed can never be negative (i.e., we do not allow the vehicle to shift into reverse). Every TIMER steps, the vehicle decides whether to accelerate or brake, according to the sensor readings and its distance from the objective.

To specify the details of the behaviour of the vehicle we use our tool STARK. The vehicle is modelled as a STARK component, consisting of a set of local variables, used to allocate sensor readings and the value of the acceleration actuator, and a STARK controller, i.e., the process rendering the behaviour of the vehicle. Due to space limitations, we give only an informal

presentation. The interested reader can find the full STARK specification at https://github.com/quasylab/jspear/tree/Tony. The behaviour of the controller is specified by means of four processes, or *states*: Ctrl, Accelerate, Decelerate and Stop. The computation starts from Ctrl that checks, every TIMER steps, whether the vehicle can accelerate or if it has to brake, and sets the acceleration actuator accordingly. The decision is taken on the basis of the sensed speed and the distance from the objective. States Accelerate and Decelerate manage, respectively, the acceleration and braking phases: the vehicle maintains a constant acceleration (of A $\mathrm{m/s^2}$ in the case of Accelerate, and of -B $\mathrm{m/s^2}$ for Decelerate), for TIMER steps; then Ctrl is woken up for a new check. When the speed becomes zero, and it is not possible to get closer to the objective, process Stop sets the acceleration actuator to $0\,\mathrm{m/s^2}$, and the vehicle becomes stationary.

To model the evolution of the scenario we use a STARK environment, that allows us to set the position of the objective, and to model the movement of the vehicle towards it. In this simple scenario, the uncertainty in the model is given by the precision error in the readings of sensor s_speed. Hence, we include a random noise in the updates of that value made by the environment.

3. The Distribution Temporal Logic

We now introduce the *Distribution Temporal Logic* (DisTL) which allows us to specify requirements on the *expected* behaviour of the system, in the presence of uncertainties and perturbations. The logic bases on two atomic properties, $\mathbf{target}(\mu)^\rho_q$ and $\mathbf{brink}(\mu)^\rho_q$, where μ is a distribution over data states in $(\mathcal{D}, \mathcal{B}_\mathcal{D})$, ρ is a penalty function and q is a real in $[0, 1]$. We will use $\mathbf{target}(\mu)^\rho_q$ to express a desirable behaviour, whereas $\mathbf{brink}(\mu)^\rho_q$ can be used for unwanted, or hazardous, behaviours. These formulae are evaluated over a evolution sequence $\mathcal{S}$ and a time step τ. Let us analyse the formula $\mathbf{target}(\mu)^\rho_q$. To establish whether the system exhibits the desired behaviour, we compare the given distribution μ with the distribution $\mathcal{S}_\tau$: our means of comparison is the Wasserstein lifting of the hemimetric between data states evaluated with respect to the penalty ρ. (Notice that ρ is a parameter of the formula $\mathbf{target}(\mu)^\rho_q$. This is due to the fact that the penalty is not a property of the system but part of the requirements imposed on its behaviour.) As μ is our target distribution, it is natural to check whether $\mathcal{S}_\tau$ is *worse than* μ, i.e., to evaluate the distance $\mathbf{W}(\mathfrak{m}^\rho_\tau)(\mu, \mathcal{S}_\tau)$. Given the presence of uncertainties, it would not be feasible to say that the system satisfies the considered formula if and only if $\mathbf{W}(\mathfrak{m}^\rho_\tau)(\mu, \mathcal{S}_\tau) = 0$. Instead, we use the parameter q as a tolerance on the distance: if $\mathcal{S}_\tau$ is such that $\mathbf{W}(\mathfrak{m}^\rho_\tau)(\mu, \mathcal{S}_\tau) \leq q$, then the behaviour of the system can be considered acceptable. In other words, q is the *maximal acceptable distance* between the desired behaviour μ and the current behaviour $\mathcal{S}_\tau$.

Conversely, in the formula $\mathbf{brink}(\mu)^\rho_q$ the distribution μ expresses some *unwanted, hazardous*, behaviour. Hence, the distribution $\mathcal{S}_\tau$ reached by the system must be better than μ, i.e., $\mathbf{W}(\mathfrak{m}^\rho_\tau)(\mathcal{S}_\tau, \mu) > 0$. Also in this case, due to the presence of uncertainties, we need to make use of a threshold parameter q: assuming a distribution $\mathcal{S}_\tau$ acceptable when it is only *slightly* better than μ can still lead to an unwanted behaviour (because, in this case, the difference between the two distributions may be due only to some noise). Hence, we let q be the *minimal required distance* between $\mathcal{S}_\tau$ and μ, so that $\mathcal{S}_\tau$ is an acceptable behaviour if and only if $\mathbf{W}(\mathfrak{m}^\rho_\tau)(\mathcal{S}_\tau, \mu) \geq q$.

Let $\mathrm{var}(\mu) \subseteq \mathrm{Var}$ be the set of data variables over which the distribution μ is defined.

Similarly, for a penalty function ρ, we can consider the set $\mathrm{var}(\rho) \subseteq \mathrm{Var}$.

Definition 5 (DisTL)**.** The modal logic DisTL consists in the set of formulae $\mathcal{L}$ defined by the following syntax:

$$\varphi ::= \top \mid \mathbf{target}(\mu)^{\rho}_{q} \mid \mathbf{brink}(\mu)^{\rho}_{q} \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi_1 \,\mathcal{U}^{[a,b]}\, \varphi_2$$

with φ ranging over $\mathcal{L}$, $\mu \in \Delta(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$ a distribution over data states, ρ a penalty function such that $\mathrm{var}(\rho) \subseteq \mathrm{var}(\mu)$, $q \in [0,1]$, and $[a,b]$ an interval in OT.

Disjunction and negation are the standard Boolean connectives, and $\varphi_1 \,\mathcal{U}^{[a,b]}\, \varphi_2$ is the *bounded until* operator stating that φ_1 is satisfied until, at a time in $[a,b]$, φ_2 is. As expected, other standard operators can be defined as macros in our logic:

$$\varphi_1 \wedge \varphi_2 \equiv \neg(\neg\varphi_1 \vee \neg\varphi_2) \qquad \Diamond^{[a,b]}\varphi \equiv \top \,\mathcal{U}^{[a,b]}\, \varphi_2 \qquad \Box^{[a,b]}\varphi \equiv \neg\Diamond^{[a,b]}\neg\varphi.$$

Formulae are evaluated over evolution sequences and time steps. In a *quantitative semantics* approach, for a formula φ, an evolution sequence $\mathcal{S}$, and a time step τ, the value $[\![\varphi]\!]_{\mathcal{S},\tau} \in [-1,1]$ expresses the *robustness* of $\mathcal{S}$ with respect to φ at time τ, i.e., how much the behaviour $\mathcal{S}_\tau$ can be modified, either while preserving the validity of property φ (if φ is already satisfied), or in order to obtain it.

Definition 6 (DisTL: quantitative semantics)**.** For any evolution sequence $\mathcal{S}$, time step τ, and DisTL formula φ, the *robustness* of $\mathcal{S}$ with respect to φ at τ, notation $[\![\varphi]\!]_{\mathcal{S},\tau} \in [-1,1]$, is defined inductively with respect to the structure of φ as follows:

$$\begin{aligned}
[\![\top]\!]_{\mathcal{S},\tau} &= 1 \\
[\![\mathbf{target}(\mu)^{\rho}_{q}]\!]_{\mathcal{S},\tau} &= q - \mathbf{W}(\mathfrak{m}^{\rho}_{\tau})(\mu, \mathcal{S}_\tau) \\
[\![\mathbf{brink}(\mu)^{\rho}_{q}]\!]_{\mathcal{S},\tau} &= \mathbf{W}(\mathfrak{m}^{\rho}_{\tau})(\mathcal{S}_\tau, \mu) - q \\
[\![\neg\varphi]\!]_{\mathcal{S},\tau} &= -[\![\varphi]\!]_{\mathcal{S},\tau} \\
[\![\varphi_1 \vee \varphi_2]\!]_{\mathcal{S},\tau} &= \max\{[\![\varphi_1]\!]_{\mathcal{S},\tau}, [\![\varphi_2]\!]_{\mathcal{S},\tau}\} \\
[\![\varphi_1 \,\mathcal{U}^{[a,b]}\, \varphi_2]\!]_{\mathcal{S},\tau} &= \max_{\tau' \in [\tau+a, \tau+b]} \min\left\{ [\![\varphi_2]\!]_{\mathcal{S},\tau'}, \min_{\tau'' \in [\tau+a, \tau')} [\![\varphi_1]\!]_{\mathcal{S},\tau''} \right\}.
\end{aligned}$$

Intuitively, the value $\mathbf{W}(\mathfrak{m}^{\rho}_{\tau})(\mu, \mathcal{S}_\tau)$ quantifies the difference between the distribution $\mathcal{S}_\tau$ reached by the system at time τ and μ. Hence, on the one hand the robustness $[\![\mathbf{target}(\mu)^{\rho}_{q}]\!]_{\mathcal{S},\tau}$ expresses whether the distribution in the evolution sequence of $\mathbf{s}$ is within the maximal acceptable distance q from μ. On the other hand, it also expresses how much $\mathcal{S}_\tau$ can be modified while guaranteeing that the behaviour of the system remains within the specified parameters. Clearly, the closer μ and $\mathcal{S}_\tau$, the higher the robustness. Similarly, $[\![\mathbf{brink}(\mu)^{\rho}_{q}]\!]_{\mathcal{S},\tau}$ quantifies the robustness with respect to μ (and ρ) in terms of how much $\mathcal{S}_\tau$ may *get close* to μ while keeping the minimal required distance q. Hence, the farther $\mathcal{S}_\tau$ and μ, the higher the robustness. The semantics of boolean connectives and bounded until is standard. Notice that due to the potential asymmetry of our distances, it is not true in general that $\mathbf{brink}(\mu)^{\rho}_{q} = \neg\mathbf{target}(\mu)^{\rho}_{1-q}$.

***Example* 2.** Let us now use DisTL to formalise the requirement on the final position of the vehicle discussed at the beginning of this section: we want to express that it is distributed like a Gaussian centred on the objective and with variance σ^2, for some σ. In the STARK implementation of the system, the position of the vehicle is modelled in terms of its physical distance from the objective: when variable p_dist equals 0, the position of the vehicle corresponds to the objective. Hence, we can use

$$\mu_{\text{pos}} = \text{p\_dist} \sim \mathcal{G}(0, \sigma^2)$$

as the target distribution over the position. Given the penalty ρ_{pos} defined in Example 1, and a desired tolerance ε, we can use the formula

$$\varphi_1 = \Box^{[\tau_1,\tau_2]}\mathbf{target}(\mu_{\text{pos}})_{\varepsilon}^{\rho_{\text{pos}}}$$

to capture the requirement on the final position, where the time interval $[\tau_1, \tau_2]$ is chosen according to the systems parameters.

Clearly, we can use DisTL also to express strict requirements (i.e., without approximations and tolerances): for instance, we must require that the vehicle is stationary in the final position, i.e., that its speed equals 0. This can be done by means of a Dirac (or point) distribution $\delta_0(\text{p\_speed})$, henceforth denoted μ_{sp}, where p_speed is the variable storing the value of the physical speed of the vehicle. Consider the penalty function

$$\rho_{\text{sp}}(\mathbf{d}) = (\mathbf{d}(\text{p\_speed})/\text{MAX\_SPEED@0})^{\mathfrak{h}},$$

where MAX_SPEED is the parameter storing the maximal speed of the vehicle. We can then use the atomic formula $\mathbf{target}(\mu_{\text{sp}})_0^{\rho_{\text{sp}}}$ to express that the speed of the vehicle *must* be 0 (notice the tolerance 0). Then, we can combine the two formulae to express that the vehicle is expected to stop in an ε-neighbourhood of the objective, within a time horizon $\mathfrak{h}$ (determined according to the other parameters of the system, like TIMER, A, and B):

$$\varphi_2 = \Diamond^{[0,\mathfrak{h}]}\left(\mathbf{target}(\mu_{\text{sp}})_0^{\rho_{\text{sp}}} \wedge \mathbf{target}(\mu_{\text{pos}})_{\varepsilon}^{\rho_{\text{pos}}}\right).$$

Soundness and Completeness of the Robustness Semantics We can show that DisTL characterises the distance between evolution sequences. More precisely, the quantitative semantics of DisTL induces a distance between evolution sequences that coincides with the symmetrisation of the hemimetric $\mathfrak{Em}_{\text{OT}}^{\rho}$ and is therefore a pseudometric. Clearly, since the evolution metric is defined in terms of a given penalty function ρ, it will be characterised by the distance over formulae in $\mathcal{L}_\rho$, which is the sub-class of $\mathcal{L}$ with atomic propositions of the form $(\cdot)_{(\cdot)}^{\rho}$.

Definition 7 (DisTL distance)**.** The *DisTL distance* between $\mathcal{S}_1, \mathcal{S}_2 \in \mathbf{S}_{\mathcal{D}}$ with respect to a penalty function ρ and OT is defined as

$$\mathfrak{L}_{\text{OT}}^{\rho}(\mathcal{S}_1, \mathcal{S}_2) = \sup_{\varphi\in\mathcal{L}_\rho,\tau\in\text{OT}} \left|[\![\varphi]\!]_{\mathcal{S}_1,\tau} - [\![\varphi]\!]_{\mathcal{S}_2,\tau}\right|.$$

Firstly, we show that the symmetrisation of $\mathfrak{Em}_{\text{OT}}^{\rho}$ is an upper bound to $\mathfrak{L}_{\text{OT}}^{\rho}$.

Lemma 1. *For any penalty function ρ, and $\mathcal{S}_1, \mathcal{S}_2 \in \mathbf{S}_{\mathcal{D}}$ we have:*

$$\mathfrak{L}^{\rho}_{\mathrm{OT}}(\mathcal{S}_1, \mathcal{S}_2) \leq \max\left\{\mathfrak{Em}^{\rho}_{\mathrm{OT}}(\mathcal{S}_1, \mathcal{S}_2), \mathfrak{Em}^{\rho}_{\mathrm{OT}}(\mathcal{S}_2, \mathcal{S}_1)\right\}.$$

Then, we show that for all evolution sequences $\mathcal{S}_1, \mathcal{S}_2$ there exists a formula φ in $\mathcal{L}_\rho$ such that the symmetrisation of $\mathfrak{Em}^{\rho}_{\mathrm{OT}}$ coincides with the difference in the evaluations of φ over $\mathcal{S}_1, \mathcal{S}_2$.

Lemma 2. *For all $\mathcal{S}_1, \mathcal{S}_2 \in \mathbf{S}_{\mathcal{D}}$ and penalty functions ρ, there is a formula $\varphi \in \mathcal{L}_\rho$ with*

$$|[\![\varphi]\!]_{\mathcal{S}_1,\tau} - [\![\varphi]\!]_{\mathcal{S}_2,\tau}| = \max\left\{\mathfrak{Em}^{\rho}_{\mathrm{OT}}(\mathcal{S}_1, \mathcal{S}_2), \mathfrak{Em}^{\rho}_{\mathrm{OT}}(\mathcal{S}_2, \mathcal{S}_1)\right\},$$

for some $\tau \in \mathrm{OT}$.

From Lemma 1 and Lemma 2 we infer that the DisTL distance $\mathfrak{L}^{\rho}_{\mathrm{OT}}$ and the symmetrisation of $\mathfrak{Em}^{\rho}_{\mathrm{OT}}$ coincide.

Theorem 1. *For all evolution sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ we have that:*

$$\mathfrak{L}^{\rho}_{\mathrm{OT}}(\mathcal{S}_1, \mathcal{S}_2) = \max\left\{\mathfrak{Em}^{\rho}_{\mathrm{OT}}(\mathcal{S}_1, \mathcal{S}_2), \mathfrak{Em}^{\rho}_{\mathrm{OT}}(\mathcal{S}_2, \mathcal{S}_1)\right\}.$$

Theorem 1 entails the *soundness* (Lemma 1) and *completeness* (Lemma 2) of our notion of robustness. In particular, as a direct consequence of Theorem 1, we can obtain the following classic result (see, e.g., [24]): whenever the robustness of a evolution sequence $\mathcal{S}$ with respect to a formula φ is greater than the distance between $\mathcal{S}$ and $\mathcal{S}'$, then the robustness of $\mathcal{S}'$ with respect to φ is positive as well.

Corollary 1. *Let φ be any formula in $\mathcal{L}_\rho$, $\tau \in \mathrm{OT}$ and let $i \in \{1, 2\}$. Whenever $[\![\varphi]\!]_{\mathcal{S}_i,\tau} \geq \max\left\{\mathfrak{Em}^{\rho}_{\mathrm{OT}}(\mathcal{S}_1, \mathcal{S}_2), \mathfrak{Em}^{\rho}_{\mathrm{OT}}(\mathcal{S}_2, \mathcal{S}_1)\right\}$, then $[\![\varphi]\!]_{\mathcal{S}_{3-i},\tau} \geq 0$.*

4. Statistical Model Checking

In this section we present an algorithm, based on statistical techniques and simulation, that allows us to estimate the robustness of a system $\mathbf{s}$ with respect to a DisTL formula φ. This algorithm consists in three basic steps:

(i) A randomised procedure, based on simulation, that allows us to estimate the evolution sequence of system $\mathbf{s}$, assuming an initial data state $\mathbf{d_s}$: Starting from $\mathbf{d_s}$ we sample N sequences of data states $\mathbf{d}^j_0, \ldots, \mathbf{d}^j_k$, for $j = 1, \ldots, N$; then all the data states collected at time i are used to estimate the distribution $\mathcal{S}^i_{\delta_{\mathbf{d_s}}}$.

(ii) A mechanism to estimate the Wasserstein distance between two (unknown) distributions μ and ν on $(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$, similar to the one presented in [25]: To estimate $\mathbf{W}(\mathfrak{m}^{\rho}_i)(\mu, \nu)$ we use N independent samples $\{\mathbf{d}^1_1, \ldots, \mathbf{d}^N_1\}$ taken from μ and ℓN independent samples $\{\mathbf{d}^1_2, \ldots, \mathbf{d}^{\ell N}_2\}$ taken from ν.

(iii) A procedure that computes the robustness by inspecting the syntax of φ and by using the first two components.

1: **function** SAT($\mathbf{d_s}, \tau, \varphi, \ell, N$)
2: $\mathfrak{h} \leftarrow$ HORIZON(φ)
3: $\overline{E} \leftarrow$ ESTIMATE($\mathbf{d_s}, \mathfrak{h}, N$)
4: **return** EVAL($\overline{E}, \tau, \varphi, \ell, N$)
5: **end function**

Figure 1: Function used to evaluate system robustness with respect to a formula.

The proposed approach has been implemented in Java, as part of STARK, and is available at https://github.com/quasylab/jspear/tree/Tony. We omit the presentation of the first two steps (that have already been discussed at length in [3, 7]), and we give an overview of the third step. We limit ourselves to recall the following result, from [2, 3], on the estimation of the Wasserstein distance:

Theorem 2 ([2, 3]). *Let $\mu, \nu \in \Delta(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$ be unknown. Let $\{\mathbf{d}_1^1, \ldots, \mathbf{d}_1^N\}$ be independent samples taken from μ, and $\{\mathbf{d}_2^1, \ldots, \mathbf{d}_2^{\ell \cdot N}\}$ independent samples taken from ν. Let $\{\omega_j = \rho_i(\mathbf{d}_1^j)\}$ and $\{\nu_h = \rho_i(\mathbf{d}_2^h)\}$ be the ordered sequences obtained by applying the penalty ρ_i to the samples. Then, it holds, almost surely, that $\mathbf{W}(\mathfrak{m}_i^\rho)(\mu, \nu) = \lim_{N \to \infty} \frac{1}{\ell N} \sum_{h=1}^{\ell N} \max\left\{\nu_h - \omega_{\lceil \frac{h}{\ell} \rceil}, 0\right\}$.*

Statistical Estimation of Robustness The computation of the robustness of a system $\mathbf{s}$ with respect to a formula φ, at a given time step τ and starting from the data state $\mathbf{d_s}$, is performed via the function SAT given in Figure 1. Together with the data state $\mathbf{d_s}$, the step τ, and the formula φ, function SAT takes as parameters the two integers ℓ and N identifying the number of samplings that will be used to estimate the Wasserstein metric. This function consists of three steps. First the *time horizon* $\mathfrak{h}$ of the formula φ is computed (by induction on the structure of φ) to identify the number of steps needed to evaluate the robustness. In the second step, function ESTIMATE is used to simulate the evolution sequence of $\mathbf{s}$ from $\mathbf{d_s}$ by collecting the sets of samplings $\overline{E} = E_0, \ldots, E_{\mathfrak{h}}$. Then, in the third step, function EVAL, presented in Figure 2, is used for the evaluation of the robustness. The structure of EVAL is similar to the monitoring function for STL defined in [20]. Function EVAL is defined recursively on the syntax of φ. In the cases of the atomic formulae $\mathbf{target}(\mu)_q^\rho$ and $\mathbf{brink}(\mu)_q^\rho$, firstly we use function SAMPLE to obtain $\ell \cdot N$ independent samples of the distribution μ. Then, we use function WASS to compute the Wasserstein distance between the sampling of μ and the sampling E_τ of the distribution reached at step τ by $\mathbf{s}$.

The following theorem guarantees that when N goes to infinite, the robustness computed by function SAT converges, almost surely, to the exact value.

Theorem 3. *For any formula φ, system $\mathbf{s}$, data state $\mathbf{d_s}$, time step τ, and integer $\ell > 0$*

$$\lim_{N \mapsto \infty} \mathit{SAT}(\mathbf{d_s}, \tau, \varphi, \ell, N) = [\![\varphi]\!]_{\mathcal{S}_{\mathbf{d_s}}, \tau}.$$

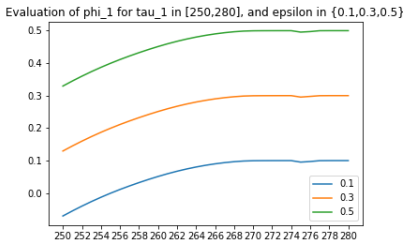
Proof. The proof follows by induction on the structure of the formula φ, using Theorem 2 to deal with the base cases of $\varphi = \mathbf{target}(\mu)_p^\rho$ and $\varphi = \mathbf{brink}(\mu)_p^\rho$. □

```
function Eval(E, τ, φ, ℓ, N)
    match φ
        with ⊤ :
            return 1.0
        with target(μ)_q^ρ :
            E' ← Sample(μ, ℓ · N)
            return q − Wass(E', E_τ, ρ)
        with brink(μ)_p^ρ :
            E' ← Sample(μ, ℓ · N)
            return Wass(E_τ, E', ρ) − q
        with ¬φ_1 :
            return −Eval(E, τ, φ_1, ℓ, N)
        with φ_1 ∨ φ_2 :
            return max{Eval(E, τ, φ_1, ℓ, N), Eval(E, τ, φ_2, ℓ, N)
        with φ_1 U^[a,b] φ_2 :
            res ← 0.0
            for i ∈ [τ + a, τ + b] do
                res_1 ← 0.0
                for j ∈ [τ + a, i] do
                    res_1 ← min{res_1, Eval(E, j, φ_1, ℓ, N)}
                end for
                res ← max{res, min{res_1, Eval(E, i, φ_2, ℓ, N)}}
            end for
            return res
end function
```

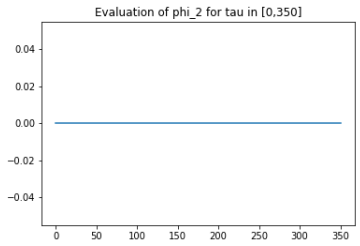
Figure 2: Evaluation of robustness.

***Example* 3.** The procedure outlined above can be used for the evaluation of the requirements on the vehicle scenario presented in Example 2. Let $\mathcal{V}$ be the evolution sequence of the system obtained from the following initial parameters: p_dist $= 10000$; p_speed $= 25.0$; MAX_SPEED $= 40.0$; A $= 0.25$; B $= 2.0$; TIMER $= 1.0$. In Figure 3a we report the evaluations, in $\mathcal{V}$ and time step 0, of various instances of the formula φ_1. We consider a different instance for each $\tau_1 \in [250, 280]$, while we fix the other parameters: $\tau_2 = 350$; $N = 100$ and $\ell = 10$. For each instance, we consider three variations, according to the threshold of the atomic formula: $\varepsilon \in \{0.1, 0.3, 0.5\}$.

Finally, we remark that $[\![\varphi_2]\!]_{\mathcal{V},\tau} = 0.0$ for all $\tau \in [0, 350]$, as shown in Figure 3b. In terms of robustness semantics, an evaluation to 0.0 is usually considered as non-informative, as it gives us no information on how much the behaviour of the system can be modified in order to validate (or invalidate) the considered property. However, in this specific case, 0.0 is the best value that we can obtain. In fact, the formula φ_2 contains a strict requirement on the speed, which is required to be exactly 0.0. Hence, $[\![\varphi_2]\!]_{\mathcal{V},\tau} = 0.0$ means that this requirement is actually met and, at the same time, even the tiniest modification in the behaviour might cause the formula to be invalidated.



(a) Evaluations of φ_1.



(b) Evaluations of φ_2.

Figure 3: Evaluations of various instances of formulae φ_1 and φ_2.

5. Concluding Remarks

Differently from the other probabilistic temporal logics usually considered in the literature, DisTL can be used to express the properties of the distributions expressing the transient and expected behaviour of the system. Up to our knowledge, [26] is the only other paper proposing to substitute probabilistic guarantees on the temporal properties with a richer description of the probabilistic events. In detail, [26] introduces *ProbSTL*, a stochastic variant of STL tailored to the incremental runtime verification of safe behaviour of robotic systems under uncertainties, based on a predictive stream reasoning tool: their stochastic signal is given by the prediction on the possible future trajectories of a system, taking uncertainties into account. Yet, ProbSTL specifications are tested only on the current trajectory of the system. This is the main difference with our work, since our logic has been built to express the overall uncertain behaviour of the system. This disparity is also a consequence of the different application context: off-line verification for us, runtime verification in [26]. However, as future work, we plan to develop a predictive model for the runtime monitoring of DisTL specifications. In particular, inspired by [27, 28] where deep neural networks are used as reachability predictors for predictive monitoring, we intend to integrate our work with learning techniques, to favour the computation and evaluation of the predictions.

In Markov processes as *transformers of distributions* [29, 30], state-to-state transition probabilities are interpreted as a single distribution over the state space. We remark that the state space in [29, 30] is finite and discrete, whereas our evolution sequences are defined in the continuous setting, which means that we are not introducing any limiting assumption on the behaviour of the environment. Moreover, the temporal logics used to model check properties of transformers of distributions, respectively *iLTL* in [29] and the *almost acyclic* Büchi automata in [30], have a boolean semantics, and are thus not comparable to DisTL, in which formulae are interpreted in terms of robustness.

A statistical model checking algorithm for PCTL specifications over Markov chains has been proposed in [31], using stratified sampling. This allows for the generation of negatively correlated samples, thus reducing the number of samples needed to obtain confident results at the price of restricting the form of the PCTL formulae to be checked. We plan to study the use of stratified sampling in our model checking algorithm.

We also plan to investigate the application of our framework to the analysis of biological

systems. Some quantitative extensions of temporal logics have already been proposed in that setting (e.g. [32, 33, 34]) to capture the notion of robustness from [35] or similar proposals [36]. It would be interesting to see whether the use of DisTL and evolution sequences can lead to new results in this setting.

Acknowledgments

This work has been partially supported by the project *Programs in the wild: uncertainties, adaptability and verification* (ULTRON) of the Icelandic Research Fund (grant No. 228376-051). 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.

References

[1] R. Rajkumar, I. Lee, L. Sha, J. A. Stankovic, Cyber-physical systems: the next computing revolution, in: DAC, ACM, 2010, pp. 731–736.

[2] V. Castiglioni, M. Loreti, S. Tini, How adaptive and reliable is your program?, in: Proceedings of FORTE 2021, volume 12719 of *Lecture Notes in Computer Science*, 2021, pp. 60–79. doi:10.1007/978-3-030-78089-0_4.

[3] V. Castiglioni, M. Loreti, S. Tini, A framework to measure the robustness of programs in the unpredictable environment, Log. Methods Comput. Sci. 19 (2023). doi:10.46298/lmcs-19(3:2)2023.

[4] L. de Alfaro, T. A. Henzinger, R. Majumdar, Discounting the Future in Systems Theory, in: Proceedings of ICALP'03, ICALP '03, Springer, 2003, pp. 1022–1037. doi:10.1007/3-540-45061-0_79.

[5] J. Desharnais, V. Gupta, R. Jagadeesan, P. Panangaden, Metrics for labelled Markov processes, Theor. Comput. Sci. 318 (2004) 323–354. doi:10.1016/j.tcs.2003.09.013.

[6] V. Castiglioni, M. Loreti, S. Tini, The metric linear-time branching-time spectrum on nondeterministic probabilistic processes, Theor. Comput. Sci. 813 (2020) 20–69. doi:10.1016/j.tcs.2019.09.019.

[7] V. Castiglioni, M. Loreti, S. Tini, RobTL: A temporal logic for the robustness of cyber-physical systems, CoRR abs/2212.11158 (2022). doi:10.48550/arXiv.2212.11158. arXiv:2212.11158.

[8] M. Fränzle, J. Kapinski, P. Prabhakar, Robustness in cyber-physical systems, Dagstuhl Reports 6 (2016) 29–45. doi:10.4230/DagRep.6.9.29.

[9] M. Rungger, P. Tabuada, A notion of robustness for cyber-physical systems, IEEE Trans. Autom. Control. 61 (2016) 2108–2123. doi:10.1109/TAC.2015.2492438.

[10] A. Shahrokni, R. Feldt, A systematic review of software robustness, Information and Software Technology 55 (2013) 1–17. doi:https://doi.org/10.1016/j.infsof.2012.06.002.

[11] E. D. Sontag, Input to State Stability: Basic Concepts and Results, Springer Berlin Heidelberg, 2008, pp. 163–220. doi:10.1007/978-3-540-77653-6_3.

[12] V. Castiglioni, M. Loreti, S. 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*, 2023, pp. 115–132. doi:10.1007/978-3-031-35361-1_6.
[13] H. Hansson, B. Jonsson, A logic for reasoning about time and reliability, Formal Asp. Comput. 6 (1994) 512–535. doi:10.1007/BF01211866.
[14] A. Aziz, K. Sanwal, V. Singhal, R. K. Brayton, Model-checking continous-time Markov chains, ACM Trans. Comput. Log. 1 (2000) 162–170. doi:10.1145/343369.343402.
[15] A. Aziz, K. Sanwal, V. Singhal, R. K. Brayton, Verifying continuous time Markov chains, in: Proceedings of CAV '96, volume 1102 of *Lecture Notes in Computer Science*, 1996, pp. 269–276. doi:10.1007/3-540-61474-5_75.
[16] A. Pnueli, The temporal logic of programs, in: Proceedings of FOCS 1977, IEEE Computer Society, 1977, pp. 46–57. doi:10.1109/SFCS.1977.32.
[17] M. Tiger, F. Heintz, Stream reasoning using temporal logic and predictive probabilistic state models, in: Proceedings of TIME 2016, 2016, pp. 196–205. doi:10.1109/TIME.2016.28.
[18] D. Sadigh, A. Kapoor, Safe control under uncertainty with probabilistic signal temporal logic, in: Proceedings of Robotics: Science and Systems XII 2016, 2016. doi:10.15607/RSS.2016.XII.017.
[19] R. Koymans, Specifying real-time properties with metric temporal logic, Real Time Syst. 2 (1990) 255–299. doi:10.1007/BF01995674.
[20] O. Maler, D. Nickovic, Monitoring temporal properties of continuous signals, in: Proceedings of FORMATS and FTRTFT 2004, volume 3253 of *Lecture Notes in Computer Science*, 2004, pp. 152–166. doi:10.1007/978-3-540-30206-3_12.
[21] L. N. Vaserstein, Markovian processes on countable space product describing large systems of automata., Probl. Peredachi Inf. 5 (1969) 64–72.
[22] V. I. Bogachev, Measure Theory, number v. 1 in Measure Theory, Springer-Verlag, Berlin/Heidelberg, 2007. doi:10.1007/978-3-540-34514-5.
[23] O. P. Faugeras, L. Rüschendorf, Risk excess measures induced by hemi-metrics, Probability, Uncertainty and Quantitative Risk 3:6 (2018). doi:10.1186/s41546-018-0032-0.
[24] A. Donzé, O. Maler, Robust satisfaction of temporal logic over real-valued signals, in: Proceedings of FORMATS 2010, volume 6246 of *Lecture Notes in Computer Science*, 2010, pp. 92–106. doi:10.1007/978-3-642-15297-9_9.
[25] D. Thorsley, E. Klavins, Approximating stochastic biochemical processes with Wasserstein pseudometrics, IET Syst. Biol. 4 (2010) 193–211. doi:10.1049/iet-syb.2009.0039.
[26] M. Tiger, F. Heintz, Incremental reasoning in probabilistic signal temporal logic, Int. J. Approx. Reason. 119 (2020) 325–352. doi:10.1016/j.ijar.2020.01.009.
[27] D. Phan, N. Paoletti, T. Zhang, R. Grosu, S. A. Smolka, S. D. Stoller, Neural state classification for hybrid systems, in: Proceedings of ATVA 2018, volume 11138 of *Lecture Notes in Computer Science*, 2018, pp. 422–440. doi:10.1007/978-3-030-01090-4_25.
[28] L. Bortolussi, F. Cairoli, N. Paoletti, S. A. Smolka, S. D. Stoller, Neural predictive monitoring, in: Proceedings of RV 2019, volume 11757 of *Lecture Notes in Computer Science*, 2019, pp. 129–147. doi:10.1007/978-3-030-32079-9_8.
[29] Y. Kwon, G. Agha, Linear inequality LTL (iltl): A model checker for discrete time markov chains, in: Proceedings of ICFEM 2004, volume 3308 of *Lecture Notes in Computer Science*, 2004, pp. 194–208. doi:10.1007/978-3-540-30482-1_21.
[30] V. A. Korthikanti, M. Viswanathan, G. Agha, Y. Kwon, Reasoning about mdps as transform-

ers of probability distributions, in: Proceedings of QEST 2010, IEEE Computer Society, 2010, pp. 199–208. doi:10.1109/QEST.2010.35.

[31] Y. Wang, N. Roohi, M. West, M. Viswanathan, G. E. Dullerud, Statistical verification of PCTL using antithetic and stratified samples, Formal Methods Syst. Des. 54 (2019) 145–163. doi:10.1007/s10703-019-00339-8.

[32] F. Fages, A. Rizk, On temporal logic constraint solving for analyzing numerical data time series, Theor. Comput. Sci. 408 (2008) 55–65. doi:10.1016/j.tcs.2008.07.004.

[33] A. Rizk, G. Batt, F. Fages, S. Soliman, A general computational method for robustness analysis with applications to synthetic gene networks, Bioinform. 25 (2009). doi:10.1093/bioinformatics/btp200.

[34] A. Rizk, G. Batt, F. Fages, S. Soliman, Continuous valuations of temporal logic specifications with applications to parameter optimization and robustness measures, Theor. Comput. Sci. 412 (2011) 2827–2839. doi:10.1016/j.tcs.2010.05.008.

[35] H. Kitano, Towards a theory of biological robustness, Molecular Systems Biology 3 (2007) 137. doi:https://doi.org/10.1038/msb4100179. arXiv:https://www.embopress.org/doi/pdf/10.1038/msb4100179.

[36] L. Nasti, R. Gori, P. Milazzo, Formalizing a notion of concentration robustness for biochemical networks, in: Proceedings of STAF 2018, volume 11176 of *Lecture Notes in Computer Science*, 2018, pp. 81–97. doi:10.1007/978-3-030-04771-9_8.