How Adaptive and Reliable is your Program?

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. We consider the problem of *modelling* and *verifying* the behaviour of systems characterised by a close interaction of a *program* with the *environment*. We propose to model the program-environment interplay in terms of the probabilistic modifications they induce on a set of application-relevant data, called *data space*. The behaviour of a system is thus identified with the probabilistic evolution of the initial data space. Then, we introduce a metric, called *evolution metric*, measuring the differences in the *evolution sequences* of systems and that can be used for system verification as it allows for expressing how well the program is fulfilling its tasks. We use the metric to express the properties of *adaptability* and *reliability* of a program, which allow us to identify potential critical issues of it w.r.t. changes in the initial environmental conditions. We also propose an *algorithm*, based on statistical inference, for the evaluation of the evolution metric.

1 Introduction

With the ever-increasing complexity of the digital world and the massive diffusion of IoT systems [13], cyber-physical systems [17], and smart devices, we are progressively witnessing the rise of a new class of software applications, henceforth *programs*, that must be able to deal with highly changing operational conditions, henceforth *environment*. Examples of this kind of programs are the software components of unmanned vehicles, (on-line) service applications, the devices in a smart house, etc, which have to interact with other programs and heterogeneous devices, and with physical phenomena like wind, temperature, etc. In what follows, we will use the term *system* to denote the combination of the environment and the program acting on it. Hence, the *behaviour* of a system is the result of the program-environment interplay.

The main challenge in the analysis and verification of these systems is then the dynamical and, sometimes, unpredictable behaviour of the environment. The highly dynamical behaviour of physical processes can only be approximated in order to become computationally tractable and it can constitute a safety hazard for the devices in the system (like, e.g., an unexpected gust of wind for a drone that is autonomously setting its trajectory to avoid obstacles); some devices or programs may appear, disappear, or become temporarily unavailable; faults or conflicts may occur (like, e.g., in a smart home the program responsible for the ventilation of a room may open a window causing a conflict with the program that has to limit the noise level); sensors may introduce some measurement errors; etc. The introduction of *uncertainties* and *approximations* in these systems is therefore inevitable.

In the literature, we can find a wealth of proposals of stochastic and probabilistic models, as, e.g., *Stochastic Hybrid Systems* [5,12] and *Markov Decision Processes* [15], and *ad hoc* solutions for specific application contexts, as, e.g., establishing safety guarantees for drones flying under particular circumstances [10, 24]. Yet, in these studies, either the environment is not explicitly taken into account or it is modelled only deterministically. In addition to that, due to the variety of applications and heterogeneity of systems, no general formal framework to deal with these challenges has been proposed so far. The lack of concise abstractions and of an automatic support makes the analysis and verification of the considered systems difficult, laborious, and error prone.

Our contribution. With this paper we aim at taking a first step towards a solution of the above-mentioned challenges, with a special focus on *verification*, by developing the tools for the verification of the ability of programs to *adjust* their behaviour to the unpredictable environment. Formally, we introduce two *measures* allowing us to assess how well a given program can perform under perturbations in the environmental conditions. We call these two measures *adaptability* and *reliability*. As an example, consider a drone that is autonomously flying along a given trajectory. In this setting, a perturbation can be given by a gust of wind that moves the drone out of its trajectory. We will say that the program controlling the drone is *adaptable* if it can retrieve the initial trajectory within a suitable amount of time. In other words, we say that a program is adaptable if no matter how much its behaviour is affected by the perturbations, it is able to react to them and regain its intended behaviour within a given amount of time. On the other hand, it may be the case that the drone is able to detect the presence of a gust of wind and can oppose to it, being only slightly moved from its initial trajectory. In this case, we say that the program controlling the drone is *reliable*. Hence, *reliability* expresses the ability of a program to maintain its intended behaviour (up-to some reasonable tolerance) despite the presence of perturbations in the environment.

In order to measure the adaptability and reliability of a program, we need to be able to express *how well* it is fulfilling its tasks. The systems that we are considering are strongly characterised by a quantitative behaviour, given by both the presence of uncertainties and the data used by program and environment. It seems then natural, and reasonable, to quantify the differences in the behaviour of systems by means of a metric over data. However, in order to informally discuss our proposal of a *metric semantics* for the kind of systems that we are considering, we need first to explain how the behaviour of these systems is defined, namely to introduce our general formal model for them. Our idea is to favour the modelling of the program-environment interplay over precise specifications of the operational behaviour of a program. In the last decade, many researchers have focused their studies on formal models capturing both the *qualitative* and *quantitative* behaviour of systems: Probabilistic Automata [19], Stochastic Process Algebras [3, 6, 11], Labelled Markov Chains and Stochastic Hybrid Systems [5, 12]. A common feature of these models is that the (quantitative, labelled) transitions expressing the computation steps directly model the behaviour of the system as a whole.

In this paper we take a ***different point of view***: we propose to model the behaviour of program and environment separately, and then explicitly represent their *interaction* in a purely *data-driven* fashion. In fact, while the environmental conditions are (partially) available to the program as a set of data, allowing it to adjust its behaviour to the current situation, the program is also able to use data to (partially) control the environment and fulfil its tasks. It is then natural to model the program-environment interplay in terms of the changes they induce on a set of application-relevant data, henceforth referred to as the *data space*. This feature will allow for a significant simplification in modelling the behaviour of the program, which can be *isolated* from that of the environment, thus favouring its analysis. Moreover, as common practice to favour *computational tractability* [1, 2], we adopt a *discrete time* approach.

We can then study the behaviour of the system as a whole by analysing how data evolve in time. In our model, a *system* consists in *three distinct components*: 1. a *process* P describing the behaviour of the program, 2. a *data state* $\mathbf{d}$ describing the current state of the data space, and 3. an *environment evolution* $\mathcal{E}$ describing the effect of the environment on $\mathbf{d}$. As we focus on the interaction with the environment, we abstract from the internal computation of the program and model only its activity on $\mathbf{d}$. At each step, a process can *read/update* values in $\mathbf{d}$ and $\mathcal{E}$ applies on the resulting data state, providing a new data state at the next step. To deal with the uncertainties, we introduce *probability* at two levels: (i) we use the ***discrete*** *generative probabilistic model* [9] to define processes, and (ii) $\mathcal{E}$ induces a ***continuous*** *distribution* over data states. The behaviour of the system is then entirely expressed by its *evolution sequence*, i.e., the sequence of distributions over data states obtained at each step. Given the novelties of our model, as a side contribution we show that this behaviour defines a *Markov process*.

It is now reasonable to define our *metric semantics* in terms of a (time-dependent) distance on the evolution sequences of systems, which we call the *evolution metric*. The *evolution metric* will allow us to: 1. verify how well a program is fulfilling its tasks by comparing it with its specification, 2. compare the activity of different programs in the same environment, 3. compare the behaviour of one program w.r.t. different environments and changes in the initial conditions. The third feature will allow us to measure the *adaptability* and *reliability* of programs.

The evolution metric will consist of two components: a *metric on data states* and the *Wasserstein metric* [22]. The former is defined in terms of a (time-dependent) *penalty function* allowing us to compare two data states only on the base of the objectives of the program (which can be in turn expressed in terms of data). The latter lifts the metric on data states to a metric on distributions on data states. We then obtain a metric on evolution sequences by considering the maximal of the Wasserstein distances over time. We provide an *algorithm* for the estimation of the evolution sequences of systems and thus for the evaluation of the evolution metric. Following [21], the Wasserstein metric is evaluated in time $O(N \log N)$, where N is the (maximum) number of samples.

As an example of application of our framework, we use it to model a simple smart-room scenario. We consider two programs: the thermostat of an heating system, and an air quality controller. The former has to keep the room temperature within a desired comfort interval. The latter has to keep the quality of the air above a given threshold. We use our algorithm to evaluate the differences between two systems having the same programs but starting from different initial conditions. Finally, we apply it to measure the adaptability and reliability of the considered programs.

Due to space limitations, the proofs and the simplest parts of the algorithm have been moved to the Appendix.

2 Background

Measurable spaces. A *σ-algebra* over a set Ω is a family Σ of subsets of Ω s.t. $\Omega \in \Sigma$, and Σ is closed under complementation and under countable union. The pair (Ω, Σ) is called a *measurable space* and the sets in Σ are called *measurable sets*, ranged over by $\mathbb{A}, \mathbb{B}, \dots$. For an arbitrary family Φ of subsets of Ω, the σ-algebra *generated* by Φ is the smallest σ-algebra over Ω containing Φ. In particular, we recall that given a topology T over Ω, the *Borel σ-algebra* over Ω, denoted $\mathcal{B}(\Omega)$, is the σ-algebra generated be the open sets in T. Given two measurable spaces (Ω_i, Σ_i), $i = 1, 2$, the *product σ-algebra* $\Sigma_1 \times \Sigma_2$ is the σ-algebra on $\Omega_1 \times \Omega_2$ generated by the sets $\{\mathbb{A}_1 \times \mathbb{A}_2 \mid \mathbb{A}_i \in \Sigma_i\}$.

Given measurable spaces $(\Omega_1, \Sigma_1), (\Omega_2, \Sigma_2)$, a function $f \colon \Omega_1 \to \Omega_2$ is said to be Σ_1-*measurable* if $f^{-1}(\mathbb{A}_2) \in \Sigma_1$ for all $\mathbb{A}_2 \in \Sigma_2$, with $f^{-1}(\mathbb{A}_2) = \{\omega \in \Omega_1 | f(\omega) \in \mathbb{A}_2\}$.

Probability spaces. A *probability measure* on a measurable space (Ω, Σ) is a function $\mu \colon \Sigma \to [0, 1]$ such that: i) $\mu(\Omega) = 1$, ii) $\mu(\mathbb{A}) \geq 0$ for all $\mathbb{A} \in \Sigma$, and iii) $\mu(\bigcup_{i \in I} \mathbb{A}_i) = \sum_{i \in I} \mu(\mathbb{A}_i)$ for every countable family of pairwise disjoint measurable sets $\{\mathbb{A}_i\}_{i \in I} \subseteq \Sigma$. Then (Ω, Σ, μ) is called a *probability space*.

Notation *With a slight abuse of terminology, we shall henceforth use the term* distribution *in place of the term probability measure.*

We let $\Delta(\Omega, \Sigma)$ denote the set of all distributions over (Ω, Σ). For $\omega \in \Omega$, the *Dirac distribution* δ_ω is defined by $\delta_\omega(\mathbb{A}) = 1$, if $\omega \in \mathbb{A}$, and $\delta_\omega(\mathbb{A}) = 0$, otherwise, for all $\mathbb{A} \in \Sigma$. For a countable set of reals $(p_i)_{i \in I}$ with $p_i \geq 0$ and $\sum_{i \in I} p_i = 1$, the *convex combination* of the distributions $\{\mu_i\}_{i \in I} \subseteq \Delta(\Omega, \Sigma)$ is the distribution $\sum_{i \in I} p_i \cdot \mu_i$ in $\Delta(\Omega, \Sigma)$ defined by $(\sum_{i \in I} p_i \cdot \mu_i)(\mathbb{A}) = \sum_{i \in I} p_i \mu_i(\mathbb{A})$, for all $\mathbb{A} \in \Sigma$. A distribution $\mu \in \Delta(\Omega, \Sigma)$ is called *discrete* if $\mu = \sum_{i \in I} p_i \cdot \delta_{\omega_i}$, with $\omega_i \in \Omega$, for some countable set of indexes I. In this case, the *support* of μ is $\mathsf{supp}(\mu) = \{\omega_i \mid i \in I\}$.

The Wasserstein hemimetric. A *metric* on a set Ω is a function $m \colon \Omega \times \Omega \to \mathbb{R}^{\geq 0}$ s.t. $m(\omega_1, \omega_2) = 0$ iff $\omega_1 = \omega_2$, $m(\omega_1, \omega_2) = m(\omega_2, \omega_1)$, and $m(\omega_1, \omega_2) \leq m(\omega_1, \omega_3) + m(\omega_3, \omega_2)$, for all $\omega_1, \omega_2, \omega_3 \in \Omega$. We obtain a *hemimetric* by relaxing the first property to $m(\omega_1, \omega_2) = 0$ if $\omega_1 = \omega_2$, and by dropping the requirement on symmetry. A (hemi)metric m is *l-bounded* if $m(\omega_1, \omega_2) \leq l$ for all $\omega_1, \omega_2 \in \Omega$. For a (hemi)metric on Ω, the pair (Ω, m) is a *(hemi)metric space*.

In order to define a *hemimetric on distributions* we use the Wasserstein lifting [22]. We recall that a *Polish space* is a separable completely metrisable topological space.

Definition 1 (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*

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

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 Wasserstein distance was originally given on metrics, the Wasserstein hemimetric given above is well-defined. A formal proof of this can be found in [7] and the references therein. We refer the interested reader to Appendix A for more details.

Notation *As elsewhere in the literature, we use the term* metric *in place of hemimetric.*

3 The model

In this section, we introduce the three components of our systems, namely the *data space*, the *process* describing the behaviour of the program, and the *environment evolution* describing the effects of the environment. The following example perfectly embodies the kind of program-environment interactions we are interested in.

Example 1. We consider a *smart-room scenario* in which the program should guarantee that both the *temperature* and the *air quality* in the room are in a given *comfort zone*. The room is equipped with a *heating system* and an *air filtering system*. Both are equipped with a *sensor* and an *actuator*. In the heating system the sensor is a thermometer that reads the room temperature, while the actuator is used to turn the heater on or off. Similarly, in the air filtering system the sensor perceives the air quality, giving a value in $[0,1]$, while the actuator activates the air exchangers. The environment models the evolution of temperature and air quality in the room, as described by the following stochastic difference equations, with sample time interval $\varDelta\tau = 1$:

$$T(\tau+1) = T(\tau) + a(e(\tau)) \cdot (T_e - T(\tau)) + h(\tau) \cdot b \cdot (T_h - T(\tau)) \tag{1}$$

$$T_s(\tau) = T(\tau) + n_t(\tau) \tag{2}$$

$$A(\tau+1) = A(\tau) + e(\tau) \cdot q^+ \cdot (1 - A(\tau)) - (1 - e(\tau)) \cdot q^- \cdot A(\tau) \tag{3}$$

$$A_s(\tau) = A(\tau) + n_a(\tau) \tag{4}$$

Above, $T(\tau)$ and $A(\tau)$ are the room temperature and air quality at time τ, while $T_s(\tau)$ and $A_s(\tau)$ are the respective values read by sensors, which are obtained from the real ones by adding *noises* $n_t(\tau)$ and $n_a(\tau)$, that we assume to be distributed as Gaussian (normal) distributions $\mathcal{N}(0, \upsilon_t^2)$ and $\mathcal{N}(0, \upsilon_a^2)$, resp., for some suitable υ_t^2 and υ_a^2. Then, $h(\tau)$ and $e(\tau)$ represent the state of the actuators of the heating and air filtering system, respectively. Both take value 1 when the actuator is *on*, and 0 otherwise. Following [2,8,14], the temperature dynamics depends on two (non negative) values, $a(e(\tau))$ and b, giving the average heat transfer rates normalised w.r.t. the thermal capacity of the room. In detail, $a(e(\tau))$ is the heat loss rate from the room (through walls, windows, etc.) to the external ambient for which we assume a constant temperature T_e. In our case, this value depends on $e(\tau)$, since the loss rate increases when the air exchangers are on. Then, b is the heat transfer rate from the heater, whose temperature is the constant T_h, to the room. The air quality dynamics is similar: when the air exchangers are off, the air quality decreases with a rate q^-, while it increases of a rate q^+ when they are on.

Modelling the data space. We define the data space by means of a *finite* set of *variables* Var representing: i) *environmental conditions* (pressure, temperature, humidity, etc.,);

ii) *values perceived by sensors* (unavoidably affected by imprecision and approximations); iii) *state of actuators* (usually elements in a discrete domain). For each $x \in \mathrm{Var}$ we assume a measurable space $(\mathcal{D}_x, \mathcal{B}_x)$, with $\mathcal{D}_x \subseteq \mathbb{R}$ the domain of x and $\mathcal{B}_x$ the Borel σ-algebra on $\mathcal{D}_x$. Without loosing generality, we can assume that $\mathcal{D}_x$ is either a *finite set* or a *compact* subset of $\mathbb{R}$. Notably, $\mathcal{D}_x$ is a Polish space. As Var is a finite set, we can always assume it to be ordered, i.e., $\mathrm{Var} = \{x_1, \dots, x_n\}$ for some $n \in \mathbb{N}$.

Definition 2 (Data space). *We define the* data space *over* Var, *notation* $\mathcal{D}_{\mathrm{Var}}$, *as the Cartesian product of the variables domains, namely* $\mathcal{D}_{\mathrm{Var}} = \bigtimes_{i=1}^{n} \mathcal{D}_{x_i}$. *Then, as a* σ*-algebra on* $\mathcal{D}_{\mathrm{Var}}$ *we consider the the product* σ*-algebra* $\mathcal{B}_{\mathcal{D}_{\mathrm{Var}}} = \bigtimes_{i=1}^{n} \mathcal{B}_{x_i}$.

Example 2. The data space for the system in Example 1 is defined on the variables T, T_s, h, A, A_s and e. Their domains are $\mathcal{D}_T = \mathcal{D}_{T_s} = [t_m, t_M]$, for suitable values $t_m < t_M$, $\mathcal{D}_A = \mathcal{D}_{A_s} = [0, 1]$, and $\mathcal{D}_h = \mathcal{D}_e = \{0, 1\}$.

When no confusion arises, we will use $\mathcal{D}$ and $\mathcal{B}_{\mathcal{D}}$ in place of $\mathcal{D}_{\mathrm{Var}}$ and $\mathcal{B}_{\mathcal{D}_{\mathrm{Var}}}$, respectively. The elements in $\mathcal{D}$ are the n-ples of the form $(v_1, \dots, v_n)$, with $v_i \in \mathcal{D}_{x_i}$, which can be also identified by means of functions $\mathbf{d}\colon \mathrm{Var} \to \mathbb{R}$ from variables to values, with $\mathbf{d}(x) \in \mathcal{D}_x$ for all $x \in \mathrm{Var}$. Each function $\mathbf{d}$ identifies a particular configuration of the data in the data space, and it is thus called a *data state*.

Definition 3 (Data state). *A* data state *is a mapping* $\mathbf{d}\colon \mathrm{Var} \to \mathbb{R}$ *from state variables to values, with* $\mathbf{d}(x) \in \mathcal{D}_x$ *for all* $x \in \mathrm{Var}$.

For simplicity, we shall write $\mathbf{d} \in \mathcal{D}$ in place of $(\mathbf{d}(x_1), \dots, \mathbf{d}(x_n)) \in \mathcal{D}$. Since program and environment interact on the basis of the *current* values of data, we have that at each step there is a data state $\mathbf{d}$ that identifies the *current state of the data space* on which the next computation step is built. Given a data state $\mathbf{d}$, we let $\mathbf{d}[x = v]$ denote the data state $\mathbf{d}'$ associating v with x, and $\mathbf{d}(y)$ with any $y \neq x$.

Modelling processes. We introduce a simple process calculus allowing us to specify programs that interact with a data state $\mathbf{d}$ in a given environment. We assume that the action performed by a process at a given computation step is determined probabilistically, according to the *generative* probabilistic model [9].

Definition 4 (Syntax of processes). *We let* $\mathcal{P}$ *be the set of* processes P *defined by:*

$$\begin{aligned} P &::= (\overline{e} \to \overline{x}).P' \;\mid\; \text{if } [e]\ P_1 \text{ else } P_2 \;\mid\; \textstyle\sum_{i \in I} p_i \cdot P_i \;\mid\; P_1 \|_p P_2 \;\mid\; A \\ e &::= v \in \mathbb{R} \;\mid\; x \in \mathrm{Var} \;\mid\; op_k(e_1, \dots, e_k) \end{aligned}$$

where $p, p_1, \dots$ *range over* probability weights *in* $[0, 1]$, I *is finite,* A *ranges over* process variables, op_k *indicates a* measurable operator $\mathbb{R}^k \to \mathbb{R}$, *and* $\bar{\cdot}$ *denotes a finite sequence of elements. We assume to have a single definition* $A \stackrel{def}{=} P$ *for each process variable* A. *Moreover, we require that* $\sum_{i \in I} p_i = 1$ *for any process* $\sum_{i \in I} p_i \cdot P_i$.

Process $(\overline{e} \to \overline{x}).P$ evaluates the sequence of expressions $\overline{e}$ with the current data state $\mathbf{d}$ and assigns the results $[\![\overline{e}]\!]_{\mathbf{d}}$ to the sequence of variables $\overline{x}$. We may use $\surd$ to denote the prefix $(\emptyset \to \emptyset)$. Process if $[e]$ P_1 else P_2 behaves either as P_1 when $[\![e]\!]_{\mathbf{d}} = \top$, or as P_2 when $[\![e]\!]_{\mathbf{d}} = \bot$. Then, $\sum_{i=1}^{n} p_i \cdot P_i$ is the *generative probabilistic choice*: process P_i has probability p_i to move. The *generative probabilistic interleaving* construct $P_1 \|_p P_2$ lets the two argument processes to interleave their actions, where at each step P_1 moves with probability p and P_2 with probability $1-p$. Process variables allow us to specify recursive behaviours by means of equations of the form $A \overset{def}{=} P$. To avoid Zeno behaviours we assume that all occurrences of process variables appear *guarded* by prefixing constructs in P. We assume the standard notions of *free* and *bound* process variables. A program is then a *closed* process, i.e., a process without free variables.

Formally, actions performed by a process can be abstracted in terms of the *effects* they have on the data state, i.e., via *substitutions* of the form $\theta = [x_{i_1} \leftarrow v_{i_1}, \dots, x_{i_k} \leftarrow v_{i_k}]$, also denoted $\overline{x} \leftarrow \overline{v}$ if $\overline{x} = x_{i_1}, \dots, x_{i_k}$ and $\overline{v} = v_{i_1}, \dots, v_{i_k}$. Since in Definition 4 operations op_k are assumed to be measurable, we can model the effects as $\mathcal{B}_{\mathcal{D}}$-measurable functions $\theta \colon \mathcal{D} \to \mathcal{D}$ s.t. $\theta(\mathbf{d}) := \mathbf{d}[\overline{x} = \overline{v}]$ whenever $\theta = \overline{x} \leftarrow \overline{v}$. We denote by Θ the set of effects. The behaviour of a process can then be defined by means of a function $\mathsf{pstep} \colon \mathcal{P} \times \mathcal{D} \to \Delta(\Theta \times \mathcal{P})$ that given a process P and a data state $\mathbf{d}$ yields a *discrete* distribution over $\Theta \times \mathcal{P}$. Function pstep is defined as follows:

$$\text{(PR1)}\quad \mathsf{pstep}((\overline{e} \to \overline{x}).P', \mathbf{d}) = \delta_{(\overline{x} \leftarrow [\![\overline{e}]\!]_{\mathbf{d}}, P')}$$

$$\text{(PR2)}\quad \mathsf{pstep}(\text{if } [e]\ P_1 \text{ else } P_2, \mathbf{d}) = \begin{cases} \mathsf{pstep}(P_1, \mathbf{d}) & \text{if } [\![e]\!]_{\mathbf{d}} = 1 \\ \mathsf{pstep}(P_2, \mathbf{d}) & \text{if } [\![e]\!]_{\mathbf{d}} = 0 \end{cases}$$

$$\text{(PR3)}\quad \mathsf{pstep}(\textstyle\sum_i p_i \cdot P_i, \mathbf{d}) = \sum_i p_i \cdot \mathsf{pstep}(P_i, \mathbf{d})$$

$$\text{(PR4)}\quad \mathsf{pstep}(P_1 \|_p P_2, \mathbf{d}) = p \cdot (\mathsf{pstep}(P_1, \mathbf{d}) \|_p P_2) + (1-p) \cdot (P_1 \|_p \mathsf{pstep}(P_2, \mathbf{d}))$$

$$\text{(PR5)}\quad \mathsf{pstep}(A, \mathbf{d}) = \mathsf{pstep}(P, \mathbf{d}) \qquad (\text{if } A \overset{def}{=} P).$$

In rule (PR4), for $\pi \in \Delta(\Theta \times \mathcal{P})$, we let $\pi \|_p P$ (resp. $P \|_p \pi$) denote the distribution $\pi' \in \Delta(\Theta \times \mathcal{P})$ s.t.: $\pi'(\theta, P') = \pi(\theta, P'')$, whenever $P' = P'' \|_p P$ (resp. $P' = P \|_p P''$), and 0, otherwise.

Proposition 1 (Properties of process semantics). *Let $P \in \mathcal{P}$ and $\mathbf{d} \in \mathcal{D}$. Then* $\mathsf{pstep}(P, \mathbf{d})$ *is a discrete distribution with finite support.*

Example 3. We define a program to control the smart-room scenario of Example 1. In detail, we want to guarantee that the temperature in the room is in the interval $Z = [t_{\min}, t_{\max}] \subseteq \mathcal{D}_T$, while the air quality is above a given threshold $q_a \in \mathcal{D}_A$. The following process $\mathsf{A}^T_{\mathit{off}}$ (resp. A^T_{on}) turns the heating system *on* (resp. *off*) when the temperature acquired by the sensor goes under $t_{\min} - \varepsilon_t$ (resp. over $t_{\max} + \varepsilon_t$). The use of the tolerance ε_t guarantees that the heating system is not repeatedly turned on/off.

$$\begin{aligned} \mathsf{A}^T_{\mathit{off}} &\overset{def}{=} \text{if } [T_s < t_{\min} - \varepsilon_t]\ (1 \to h).\mathsf{A}^T_{on} \text{ else } \surd.\mathsf{A}^T_{\mathit{off}} \\ \mathsf{A}^T_{on} &\overset{def}{=} \text{if } [T_s > t_{\max} + \varepsilon_t]\ (0 \to h).\mathsf{A}^T_{\mathit{off}} \text{ else } \surd.\mathsf{A}^T_{on}. \end{aligned}$$

The behaviour of program components controlling the air filtering system is similar and implemented by the following processes $\mathsf{A}^A_{\mathit{off}}$ and A^A_{on}:

$$\begin{aligned} \mathsf{A}^A_{\mathit{off}} &\overset{def}{=} \text{if } [A_s \leq q_a - \varepsilon_a]\ (1 \to e).\mathsf{A}^A_{on} \text{ else } \surd.\mathsf{A}^A_{\mathit{off}} \\ \mathsf{A}^A_{on} &\overset{def}{=} \text{if } [A_s > q_a + \varepsilon_a]\ (0 \to e).\mathsf{A}^A_{\mathit{off}} \text{ else } \surd.\mathsf{A}^A_{on}. \end{aligned}$$

The composition of the programs is given by the process $\mathsf{P} = \mathsf{A}^{T}_{\mathit{off}} \parallel_{0.5} \mathsf{A}^{A}_{\mathit{off}}$.

Modelling the environment. We model the action of the environment by a mapping $\mathcal{E}$, called *environment evolution*, taking a data state to a distribution over data states.

Definition 5 (Environment evolution). *An* environment evolution *is a function $\mathcal{E}\colon \mathcal{D} \to \Delta(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$ s.t. for each $\mathbb{D} \in \mathcal{B}_{\mathcal{D}}$ the mapping $\mathbf{d} \mapsto \mathcal{E}(\mathbf{d})(\mathbb{D})$ is $\mathcal{B}_{\mathcal{D}}$-measurable.*

Due to the interaction with the program, the probability induced by $\mathcal{E}$ at the next time step *depends only* on the current state of the data space. It is then natural to assume that the behaviour of the environment is modelled as a discrete time Markov process.

Example 4. For our smart-room scenario, the environment evolution $\mathcal{E}$ can be derived directly from Equations (1)–(4). Notice that, in this case, randomness follows from the Gaussian noises associated with the temperature and air quality sensors.

Modelling system's behaviour. We use the notion of *configuration* to model the state of the system at each time step.

Definition 6 (Configuration). *A* configuration *is a triple $c = \langle P, \mathbf{d}\rangle_{\mathcal{E}}$, where P is a process, $\mathbf{d}$ is a data state and $\mathcal{E}$ is an environment evolution. We denote by $\mathcal{C}_{\mathcal{P},\mathcal{D},\mathcal{E}}$ the set of configurations defined over $\mathcal{P}, \mathcal{D}$ and $\mathcal{E}$.*

When no confusion arises, we shall write $\mathcal{C}$ in place of $\mathcal{C}_{\mathcal{P},\mathcal{D},\mathcal{E}}$.

Let $(\mathcal{P}, \Sigma_{\mathcal{P}})$ be the measurable space of processes, where $\Sigma_{\mathcal{P}}$ is the power set of $\mathcal{P}$, and $(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$ be the measurable space of data states. As $\mathcal{E}$ is fixed, we can identify $\mathcal{C}$ with $\mathcal{P} \times \mathcal{D}$ and equip it with the product σ-algebra $\Sigma_{\mathcal{C}} = \Sigma_{\mathcal{P}} \times \mathcal{B}_{\mathcal{D}}$: $\Sigma_{\mathcal{C}}$ is generated by the sets $\{\langle \mathbb{P}, \mathbb{D}\rangle_{\mathcal{E}} \mid \mathbb{P} \in \Sigma_{\mathcal{P}}, \mathbb{D} \in \mathcal{B}_{\mathcal{D}}\}$, where $\langle \mathbb{P}, \mathbb{D}\rangle_{\mathcal{E}} = \{\langle P, \mathbf{d}\rangle_{\mathcal{E}} \mid P \in \mathbb{P}, \mathbf{d} \in \mathbb{D}\}$.

Notation *For $\mu_{\mathcal{P}} \in \Delta(\mathcal{P}, \Sigma_{\mathcal{P}})$ and $\mu_{\mathcal{D}} \in \Delta(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$ we let $\mu = \langle \mu_{\mathcal{P}}, \mu_{\mathcal{D}}\rangle_{\mathcal{E}}$ denote the product distribution on $(\mathcal{C}, \Sigma_{\mathcal{C}})$, i.e., $\mu(\langle \mathbb{P}, \mathbb{D}\rangle_{\mathcal{E}}) = \mu_{\mathcal{P}}(\mathbb{P}) \cdot \mu_{\mathcal{D}}(\mathbb{D})$ for all $\mathbb{P} \in \Sigma_{\mathcal{P}}$ and $\mathbb{D} \in \mathcal{B}_{\mathcal{D}}$. If $\mu_{\mathcal{P}} = \delta_P$ for some $P \in \mathcal{P}$, we shall denote $\langle \delta_P, \mu_{\mathcal{D}}\rangle_{\mathcal{E}}$ simply by $\langle P, \mu_{\mathcal{D}}\rangle_{\mathcal{E}}$.*

We aim to express the behaviour of a system in terms of the changes on data. We start with the *one-step* behaviour of a configuration, in which we combine the effects on the data state induced by the activity of the process (given by pstep) and the subsequent action by the environment. Formally, we define a function cstep that, given a configuration, yields a distribution on $(\mathcal{C}, \Sigma_{\mathcal{C}})$ (Def. 7 below). Then, we use cstep to define the *multi-step* behaviour of configuration c as a sequence $\mathcal{S}^{\mathcal{C}}_{c,0}, \mathcal{S}^{\mathcal{C}}_{c,1}, \ldots$ of distributions on $(\mathcal{C}, \Sigma_{\mathcal{C}})$. To this end, we show that cstep is a Markov kernel (Prop. 3 below). Finally, to abstract from processes and focus only on data, from the sequence $\mathcal{S}^{\mathcal{C}}_{c,0}, \mathcal{S}^{\mathcal{C}}_{c,1}, \ldots$, we obtain a sequence of distributions $\mathcal{S}^{\mathcal{D}}_{c,0}, \mathcal{S}^{\mathcal{D}}_{c,1}, \ldots$ on $(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$ called the *evolution sequence* of the system (Def. 9 below).

Definition 7 (One-step semantics of configurations). *Function $\mathsf{cstep}\colon \mathcal{C} \to \Delta(\mathcal{C}, \Sigma_{\mathcal{C}})$ is defined for all configurations $\langle P, \mathbf{d}\rangle_{\mathcal{E}} \in \mathcal{C}$ by*

$$\mathsf{cstep}(\langle P, \mathbf{d}\rangle_{\mathcal{E}}) = \sum_{(\theta, P') \in \mathsf{supp}(\mathsf{pstep}(P, \mathbf{d}))} \mathsf{pstep}(P, \mathbf{d})(\theta, P') \cdot \langle P', \mathcal{E}(\theta(\mathbf{d}))\rangle_{\mathcal{E}}. \qquad (5)$$

The next result follows by $\mathcal{E}(\theta(\mathbf{d})) \in \Delta(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$ (Def. 5), which ensures that $\langle P', \mathcal{E}(\theta(\mathbf{d}))\rangle_{\mathcal{E}} \in \Delta(\mathcal{C}, \Sigma_{\mathcal{C}})$, and $\mathsf{pstep}(P, \mathbf{d}) \in \Delta(\Theta \times \mathcal{P})$ (Prop. 1).

Proposition 2. *For any configuration $c \in \mathcal{C}$, $\mathsf{cstep}(c)$ is a distribution on $(\mathcal{C}, \Sigma_{\mathcal{C}})$.*

Since $\mathsf{cstep}(c) \in \Delta(\mathcal{C}, \Sigma_{\mathcal{C}})$ for each $c \in \mathcal{C}$, we can rewrite $\mathsf{cstep}\colon \mathcal{C} \times \Sigma_{\mathcal{C}} \to [0,1]$, so that for each configuration $c \in \mathcal{C}$ and measurable set $\mathbb{C} \in \Sigma_{\mathcal{C}}$, $\mathsf{cstep}(c)(\mathbb{C})$ denotes the probability of reaching in one step a configuration in $\mathbb{C}$ starting from c. We can prove that cstep is the Markov kernel of the Markov process modelling our system. This follows by Proposition 2 and by proving that for each $\mathbb{C} \in \Sigma_{\mathcal{C}}$, the mapping $c \mapsto \mathsf{cstep}(c)(\mathbb{C})$ is $\Sigma_{\mathcal{C}}$-measurable for all $c \in \mathcal{C}$ (a detailed proof is in Appendix C).

Proposition 3. *The function* cstep *is a Markov kernel.*

Hence, the multi-step behaviour of configuration c can be defined as a time homogeneous Markov process having cstep as Markov kernel and δ_c as initial distribution.

Definition 8 (Multi-step semantics of configurations). *Let $c \in \mathcal{C}$ be a configuration. The* multi-step behaviour *of c is the sequence of distributions $\mathcal{S}^{\mathcal{C}}_{c,0}, \mathcal{S}^{\mathcal{C}}_{c,1}, \dots$ on $(\mathcal{C}, \Sigma_{\mathcal{C}})$ defined inductively as follows:*

$$\mathcal{S}^{\mathcal{C}}_{c,0}(\mathbb{C}) = \delta_c(\mathbb{C}), \textit{ for all } \mathbb{C} \in \Sigma_{\mathcal{C}}$$
$$\mathcal{S}^{\mathcal{C}}_{c,i+1}(\mathbb{C}) = \int_{\mathcal{C}} \mathsf{cstep}(b)(\mathbb{C}) \mathrm{d}(\mathcal{S}^{\mathcal{C}}_{c,i}(b)), \textit{ for all } \mathbb{C} \in \Sigma_{\mathcal{C}}.$$

We can prove that $\mathcal{S}^{\mathcal{C}}_{c,0}, \mathcal{S}^{\mathcal{C}}_{c,1}, \dots$ are well defined, namely they are distributions on $(\mathcal{C}, \Sigma_{\mathcal{C}})$. The proof follows by an easy induction based on Proposition 3.

Proposition 4. *For any $c \in \mathcal{C}$, all $\mathcal{S}^{\mathcal{C}}_{c,0}, \mathcal{S}^{\mathcal{C}}_{c,1}, \dots$ are distributions on $(\mathcal{C}, \Sigma_{\mathcal{C}})$.*

As the program-environment interplay can be observed only in the changes they induce on the data states, we define the *evolution sequence* of a configuration as the sequence of distributions over data states that are reached by it, step-by-step.

Definition 9 (Evolution sequence). *The* evolution sequence *of a configuration $c = \langle P, \mathbf{d}\rangle_{\mathcal{E}}$ is a sequence $\mathcal{S}^{\mathcal{D}}_c \in \Delta(\mathcal{D}, \mathcal{B}_{\mathcal{D}})^{\omega}$ of distributions over $\mathcal{D}$ s.t. $\mathcal{S}^{\mathcal{D}}_c = \mathcal{S}^{\mathcal{D}}_{c,0} \dots \mathcal{S}^{\mathcal{D}}_{c,n} \dots$ if and only if for all $i \geq 0$ and for all $\mathbb{D} \in \mathcal{B}_{\mathcal{D}}$, $\mathcal{S}^{\mathcal{D}}_{c,i}(\mathbb{D}) = \mathcal{S}^{\mathcal{C}}_{c,i}(\langle \mathcal{P}, \mathbb{D}\rangle_{\mathcal{E}})$.*

4 Towards a metric for systems

We aim at defining a *distance* over the systems described in the previous section, called the *evolution metric*, allowing us to do the following:

1. Verify how well a program is fulfilling its tasks.
2. Establish whether one program behaves better than another one in an environment.
3. Compare the interactions of a program with different environments.

These three objectives can be naturally obtained thanks to the possibility of modelling the program in isolation from the environment typical of our model, and to our purely data-driven system semantics. Intuitively, since the behaviour of a system is entirely described by its evolution sequence, the evolution metric $\mathfrak{m}$ will indeed be defined as a distance on the evolution sequences of systems. However, in order to obtain the proper technical definition of $\mathfrak{m}$, some considerations are due.

Firstly, we notice that in most applications the tasks of the program can be expressed in a purely data-driven fashion. We can identify a set of *parameters of interest* such that, at any time step, any difference between them and the data actually obtained can be interpreted as a flaw in system behaviour. We use a *penalty function* ρ to quantify these differences. From the penalty function we can obtain a *distance on data states*, namely a 1-bounded *hemimetric* $m^{\mathcal{D}}$ expressing how much a data state $\mathbf{d}_2$ is worse than a data state $\mathbf{d}_1$ according to parameters of interests. Secondly, we recall that the evolution sequence of a system consists in a sequence of *distributions* over data states. Hence, we use the *Wasserstein metric* to lift $m^{\mathcal{D}}$ to a distance $\mathbf{W}(m^{\mathcal{D}})$ over distributions over data states. Informally, with the Wasserstein metric we can express how much worse a configuration is expected to behave w.r.t. another one at a given time. Finally, we need to lift $\mathbf{W}(m^{\mathcal{D}})$ to a distance on the entire evolution sequences of systems. For our purposes, a reasonable choice is to take the maximum over time of the pointwise (w.r.t. time) Wasserstein distances (see Remark 1 below for further details on this choice).

A metric on data states. We start by proposing a metric on data states, seen as *static components* in isolation from processes and environment. To this end, we introduce a *penalty function* $\rho\colon \mathcal{D} \to [0,1]$, a continuous function that assigns to each data state $\mathbf{d}$ a penalty in $[0,1]$ expressing how far the values of the parameters of interest in $\mathbf{d}$ are from their desired ones (hence $\rho(\mathbf{d}) = 0$ if $\mathbf{d}$ respects all the parameters). Since some parameters can be time-dependent, so is ρ: at any time step τ, the τ-penalty function ρ_τ compares the data states w.r.t. the values of the parameters expected at time τ.

Example 5. We recall, from Example 2, that $\mathcal{D}_T = [t_m, t_M]$ and $\mathcal{D}_A = [0,1]$. The task of our program is to keep the value of T within the comfort zone $Z = [t_{\min}, t_{\max}]$, for some $t_{\min}, t_{\max}$, and that of A above a threshold $q_a \in \mathcal{D}_A$ (cf. Example 3). Hence, we define a penalty function that assigns the penalty 0 if the value of T is in Z and that of A is greater or equal to q_a, otherwise it is proportional to how much T and A are far from Z and q_a, respectively. We let $\rho_\tau(\mathbf{d}) = \max\{\rho^T(\mathbf{d}(T)), \rho^A(\mathbf{d}(A))\}$, where $\rho^T(t)$ is 0 if $t \in [t_{\min}, t_{\max}]$ and $\frac{\max\{t - t_{\max}, t_{\min} - t\}}{\max\{t_M - t_{\max}, t_{\min} - t_m\}}$ otherwise, while $\rho^A(q) = \max\{0, q_a - q\}$.

A formal definition of the penalty function is beyond the purpose of this paper, also due to its context-dependent nature. Besides, notice that we can assume that ρ already includes some tolerances w.r.t. the exact values of the parameters in its evaluation, and thus we do not consider them. The (*timed*) *metric on data states* is then defined as the asymmetric difference between the penalties assigned to them by the penalty function.

Definition 10 (Metric on data states). *For any time step τ, let $\rho_\tau\colon \mathcal{D} \to [0,1]$ be the τ-penalty function on $\mathcal{D}$. The* τ-metric on data states *in $\mathcal{D}$, $m^{\mathcal{D}}_{\rho,\tau}\colon \mathcal{D} \times \mathcal{D} \to [0,1]$, is defined, for all $\mathbf{d}_1, \mathbf{d}_2 \in \mathcal{D}$, by $m^{\mathcal{D}}_{\rho,\tau}(\mathbf{d}_1, \mathbf{d}_2) = \max\{\rho_\tau(\mathbf{d}_2) - \rho_\tau(\mathbf{d}_1), 0\}$.*

Notice that $m^{\mathcal{D}}_{\rho,\tau}(\mathbf{d}_1,\mathbf{d}_2) > 0$ iff $\rho_\tau(\mathbf{d}_2) > \rho_\tau(\mathbf{d}_1)$, i.e., the penalty assigned to $\mathbf{d}_2$ is higher than that assigned to $\mathbf{d}_1$. For this reason, we say that $m^{\mathcal{D}}_{\rho,\tau}(\mathbf{d}_1,\mathbf{d}_2)$ expresses *how worse* $\mathbf{d}_2$ is than $\mathbf{d}_1$ w.r.t. the objectives of the system. It is not hard to see that for all $\mathbf{d}_1,\mathbf{d}_2,\mathbf{d}_3 \in \mathcal{D}$ we have $m^{\mathcal{D}}_{\rho,\tau}(\mathbf{d}_1,\mathbf{d}_2) \le 1$, $m^{\mathcal{D}}_{\rho,\tau}(\mathbf{d}_1,\mathbf{d}_1) = 0$, and $m^{\mathcal{D}}_{\rho,\tau}(\mathbf{d}_1,\mathbf{d}_2) \le m^{\mathcal{D}}_{\rho,\tau}(\mathbf{d}_1,\mathbf{d}_3) + m^{\mathcal{D}}_{\rho,\tau}(\mathbf{d}_3,\mathbf{d}_2)$, thus ensuring that $m^{\mathcal{D}}_{\rho,\tau}$ is a 1-bounded hemimetric.

Proposition 5. *Function $m^{\mathcal{D}}_{\rho,\tau}$ is a 1-bounded hemimetric on $\mathcal{D}$.*

Lifting $m^{\mathcal{D}}$ to distributions. The second step to obtain the evolution metric consists in lifting $m^{\mathcal{D}}$ to a metric on distributions on data states. Among the several notions of lifting in the literature (see [16] for a survey), we opt for that of Wasserstein, since: i) it preserves the properties of the ground metric; ii) it allows us to deal with discrete and continuous measures; iii) it is computationally tractable via statistical inference. According to Def. 1, the Wasserstein lifting of $m^{\mathcal{D}}_{\rho}$ to a distance between two distributions $\mu,\nu \in \Delta(\mathcal{D},\mathcal{B}_{\mathcal{D}})$ is defined by

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

The evolution metric. We now need to lift $\mathbf{W}(m^{\mathcal{D}})$ to a distance on evolution sequences. To this end, we observe that the evolution sequence of a configuration includes the distributions over data states induced after *each* computation step. Thus, the time step between two distributions is determined by the program. However, it could be the case that the changes on data induced by the environment can be appreciated only along wider time intervals. Our running example is a clear instance of this situation: while we can reasonably assume that the duration of the computation steps of the thermostat is of the order of a millisecond, the variations in the temperature that can be detected in the same time interval are indeed negligible w.r.t. the program's task. A significant temperature rise or drop can be observed only in longer time. To deal with this kind of situations, we introduce the notion of *observation times*, namely a *discrete* set OT of time steps at which the modifications induced by the program-environment interplay give us useful information on the evolution of the system. Hence, a comparison of the evolution sequences based on the differences in the distributions reached at the times in OT can be considered meaningful. Moreover, considering only the differences at the observation times will favour the computational tractability of the evolution metric.

We define the evolution metric as a sort of *weighted infinity norm* of the tuple of the Wasserstein distances between the distributions in the evolution sequences. As weight we consider a non-increasing function $\lambda\colon \mathrm{OT} \to (0,1]$ expressing how much the distance at time τ affects the overall distance between configurations c_1 and c_2. We refer to λ as to the *discount function*, and to $\lambda(\tau)$ as to the *discount factor at time τ*.

Definition 11 (Evolution metric). *Assume a set* OT *of observation times and a discount function λ. Let ρ be a penalty function and let $m^{\mathcal{D}}_{\rho}$ be the metric on data states defined on it. Then, the* λ-evolution metric *over ρ and* OT *is the mapping $\mathfrak{m}^{\lambda}_{\rho,\mathrm{OT}}\colon \mathcal{C}\times\mathcal{C} \to [0,1]$ defined, for all configurations $c_1,c_2 \in \mathcal{C}$, by*

$$\mathfrak{m}^{\lambda}_{\rho,\mathrm{OT}}(c_1,c_2) = \sup_{\tau\in\mathrm{OT}} \lambda(\tau)\cdot\mathbf{W}(m^{\mathcal{D}}_{\rho,\tau})\Big(\mathcal{S}^{\mathcal{D}}_{c_1,\tau},\mathcal{S}^{\mathcal{D}}_{c_2,\tau}\Big).$$

Since $\mathfrak{m}^{\mathcal{D}}_{\rho,\tau}$ is a 1-bounded hemimetric (Proposition 5) and lifting $\mathbf{W}$ preserves such a property, we can easily derive the same property for $\mathfrak{m}^{\lambda}_{\mathrm{OT}}$.

Proposition 6. *Function $\mathfrak{m}^{\lambda}_{\rho,\mathrm{OT}}$ is a 1-bounded hemimetric on $\mathcal{C}$.*

Notice that if λ is a *strictly* non-increasing function, then it specifies how much the distance of *future events* is mitigated and, moreover, it guarantees that to obtain upper bounds on the evolution metric only a *finite* number of observations is needed.

Remark 1. Usually, due to the presence of uncertainties, the behaviour of a system can be considered acceptable even if it differs from its intended one *up-to a certain tolerance*. Similarly, the properties of adaptability and reliability that we aim to study will check whether a program is able to perform well in a perturbed environment *up-to a given tolerance*. In this setting, the choice of defining the evolution metric as the pointwise maximal distance in time between the evolution sequences of systems is natural and reasonable: if in the worst case (the maximal distance) the program keeps the parameters of interest within the given tolerance, then its entire behaviour can be considered acceptable. However, with this approach we have that a program is only as good as its worst performance, and one could argue that there are application contexts in which our evolution metric would be less meaningful. For these reasons, we remark that we could have given a *parametric* version of Definition 11 and defining the evolution metric in terms of a generic *aggregation function* f over the tuple of Wasserstein distances. Then, one could choose the best instantiation for f according to the chosen application context. The use of a parametric definition would have not affected the technical development of our paper. However, to keep the notation and presentation as simple as possible, we opted to define $\mathfrak{m}^{\lambda}_{\rho,\mathrm{OT}}$ directly in the weighted infinity norm form. A similar reasoning applies to the definition of the penalty function that we gave in Example 5.

5 Estimating the evolution metric

In this section we show how the evolution metric can be estimated via statistical techniques. Firstly, we show how we can estimate the evolution sequence of a given configuration c. Then, we evaluate the distance between two configurations c_1 and c_2 on their estimated evolution sequences.

Computing empirical evolution sequences. To compute the empirical evolution sequence of a configuration c the following function EST can be used.

```
1:  function EST($c, k, N$)
2:      $\forall i : (0 \leq i \leq k) : E_i \leftarrow \emptyset$
3:      $counter \leftarrow 0$
4:      while $counter < N$ do
5:          $(c_0, \ldots, c_k) \leftarrow$ SIM$(c, k)$
6:          $\forall i : E_i \leftarrow E_i, c_i$
7:          $counter \leftarrow counter + 1$
8:      end while
9:      return $E_0, \ldots, E_k$
10: end function
```

Function EST(c, k, N) invokes N times function SIM, i.e., any simulation algorithm sampling a sequence of configurations $c_0, \ldots, c_k$, modelling k steps of a computation from $c = c_0$. Then, the sequence $E_0, \ldots, E_k$ is computed, where E_i is the tuple $c_i^1, \ldots, c_i^N$ of configurations observed at time i in each of the N sampled computations.

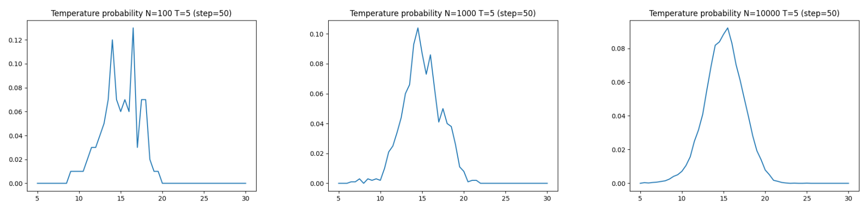


Fig. 1: Estimated distribution of the temperature after 50 steps with $N = 10^2$, $N = 10^3$ and $N = 10^4$. As comfort zone we consider the interval $[15, 20]$.

Each E_i can be used to estimate the distribution $\mathcal{S}^{\mathcal{C}}_{c,i}$. For any i, with $0 \leq i \leq k$, we let $\hat{\mathcal{S}}^{\mathcal{C},N}_{c,i}$ be the distribution s.t. for any $\mathbb{C} \in \Sigma_{\mathcal{C}}$ we have $\hat{\mathcal{S}}^{\mathcal{C},N}_{c,i}(\mathbb{C}) = \frac{|E_i \cap \mathbb{C}|}{N}$. Finally, we let $\hat{\mathcal{S}}^{\mathcal{D},N}_{c} = \hat{\mathcal{S}}^{\mathcal{D},N}_{c,0} \dots \hat{\mathcal{S}}^{\mathcal{D},N}_{c,k}$ be the *empirical evolution sequence* s.t. for any measurable set of data states $\mathbb{D} \in \mathcal{B}_{\mathcal{D}}$ we have $\hat{\mathcal{S}}^{\mathcal{D},N}_{c,i}(\mathbb{D}) = \hat{\mathcal{S}}^{\mathcal{C},N}_{c,i}(\langle \mathcal{P}, \mathbb{D} \rangle_{\mathcal{E}})$. Then, by applying the weak law of large numbers to the i.i.d samples, we get that when N goes to infinite both $\hat{\mathcal{S}}^{\mathcal{C},N}_{c,i}$ and $\hat{\mathcal{S}}^{\mathcal{D},N}_{c,i}$ converge weakly to $\mathcal{S}^{\mathcal{C}}_{c,i}$ and $\mathcal{S}^{\mathcal{D}}_{c,i}$ respectively:

$$\lim_{N \to \infty} \hat{\mathcal{S}}^{\mathcal{C},N}_{c,i} = \mathcal{S}^{\mathcal{C}}_{c,i} \qquad\qquad \lim_{N \to \infty} \hat{\mathcal{S}}^{\mathcal{D},N}_{c,i} = \mathcal{S}^{\mathcal{D}}_{c,i}. \tag{6}$$

An implementation of algorithm Sim is in Appendix D (Fig. 4). The tool and the scripts of the examples are available (in Python) at `https://github.com/quasylab/spear`.

Example 6. We apply our simulation to the heating system from Sect. 3, with initial configuration $c_1 = \langle \mathsf{P}, \{T = 5.0, T_s = 5.0, h = 0, A = 0.5, A_s = 0.5, e = 0\} \rangle_{\mathcal{E}}$, where P is the process in Ex. 3, and $[t_{\min}, t_{\max}] = [15, 20]$. In Fig. 1 the probability distribution of the temperature after 50 steps is reported. We observe that the higher the number of samplings, the smoother the plot.

Computing distance between two configurations. Function Est allows us to collect independent samples at each time step i from 0 to a deadline k. These samples can be used to estimate the distance between two configurations c_1 and c_2. Following an approach similar to the one presented in [21], to estimate the Wasserstein distance $\mathbf{W}(m^{\mathcal{D}}_{\rho,i})$ between two (unknown) distributions $\mathcal{S}^{\mathcal{D}}_{c_1,i}$ and $\mathcal{S}^{\mathcal{D}}_{c_2,i}$ we can use N independent samples $\{c_1^1, \dots, c_1^N\}$ taken from $\mathcal{S}^{\mathcal{C}}_{c_1,i}$ and $\ell \cdot N$ independent samples $\{c_2^1, \dots, c_2^{\ell \cdot N}\}$ taken from $\mathcal{S}^{\mathcal{C}}_{c_2,i}$. After that, we exploit the i-penalty function ρ and we consider the two sequences of values: $\{\omega_j = \rho_i(\mathbf{d}_1^j) | \langle P_1^j, \mathbf{d}_1^j \rangle_{\mathcal{E}_1} = c_1^j\}$ and $\{\nu_h = \rho_i(\mathbf{d}_2^h) | \langle P_2^h, \mathbf{d}_2^h \rangle_{\mathcal{E}_2} = c_2^h\}$. We can assume, without loss of generality, that these sequences are ordered, i.e., $\omega_j \leq \omega_{j+1}$ and $\nu_h \leq \nu_{h+1}$. The value $\mathbf{W}(m^{\mathcal{D}}_{\rho,i})(\mathcal{S}^{\mathcal{D}}_{c_1,i}, \mathcal{S}^{\mathcal{D}}_{c_2,i})$ can be approximated as $\frac{1}{\ell N} \sum_{h=1}^{\ell N} \max\{\nu_h - \omega_{\lceil \frac{h}{\ell} \rceil}, 0\}$. The next theorem, based on results in [21, 23], ensures that the larger the number of samplings the closer the gap between the estimated value and the exact one.

```
**function** $\textsc{Dist}(c_1, c_2, \rho, \lambda, \mathrm{OT}, N, \ell)$
    $k \leftarrow \max_{\mathrm{OT}}$
    $E_{1,1}, \dots, E_{1,k} \leftarrow \textsc{Est}(c_1, k, N)$
    $E_{2,1}, \dots, E_{2,k} \leftarrow \textsc{Est}(c_2, k, \ell N)$
    $m \leftarrow \infty$
    **for all** $i \in \mathrm{OT}$ **do**
        $m_i \leftarrow \textsc{CompW}(E_{1,i}, E_{2,i}, \rho_i)$
        $m \leftarrow \min\{m, \lambda(i) \cdot m_i\}$
    **end for**
    **return** $m$
**end function**
```

```
**function** $\textsc{CompW}(E_1, E_2, \rho)$
    $(\langle P_1^1, \mathbf{d}_1^1\rangle_{\mathcal{E}_1}, \dots, \langle P_1^N, \mathbf{d}_1^N\rangle_{\mathcal{E}_1}) \leftarrow E_1$
    $(\langle P_2^1, \mathbf{d}_2^1\rangle_{\mathcal{E}_2}, \dots, \langle P_2^{\ell N}, \mathbf{d}_2^{\ell N}\rangle_{\mathcal{E}_2}) \leftarrow E_2$
    $\forall j : (1 \leq j \leq N) : \omega_j \leftarrow \rho(\mathbf{d}_1^j)$
    $\forall h : (1 \leq h \leq \ell N) : \nu_h \leftarrow \rho(\mathbf{d}_2^h)$
    re index $\{\omega_j\}$ s.t. $\omega_j \leq \omega_{j+1}$
    re index $\{\nu_h\}$ s.t. $\nu_h \leq \nu_{h+1}$
    **return** $\frac{1}{\ell N} \sum_{h=1}^{\ell N} |\omega_{\lceil \frac{h}{\ell} \rceil} - \nu_h|$
**end function**
```

Fig. 2: Functions used to estimate the evolution metric on systems.

Theorem 1. *Let $\mathcal{S}^{\mathcal{C}}_{c_1,i}, \mathcal{S}^{\mathcal{C}}_{c_2,i} \in \Delta(\mathcal{C}, \Sigma_{\mathcal{C}})$ be unknown, and ρ be a penalty function. Let $\{\omega_j = \rho_i(\mathbf{d}_1^j)\}$ and $\{\nu_h = \rho_i(\mathbf{d}_2^h)\}$ be the ordered sequences obtained from independent samples taken from $\mathcal{S}^{\mathcal{C}}_{c_1,i}$ and $\mathcal{S}^{\mathcal{C}}_{c_2,i}$, respectively. Then, it holds, a.s., that $\mathbf{W}(m^{\mathcal{D}}_{\rho,i})(\mathcal{S}^{\mathcal{D}}_{c_1,i}, \mathcal{S}^{\mathcal{D}}_{c_2,i}) = \lim_{N\to\infty} \frac{1}{\ell N} \sum_{h=1}^{\ell N} \max\{\nu_h - \omega_{\lceil \frac{h}{\ell} \rceil}, 0\}$.*

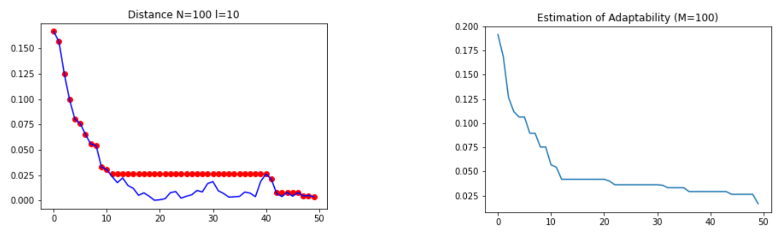
The outlined procedure is realised by functions $\textsc{Dist}$ and $\textsc{CompW}$ in Fig. 2. The former takes as input the two configurations to compare, the penalty function (seen as the sequence of the i-penalty functions), the discount function λ, the set OT of observation times which we assume to be bounded, and the parameters N and ℓ used to obtain the samplings of computation. Function $\textsc{Dist}$ collects the samples E_i of possible computations during the observation period $[0, \max_{\mathrm{OT}}]$, where $\max_{\mathrm{OT}}$ denotes the last observation time. Then, for each observation time $i \in \mathrm{OT}$, the distance at time i is computed via the function $\textsc{CompW}(E_{1,i}, E_{2,i}, \rho_i)$. Since the penalty function allows us to reduce the evaluation of the Wasserstein distance in $\mathbb{R}^n$ to its evaluation on $\mathbb{R}$, due to the sorting of $\{\nu_h \mid h \in [1, \dots, \ell N]\}$ the complexity of function $\textsc{CompW}$ is $O(\ell N \log(\ell N))$ (cf. [21]).

Example 7. We change the initial value of the air quality in the configuration c_1 in Example 6, and consider $c_2 = \langle \mathsf{P}, \{T = 5.0, T_s = 5.0, h = 0, A = 0.3, A_s = 0.3, e = 0\}\rangle_{\mathcal{E}}$. Figure 3a shows the variation in time of the distance between c_1 and c_2.

6 Adaptability and reliability of programs

In this section we exploit the evolution metric to study some dependability properties of programs, which we call *adaptability* and *reliability*, w.r.t. a data state and an environment. Both notions entail the ability of the process to induce a *similar* behaviour in systems that start from *similar* initial conditions. They differ in how time is considered.

The notion of adaptability imposes some constraints on the *long term* behaviour of systems, disregarding their possible initial dissimilarities. Given the thresholds $\eta_1, \eta_2 \in [0, 1)$ and an observable time $\tilde{\tau}$, we say that a program P is adaptable w.r.t. a data state $\mathbf{d}$ and an environment evolution $\mathcal{E}$ if whenever P starts its computation from a data state $\mathbf{d}'$ that differs from $\mathbf{d}$ for at most η_1, then we are guaranteed that the distance between the evolution sequences of the two systems after time $\tilde{\tau}$ is bounded by η_2.



(a) In blue: pointwise distance between c_1 and c_2. In red: $\mathfrak{m}^{\lambda}_{\rho,\{\tau' \geq \tau\}}(c_1, c_2)$, for each τ (Ex. 7).

(b) Adaptability of P in c_1 (from Ex. 6) for $M = 100$, $\eta_1 = 0.2$ (Ex. 8).

Fig. 3: Examples of the evaluation of the evolution metric (assuming λ being the constant 1).

Definition 12 (Adaptability). *Let $\tilde{\tau} \in \mathrm{OT}$ and $\eta_1, \eta_2 \in [0, 1)$. We say that P is* $(\tilde{\tau}, \eta_1, \eta_2)$-adaptable *w.r.t. the data state $\mathbf{d}$ and the environment evolution $\mathcal{E}$ if $\forall\, \mathbf{d}' \in \mathcal{D}$ with $m^{\mathcal{D}}_{\rho,0}(\mathbf{d}, \mathbf{d}') \leq \eta_1$ it holds $\mathfrak{m}^{\lambda}_{\{\tau \in \mathrm{OT} | \tau \geq \tilde{\tau}\}}(\langle P, \mathbf{d}\rangle_{\mathcal{E}}, \langle P, \mathbf{d}'\rangle_{\mathcal{E}}) \leq \eta_2$.*

We remark that one can always consider the data state $\mathbf{d}$ as the ideal model of the world used for the specification of P, and the data state $\mathbf{d}'$ as the real world in which P has to execute. Hence, the idea behind adaptability is that even if the initial behaviour of the two systems is quite different, P is able to reduce the gap between the real evolution and the desired one within the time threshold $\tilde{\tau}$. Notice that being $(\tilde{\tau}, \eta_1, \eta_2)$-adaptable for $\tilde{\tau} = \min\{\tau \mid \tau \in \mathrm{OT}\}$ is equivalent to being (τ, η_1, η_2)-adaptable for all $\tau \in \mathrm{OT}$.

The notion of reliability strengthens that of adaptability by bounding the distance on the evolution sequences from the beginning. A program is reliable if it guarantees that small variations in the initial conditions cause only bounded variations in its evolution.

Definition 13 (Reliability). *Let $\eta_1, \eta_2 \in [0, 1)$. We say that P is* (η_1, η_2)-reliable *w.r.t. the data state $\mathbf{d}$ and the environment evolution $\mathcal{E}$ if $\forall\, \mathbf{d}' \in \mathcal{D}$ with $m^{\mathcal{D}}_{\rho,0}(\mathbf{d}, \mathbf{d}') \leq \eta_1$ it holds $\mathfrak{m}^{\lambda}_{\mathrm{OT}}(\langle P, \mathbf{d}\rangle_{\mathcal{E}}, \langle P, \mathbf{d}'\rangle_{\mathcal{E}}) \leq \eta_2$.*

We can use our algorithm to verify adaptability and reliability of a given program. Given a configuration $\langle P, \mathbf{d}\rangle_{\mathcal{E}}$, a set OT of observation times and a given threshold $\eta_1 \geq 0$, we can sample M variations $\{\mathbf{d}_1, \dots, \mathbf{d}_M\}$ of $\mathbf{d}$, s.t. for any i, $m^{\mathcal{D}}_{\rho,0}(\mathbf{d}, \mathbf{d}_i) \leq \eta_1$. Then, for each sampled data state we can estimate the distance between $c = \langle P, \mathbf{d}\rangle_{\mathcal{E}}$ and $c_i = \langle P, \mathbf{d}_i\rangle_{\mathcal{E}}$ at the different time steps in OT, namely $\mathfrak{m}^{\lambda}_{\{\tau \in \mathrm{OT} | \tau \geq \tilde{\tau}\}}(c, c_i)$ for any $\tilde{\tau} \in \mathrm{OT}$. Finally, for each $\tilde{\tau} \in OT$, we let $l_{\tilde{\tau}} = \max_i\{\mathfrak{m}^{\lambda}_{\{\tau \in \mathrm{OT} | \tau \geq \tilde{\tau}\}}(c, c_i)\}$. We can observe that, for the chosen η_1, each $l_{\tilde{\tau}}$ gives us a lower bound to the $\tilde{\tau}$-adaptability of the program. Similarly, for $\tau_{\min} = \min_{\mathrm{OT}} \tau$, $l_{\tau_{\min}}$ gives a lower bound for its reliability.

Example 8. Figure 3b shows the evaluation of l_τ for the program P in the configuration c_1 from Example 6 with parameters $M = 100$ and $\eta_1 = 0.2$. Observe that the initial perturbation is not amplified and after 12 steps it is almost absorbed. In particular, our program is $(12, 0.2, \eta_2)$-adaptable w.r.t. the data state and the environment evolution in Example 6, for any $\eta_2 \geq 0.05 + e^{12}_{\mathbf{W}}$, where $e^{12}_{\mathbf{W}}$ is the approximation error $e^{12}_{\mathbf{W}} = |\,\mathbf{W}(m^{\mathcal{D}}_{\rho,12})(\hat{\mathcal{S}}^{\mathcal{D},1000}_{c_1,12}, \hat{\mathcal{S}}^{\mathcal{D},10000}_{c_2,12}) - \mathbf{W}(m^{\mathcal{D}}_{\rho,12})(\mathcal{S}^{\mathcal{D}}_{c_1,12}, \mathcal{S}^{\mathcal{D}}_{c_2,12})|$. We refer the interested reader to [20, Corollary 3.5, Equation (3.10)] for an estimation of $e^{12}_{\mathbf{W}}$.

References

1. Abate, A., D'Innocenzo, A., Benedetto, M.D.D.: Approximate abstractions of stochastic hybrid systems. IEEE Trans. Automat. Contr. **56**(11), 2688–2694 (2011). https://doi.org/10.1109/TAC.2011.2160595
2. Abate, A., Katoen, J., Lygeros, J., Prandini, M.: Approximate model checking of stochastic hybrid systems. Eur. J. Control **16**(6), 624–641 (2010). https://doi.org/10.3166/ejc.16.624-641
3. Bernardo, M., Nicola, R.D., Loreti, M.: A uniform framework for modeling nondeterministic, probabilistic, stochastic, or mixed processes and their behavioral equivalences. Inf. Comput. **225**, 29–82 (2013). https://doi.org/10.1016/j.ic.2013.02.004
4. Bogachev, V.I.: Measure Theory. No. v. 1 in Measure Theory, Springer-Verlag, Berlin/Heidelberg (2007). https://doi.org/10.1007/978-3-540-34514-5
5. Cassandras, C.G., Lygeros, J., (eds.): Stochastic Hybrid Systems. No. 24 in Control Engineering, CRC Press, Boca Raton, 1st edn. (2007), `https://doi.org/10.1201/9781315221625`
6. Ciocchetta, F., Hillston, J.: Bio-pepa: An extension of the process algebra PEPA for biochemical networks. Electron. Notes Theor. Comput. Sci. **194**(3), 103–117 (2008). https://doi.org/10.1016/j.entcs.2007.12.008
7. Faugeras, O.P., Rüschendorf, L.: Risk excess measures induced by hemi-metrics. Probability, Uncertainty and Quantitative Risk **3:6** (2018). https://doi.org/10.1186/s41546-018-0032-0
8. Girard, A., Gößler, G., Mouelhi, S.: Safety controller synthesis for incrementally stable switched systems using multiscale symbolic models. IEEE Trans. Automat. Contr. **61**(6), 1537–1549 (2016). https://doi.org/10.1109/TAC.2015.2478131
9. van Glabbeek, R.J., Smolka, S.A., Steffen, B.: Reactive, generative and stratified models of probabilistic processes. Inf. Comput. **121**(1), 59–80 (1995). https://doi.org/10.1006/inco.1995.1123
10. Heredia, G., Jimenez-Cano, A.E., Sánchez, I., Llorente, D., Vega, V., Braga, J., Acosta, J.Á., Ollero, A.: Control of a multirotor outdoor aerial manipulator. In: Proceedings of IROS 2014. pp. 3417–3422. IEEE (2014). https://doi.org/10.1109/IROS.2014.6943038
11. Hillston, J., Hermanns, H., Herzog, U., Mertsiotakis, V., Rettelbach, M.: Stochastic process algebras: integrating qualitative and quantitative modelling. In: Proceedings of the 7th IFIP WG6.1 International Conference on Formal Description Techniques 1994. IFIP Conference Proceedings, vol. 6, pp. 449–451 (1994)
12. Hu, J., Lygeros, J., Sastry, S.: Towars a theory of stochastic hybrid systems. In: Proceedings of HSCC 2000. Lecture Notes in Computer Science, vol. 1790, pp. 160–173 (2000). https://doi.org/10.1007/3-540-46430-1_16
13. Kopetz, H.: Internet of Things, pp. 307–323. Springer US, Boston, MA (2011). https://doi.org/10.1007/978-1-4419-8237-7_13
14. Malhame, R., Yee Chong, C.: Electric load model synthesis by diffusion approximation of a high-order hybrid-state stochastic system. IEEE Trans. Automat. Contr. **30**(9), 854–660 (1985)
15. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley Series in Probability and Statistics, Wiley, United States (2005)
16. Rachev, S.T., Klebanov, L.B., Stoyanov, S.V., Fabozzi, F.J.: The Methods of Distances in the Theory of Probability and Statistics. Springer (2013)
17. Rajkumar, R., Lee, I., Sha, L., Stankovic, J.A.: Cyber-physical systems: the next computing revolution. In: DAC. pp. 731–736. ACM (2010)
18. Royden, H.L.: Real Analysis. Macmillan, New York, NY, 3rd edn. (1988)

19. Segala, R.: Modeling and verification of randomized distributed real-time systems. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA (1995), `http://hdl.handle.net/1721.1/36560`
20. Sriperumbudur, B.K., Fukumizu, K., Gretton, A., Schölkopf, B., Lanckriet, G.R.G.: On the empirical estimation of integral probability metrics. Electronic Journal of Statistics **6**, 1550–1599 (2021). https://doi.org/10.1214/12-EJS722
21. Thorsley, D., Klavins, E.: Approximating stochastic biochemical processes with Wasserstein pseudometrics. IET Syst. Biol. **4**(3), 193–211 (2010). https://doi.org/10.1049/iet-syb.2009.0039
22. Vaserstein, L.N.: Markovian processes on countable space product describing large systems of automata. Probl. Peredachi Inf. **5**(3), 64–72 (1969)
23. Villani, C.: Optimal transport: old and new, vol. 338. Springer (2008)
24. Virágh, C., Nagy, M., Gershenson, C., Vásárhelyi, G.: Self-organized UAV traffic in realistic environments. In: Proceedings of IROS 2016. pp. 1645–1652. IEEE (2016). https://doi.org/10.1109/IROS.2016.7759265

A The Wasserstein hemimetric

Firstly, we recall that given a (hemi)metric space (Ω, m), the (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.

The definition of the Wasserstein lifting 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: 1. $\mathbb{R}$ is a Polish space; and 2. every closed subset of a Polish space is in turn a Polish space. Moreover, for any $n \in N$, 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 $\mathcal{B}(\bigtimes_{i=1}^{n} \Omega_i) = \bigotimes_{i=1}^{n} \mathcal{B}(\Omega_i)$ (a formal proof can be found in, e.g., [4, 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.

Despite the original version of the Wasserstein distance being defined on a metric on Ω, the Wasserstein hemimetric given in Definition 1 is well-defined. In particular, the Wasserstein hemimetric is given in [7] as Definition 7 (considering the compound risk excess metric defined in Equation (31) in that paper), and Proposition 4 in [7] guarantees that it is indeed a well-defined hemimetric on $\Delta(\Omega, \mathcal{B}(\Omega))$. Moreover, Proposition 6 in [7] guarantees that the same result holds for the hemimetric $m(x, y) = \max\{x - y, 0\}$ which, as we will see, plays an important role in our work (cf. Definition 10 in Section 4).

B Proof of Proposition 2

Proposition 2. *For any configuration $c \in \mathcal{C}$,* $\mathsf{cstep}(c)$ *is a distribution on $(\mathcal{C}, \Sigma_{\mathcal{C}})$.*

Proof. Being $\mathbf{d}$ a data state in $\mathcal{D}$ and θ an effect, $\theta(\mathbf{d})$ is a data state. Then, $\mathcal{E}(\theta(\mathbf{d}))$ is a probability measure on $(\mathcal{D}, \Sigma_{\mathcal{D}})$, thus implying that $\langle P', \mathcal{E}(\theta(\mathbf{d}))\rangle_{\mathcal{E}}$ is a probability measure on $(\mathcal{C}, \Sigma_{\mathcal{C}})$. Finally, by Proposition 1 we get that $\mathsf{cstep}(\langle P, \mathbf{d}\rangle_{\mathcal{E}})$ is a convex combination of probability measures on $(\mathcal{C}, \Sigma_{\mathcal{C}})$ whose weights sum up to 1, which gives the thesis.

C Proof of Proposition 3

Proposition 3. *The function* cstep *is a Markov kernel.*

Proof. We need to show that cstep satisfies the two properties of Markov kernels, namely

1. For each configuration $c \in \mathcal{C}$, the mapping $\mathbb{C} \mapsto \mathsf{cstep}(c)(\mathbb{C})$ is a probability measure on $(\mathcal{C}, \Sigma_{\mathcal{C}})$.
2. For each measurable set $\mathbb{C} \in \Sigma_{\mathcal{C}}$, the mapping $c \mapsto \mathsf{cstep}(c)(\mathbb{C})$ is $\Sigma_{\mathcal{C}}$-measurable.

Item 1 follows directly by Proposition 2.

Let us focus on item 2. By Definition 7, for each $\langle P, \mathbf{d}\rangle_{\mathcal{E}} \in \mathcal{C}$ we have that

$$\mathsf{cstep}(\langle P, \mathbf{d}\rangle_{\mathcal{E}})(\mathbb{C}) = \sum_{(\theta, P') \in \mathsf{supp}(\mathsf{pstep}(P, \mathbf{d}))} \mathsf{pstep}(P, \mathbf{d})(\theta, P') \cdot \langle P', \mathcal{E}(\theta(\mathbf{d}))\rangle_{\mathcal{E}}(\mathbb{C}) \ .$$

As, by definition, each $\theta \in \Theta$ and $\mathcal{E}$ are $\Sigma_{\mathcal{D}}$-measurable functions, we can infer that also their composition $\mathcal{E}(\theta(\cdot))$ is a $\Sigma_{\mathcal{D}}$-measurable function. Since, moreover, $\Sigma_{\mathcal{C}}$ is the (smallest) σ-algebra generated by $\Sigma_{\mathcal{P}} \times \Sigma_{\mathcal{D}}$ and every subset of $\mathcal{P}$ is a measurable set in $\Sigma_{\mathcal{P}}$, we can also infer that $\langle(\cdot), \mathcal{E}(\theta(\cdot))\rangle_{\mathcal{E}}$ is a $\Sigma_{\mathcal{C}}$-measurable function. Finally, we recall that by Proposition 1 we have that $\mathsf{supp}(\mathsf{pstep}(P, \mathbf{d}))$ is finite. Therefore, we can conclude that cstep is a $\Sigma_{\mathcal{C}}$-measurable function as linear combination of a finite collection of $\Sigma_{\mathcal{C}}$-measurable functions (see, e.g., [18, Chapter 3.5, Proposition 19]).

D The simulation algorithm

Given a configuration $\langle P, \mathbf{d}\rangle_{\mathcal{E}}$ and an integer k we can use function SIM, defined in Figure 4, to sample a sequence of configurations the form

$$\langle P_0, \mathbf{d}_0\rangle_{\mathcal{E}}, \langle P_1, \mathbf{d}_1\rangle_{\mathcal{E}}, \ldots, \langle P_k, \mathbf{d}_k\rangle_{\mathcal{E}}.$$

This sequence represents k-steps of a computation starting from $\langle P, \mathbf{d}\rangle_{\mathcal{E}} = \langle P_0, \mathbf{d}_0\rangle_{\mathcal{E}}$. Each step of the sequence is computed by using function SIMSTEP, also defined in Figure 4. There we let RAND be a function that allow us to get a uniform random number in $(0, 1]$ while SAMPLE$(\mathcal{E}, \mathbf{d})$ is used to sample a data state in the distribution $\mathcal{E}(\mathbf{d})$. We assume that for any measurable set of data states $\mathbb{D} \in \Sigma_{\mathcal{D}}$ the probability of SAMPLE$(\mathcal{E}, \mathbf{d})$ to be in $\mathbb{D}$ is equal to the probability of $\mathbb{D}$ induced by $\mathcal{E}(\mathbf{d})$, namely:

$$Pr(\text{SAMPLE}(\mathcal{E}, \mathbf{d}) \in \mathbb{D}) = \mathcal{E}(\mathbf{d})(\mathbb{D}) \ . \tag{7}$$

We can then extend this property to configurations and function SIMSTEP:

Lemma 1. *For any configuration $\langle P, \mathbf{d}\rangle_{\mathcal{E}} \in \mathcal{C}$, and for any measurable set $\mathbb{C} \in \Sigma_{\mathcal{C}}$:*

$$Pr(\text{SIMSTEP}(\langle P, \mathbf{d}\rangle_{\mathcal{E}}) \in \mathbb{C}) = \mathsf{cstep}(\langle P, \mathbf{d}\rangle_{\mathcal{E}})(\mathbb{C}) \ .$$

Proof. To prove the thesis it is enough to show that the property

$$Pr(\text{SIMSTEP}(\langle P, \mathbf{d}\rangle_{\mathcal{E}}) \in \mathbb{C}) = \mathsf{cstep}(\langle P, \mathbf{d}\rangle_{\mathcal{E}})(\mathbb{C})$$

holds on the generators of the σ-algebra $\Sigma_{\mathcal{C}}$. Hence, assume that $\mathbb{C} = \langle \mathbb{P}, \mathbb{D}\rangle_{\mathcal{E}}$ for some $\mathbb{P} \in \Sigma_{\mathcal{P}}$ and $\mathbb{D} \in \Sigma_{\mathcal{D}}$. Then we have

$$\mathsf{cstep}(\langle P, \mathbf{d}\rangle_{\mathcal{E}})(\mathbb{C})$$

```
1:  function SIM($\langle P, \mathbf{d} \rangle_{\mathcal{E}}, k$)
2:      $i \leftarrow 0$
3:      $c \leftarrow \langle P, \mathbf{d} \rangle_{\mathcal{E}}$
4:      $a \leftarrow c$
5:      while $i \leq k$ do
6:          $c \leftarrow$ SIMSTEP($c$)
7:          $a \leftarrow a, c$
8:          $i \leftarrow i + 1$
9:      end while
10:     return $a$
11: end function
```

```
1: function SIMSTEP($\langle P, \mathbf{d} \rangle_{\mathcal{E}}$)
2:     $\sum_{i=1}^{n} p_i(\theta_i, P_i) \leftarrow \mathsf{pstep}(P, \mathbf{d})$
3:     $u \leftarrow$ RAND()
4:     let $i$ s.t. $\sum_{j=1}^{i-1} p_j < u \leq \sum_{j=1}^{i} p_j$
5:     $\mathbf{d}_i' \leftarrow$ SAMPLE($\mathcal{E}, \theta_i(\mathbf{d}_i)$)
6:     return $\langle P_i, \mathbf{d}_i' \rangle_{\mathcal{E}}$
7: end function
```

Fig. 4: Functions used to simulate behaviour of a configuration.

$$\begin{aligned}
&= \sum_{(\theta,P')\in\mathsf{supp}(\mathsf{pstep}(P,\mathbf{d}))} (\mathsf{pstep}(P,\mathbf{d})(\theta,P') \cdot \langle P', \mathcal{E}(\theta(\mathbf{d}))\rangle_{\mathcal{E}})\,(\mathbb{C})\\
&= \sum_{i=1}^{n} (p_i \cdot \langle P_i, \mathcal{E}(\theta_i(\mathbf{d}))\rangle_{\mathcal{E}})\,(\mathbb{C})\\
&= \sum_{i=1}^{n} p_i \cdot \langle \mathbb{1}_{\mathbb{P}}\{P_i\}, \mathcal{E}(\theta_i(\mathbf{d}))(\mathbb{D})\rangle_{\mathcal{E}}\\
&= \sum_{\{i \mid P_i \in \mathbb{P}\}} p_i \cdot \mathcal{E}(\theta_i(\mathbf{d}))(\mathbb{D})\\
&= \sum_{\{i \mid P_i \in \mathbb{P}\}} p_i \cdot Pr(\textsc{Sample}(\mathcal{E}, \theta_i(\mathbf{d})) \in \mathbb{D})\\
&= Pr(\textsc{SimStep}(\langle P, \mathbf{d}\rangle_{\mathcal{E}}) \in \mathbb{C})
\end{aligned}$$

where

- The first step follows by the definition of cstep (Definition 7).
- The second step follows by $\mathsf{pstep}(P, \mathbf{d}) = p_1 \cdot (\theta_1, P_1) + \ldots + p_n \cdot (\theta_n, P_n)$, for some p_i, θ_i and P_i (see Proposition 1).
- The third step follows by the definition of the probability measure $\langle \mu_{\mathcal{P}}, \mu_{\mathcal{D}} \rangle_{\mathcal{E}}$ (Notation 3), in which $\mathbb{1}_{\mathbb{P}}$ denotes the characteristic function of the set $\mathbb{P}$, i.e., $\mathbb{1}_{\mathbb{P}}\{P'\} = 1$ if $P' \in \mathbb{P}$ and $\mathbb{1}_{\mathbb{P}}\{P'\} = 0$ otherwise.
- The fifth step follows by the assumption on the function SAMPLE(_) in Equation (7).
- The last step follows by the definition of function SIMPSTEP(_).

E Proof of Theorem 1

Theorem 1. *Let $\mathcal{S}^{\mathcal{C}}_{c_1,i}, \mathcal{S}^{\mathcal{C}}_{c_2,i} \in \Delta(\mathcal{C}, \Sigma_{\mathcal{C}})$ be unknown, and ρ be a penalty function. Let $\{\omega_j = \rho_i(\mathbf{d}_1^j)\}$ and $\{\nu_h = \rho_i(\mathbf{d}_2^h)\}$ be the ordered sequences obtained from independent samples taken from $\mathcal{S}^{\mathcal{C}}_{c_1,i}$ and $\mathcal{S}^{\mathcal{C}}_{c_2,i}$, respectively. Then, it holds, a.s., that* $\mathbf{W}(m^{\mathcal{D}}_{\rho,i})(\mathcal{S}^{\mathcal{D}}_{c_1,i}, \mathcal{S}^{\mathcal{D}}_{c_2,i}) = \lim_{N\to\infty} \frac{1}{\ell N} \sum_{h=1}^{\ell N} \max\{\nu_h - \omega_{\lceil \frac{h}{\ell} \rceil}, 0\}$.

Proof. We split the proof into two parts showing respectively:

$$\mathbf{W}(m^{\mathcal{D}}_{\rho,i})(\mathcal{S}^{\mathcal{D}}_{c_1,i},\mathcal{S}^{\mathcal{D}}_{c_2,i}) = \lim_{N\to\infty} \mathbf{W}(m^{\mathcal{D}}_{\rho,i})(\hat{\mathcal{S}}^{\mathcal{D},N}_{c_1,i},\hat{\mathcal{S}}^{\mathcal{D},\ell N}_{c_2,i}) \ . \tag{8}$$

$$\mathbf{W}(m^{\mathcal{D}}_{\rho,i})(\hat{\mathcal{S}}^{\mathcal{D},N}_{c_1,i},\hat{\mathcal{S}}^{\mathcal{D},\ell N}_{c_2,i}) = \frac{1}{\ell N}\sum_{h=1}^{\ell N} \max\left\{\nu_h - \omega_{\lceil \frac{h}{\ell}\rceil}, 0\right\} \ . \tag{9}$$

where $\hat{\mu}^N$ and $\hat{\nu}^{\ell N}$ are the estimated probability distributions obtained from μ and ν by sampling N and ℓN values.

- PROOF OF EQUATION (8).
  We recall that the sequence $\{\hat{\mu}^N\}$ (resp. $\{\hat{\nu}^N\}$) converges weakly to μ (resp. ν) (see Equation (6)). Moreover, we can prove that these sequences converge weakly in $\Delta(\mathcal{D},\mathcal{B}_{\mathcal{D}})$ in the sense of [23, Definition 6.8]. In fact, given the i-ranking function ρ_i, the existence of a data state $\tilde{\mathbf{d}}$ such that $\rho_i(\tilde{\mathbf{d}}) = 0$ is guaranteed (remember that the constraints used to define ρ_i are on the possible values of state variables and a data state fulfilling all the requirements is assigned value 0). Thus, for any $\mathbf{d} \in \mathcal{D}$ we have that

  $$m^{\mathcal{D}}_{\rho,i}(\tilde{\mathbf{d}},\mathbf{d}) = \max\{\rho_i(\mathbf{d}) - \rho_i(\tilde{\mathbf{d}}), 0\} = \rho_i(\mathbf{d}) \ .$$

  Since, moreover, by definition ρ_i is continuous and bounded, the weak convergence of the probability measures gives

  $$\int_{\mathcal{D}} \rho_i(\mathbf{d})\mathrm{d}(\hat{\mathcal{S}}^{\mathcal{D},N}_{c_1,i}(\mathbf{d})) \to \int_{\mathcal{D}} \rho_i(\mathbf{d})\mathrm{d}(\mathcal{S}^{\mathcal{D}}_{c_1,i}(\mathbf{d}))$$
  $$\int_{\mathcal{D}} \rho_i(\mathbf{d})\mathrm{d}(\hat{\mathcal{S}}^{\mathcal{D},\ell N}_{c_2,i}(\mathbf{d})) \to \int_{\mathcal{D}} \rho_i(\mathbf{d})\mathrm{d}(\mathcal{S}^{\mathcal{D}}_{c_2,i}(\mathbf{d}))$$

  and thus Definition 6.8.(i) of [23] is satisfied. As $\mathcal{D}$ is a Polish space, by [23, Theorem 6.9] we obtain that

  $$\hat{\mu}^N \to \mu \text{ and } \hat{\nu}^{\ell N} \to \nu \text{ implies } \mathbf{W}(m^{\mathcal{D}}_{\rho,i})(\mu,\nu) = \lim_{N\to\infty} \mathbf{W}(m^{\mathcal{D}}_{\rho,i})(\hat{\mu}^N,\hat{\nu}^{\ell N}) \ .$$

- PROOF OF EQUATION (9).
  For this part of the proof we follow [21]. Since the ranking function is continuous, it is in particular $\mathcal{B}_{\mathcal{D}}$ measurable and therefore for any probability measure μ on $(\mathcal{D},\mathcal{B}_{\mathcal{D}})$ we obtain that

  $$F_{\mu,\rho_i}(r) := \mu(\{\rho_i(\mathbf{d}) < r\})$$

  is a well defined cumulative distribution function. In particular, for $\mu = \hat{\mathcal{S}}^{\mathcal{D},N}_{c_1,i}$ we have that

  $$F_{\hat{\mathcal{S}}^{\mathcal{D},N}_{c_1,i},\rho_i}(r) = \hat{\mathcal{S}}^{\mathcal{D},N}_{c_1,i}(\{\rho_i(\mathbf{d}) < r\}) = \frac{\left|\{\langle P^j_1, \mathbf{d}^j_1\rangle_{\mathcal{E}_1} \in E_{1,i} \mid \rho_i(\mathbf{d}^j_1) < r\}\right|}{N} \ .$$

Since, moreover, we can always assume that the values $\rho_i(\mathbf{d}_1^j)$ are sorted, so that $\rho_i(\mathbf{d}_i^j) \leq \rho_i(\mathbf{d}_1^{j+1})$ for each $j = 1, \dots, N-1$, we can express the counter image of the cumulative distribution function as

$$F^{-1}_{\hat{\mathcal{S}}^{\mathcal{D},N}_{c_1,i},\rho_i}(r) = \rho_i(\mathbf{d}_1^j) \text{ whenever } \frac{j-1}{N} < r \leq \frac{j}{N} \quad . \tag{10}$$

A similar reasoning holds for $F^{-1}_{\hat{\mathcal{S}}^{\mathcal{D},\ell N}_{c_2,i},\rho_i}(r)$.
Then, by [7, Proposition 6.2], for each N we have that

$$\mathbf{W}(m^{\mathcal{D}}_{\rho,i})(\hat{\mathcal{S}}^{\mathcal{D},N}_{c_1,i}, \hat{\mathcal{S}}^{\mathcal{D},\ell N}_{c_2,i}) = \int_0^1 \max\left\{F^{-1}_{\hat{\mathcal{S}}^{\mathcal{D},\ell N}_{c_2,i},\rho_i}(r) - F^{-1}_{\hat{\mathcal{S}}^{\mathcal{D},N}_{c_1,i},\rho_i}(r), 0\right\} \mathrm{d}r \enspace .$$

Let us now partition the interval $[0,1]$ into ℓN intervals of size $\frac{1}{\ell N}$, thus obtaining

$$\mathbf{W}(m^{\mathcal{D}}_{\rho,i})(\hat{\mathcal{S}}^{\mathcal{D},N}_{c_1,i}, \hat{\mathcal{S}}^{\mathcal{D},\ell N}_{c_2,i}) = \sum_{h=1}^{\ell N}\left(\int_{\frac{h-1}{\ell N}}^{\frac{h}{\ell N}} \max\left\{F^{-1}_{\hat{\mathcal{S}}^{\mathcal{D},\ell N}_{c_2,i},\rho_i}(r) - F^{-1}_{\hat{\mathcal{S}}^{\mathcal{D},N}_{c_1,i},\rho_i}(r), 0\right\} \mathrm{d}r\right) \enspace .$$

From Equation (10), on each interval $(\frac{h-1}{\ell N}, \frac{h}{\ell N}]$ it holds that $F^{-1}_{\hat{\nu}^N,\rho_i}(r) = \rho_i(\mathbf{d}_1^{\lceil \frac{h}{\ell} \rceil})$ and $F^{-1}_{\hat{\nu}^{\ell N},\rho_i}(r) = \rho_i(\mathbf{d}_2^h)$. Moreover, both functions are constant on each the interval so that the value of the integral is given by the difference multiplied by the length of the interval:

$$\begin{aligned}\mathbf{W}(m^{\mathcal{D}}_{\rho,i})(\hat{\mathcal{S}}^{\mathcal{D},N}_{c_1,i}, \hat{\mathcal{S}}^{\mathcal{D},\ell N}_{c_2,i}) &= \sum_{h=1}^{\ell N} \frac{1}{\ell N} \max\left\{\rho_i(\mathbf{d}_2^h) - \rho_i(\mathbf{d}_1^{\lceil \frac{h}{\ell} \rceil}), 0\right\} \\ &= \sum_{h=1}^{\ell N} \frac{1}{\ell N} \max\left\{\nu_h - \omega_{\lceil \frac{h}{\ell} \rceil}, 0\right\} \enspace .\end{aligned}$$

By substituting the last equality into Equation (8) we obtain the thesis.