

# Evaluating the Effectiveness of Digital Twins Through Statistical Model Checking with Feedback and Perturbations

**Citation for published version (APA):**

Castiglioni, V., Lanotte, R., Loreti, M., & Tini, S. (2024). Evaluating the Effectiveness of Digital Twins Through Statistical Model Checking with Feedback and Perturbations. In A. E. Haxthausen, & W. Serwe (Eds.), *Formal Methods for Industrial Critical Systems: 29th International Conference, FMICS 2024, Milan, Italy, September 9–11, 2024, Proceedings* (pp. 21-39). (Lecture Notes in Computer Science (LNCS); Vol. 14952). Springer. [https://doi.org/10.1007/978-3-031-68150-9\\_2](https://doi.org/10.1007/978-3-031-68150-9_2)

**Document license:**

TAVERNE

**DOI:**

[10.1007/978-3-031-68150-9\\_2](https://doi.org/10.1007/978-3-031-68150-9_2)

**Document status and date:**

Published: 21/08/2024

**Document Version:**

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

**Please check the document version of this publication:**

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

**General rights**

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

[www.tue.nl/taverne](http://www.tue.nl/taverne)

**Take down policy**





If you believe that this document breaches copyright please contact us at:

[openaccess@tue.nl](mailto:openaccess@tue.nl)

providing details and we will investigate your claim.



# Evaluating the Effectiveness of Digital Twins Through Statistical Model Checking with Feedback and Perturbations

Valentina Castiglioni<sup>1</sup>, Ruggero Lanotte<sup>2</sup>, Michele Loreti<sup>3</sup>,  
and Simone Tini<sup>2</sup>

<sup>1</sup> Eindhoven University of Technology, Eindhoven, The Netherlands  
v.castiglioni@tue.nl

<sup>2</sup> University of Insubria, Como, Italy

<sup>3</sup> University of Camerino, Camerino, Italy

**Abstract.** We present DT-STARK, an extension of the STARK tool aimed at the verification and evaluation of the *effectiveness* of digital twins, i.e., their ability to direct the physical counterparts. To this end, we introduce *feedback* in STARK, a special mechanism that allow us to model the communications, and their effects, between the digital and the physical (perturbed) twin in a concise, clean fashion. We can then exploit the features of STARK to compare the behaviour of the twins, to verify properties over them, and to measure effectiveness. We provide some examples of the use of our tool by applying it to the evaluation of the effectiveness of digital twins in two robotic scenarios.

## 1 Introduction

Digital Twins (DT) [23,26,48], introduced over two decades ago, promised a revolutionary approach to system development, offering real-time monitoring, simulation, optimisation, and accurate prediction. This innovative approach has received considerable attention across various sectors, including smart cities, manufacturing, healthcare, and automotive industries (see, e.g., [9,25,31,41,50,51]).

The essence of DT lies in creating a virtual replica of physical systems, facilitating real-time interaction between virtual and physical counterparts to enhance decision-making processes and ensure coherent system execution. This approach aims to reduce failures, costs, and time, enhancing user safety and system efficiency [23,24]. Hence, the main objective is to closely represent the physical counterparts, aiming to anticipate system behaviours and outcomes, incorporating simulation models for predictive analysis [20,21]. We formalise this intuition in the notion of *effectiveness* of the DT, a measure of how well the DT can direct the physical counterpart. However, challenges arise due to inherent disparities between the Physical Twin (PT) and the DT, particularly in decision-making and feedback processes (see e.g., [49]). Factors such as environmental variations, sensor and actuator imprecision, challenges in building faithful and predictive models, are all limiting the predictive precision of DT, thus affecting its effectiveness. Hence, connecting modelling to the real world highlights the imperative for thorough validation, ensuring fidelity. For DT, this becomes fundamental. Therefore,

validation and verification [6, 44] represent the foundations in the development of digital models such as DT. Underlining this process is the recognition that the validity of the model depends on specific experimental conditions and defined objectives. This calls for a quantitative approach to validation and verification, rather than a qualitative one. Statistical model checking techniques [2, 35] allow us to establish whether the probability that a given system satisfies a given property is above or below a given threshold, or to evaluate the probability of satisfaction. Hence, we can apply them to DT to express a credibility percentage that reflects the nature of model evaluation. However, to the best of our knowledge, statistical model checking is usually employed to verify correctness properties, such as safety [47], of digital models without their physical counterparts. Indeed, those properties are examined under a degree of uncertainty and noise, but the focus is solely on evaluating the robustness of the digital model, rather than the *effectiveness* of the DT. This paper provides a first step towards filling this gap.

**Our Contribution: DT-STARK.** We introduce DT-STARK ([https://github.com/quasylab/jsppear/tree/digital\\_twins](https://github.com/quasylab/jsppear/tree/digital_twins)), an extension of the *Software Tool for the Analysis of Robustness in the unKnown environment* (STARK) [14, 17], as a tool for the formal verification and evaluation of the effectiveness of a DT when the PT is operating in an unpredictable environment and with uncertain data.

STARK, developed for Cyber-Physical Systems, is based on the *evolution sequence model* [11, 13], for the representation of systems behaviour, and on the *Robustness Temporal Logic (RobTL)* [12, 16], for the specification of properties. Briefly, evolution sequences are sequences of probability measures over sets of application-relevant data, called *data states*, each representing the result of the interaction of the agents (the cyber component) with the environment (the physical component) at a given time step. RobTL is a temporal logic for the evaluation of system robustness, which is specified as a temporal property of *distances* between nominal and perturbed behaviours of a system. RobTL formulae are defined by means of two simple languages: one to specify the effect of *perturbations* over an evolution sequence, and one to specify *distance expressions* between evolution sequences, in particular nominal and perturbed ones. The intuition is that we can use evolution sequences to simulate the behaviour of DT and PT: since the PT is deployed in the environment, its behaviour will be subject to perturbations. Hence, given the evolution sequence representing the behaviour of the DT, its perturbed version will model the behaviour of the PT. The objective of the DT is then to monitor the behaviour of the PT: it uses the data collected from it to run some predictive analysis, and, when necessary, it sends commands to the PT in order to ensure a coherent behaviour and possibly counter the effects of perturbations. However, the current version of STARK does not provide a mechanism allowing us to model this kind of communication.

We remedy this problem by introducing the *feedback* mechanism. Briefly, the feedback will be attached to the agent (the controller) of the PT, so that it will have direct access to the data related to its behaviour. The feedback will also have access to the evolution sequence of the DT so that it can simulate the decision-making procedure based on the current data. The communication between the twins is then simulated by letting the feedback apply the effects of those decisions directly to the data states. Indeed, the feedback mechanism allows us to abstract from the implementation details of the

communication between the two twins, in favour of a clear representation of its effects on the behaviour of the PT. We leave as future work the development of an accurate, detailed communication mechanism in DT-STARK.

We showcase the features of our approach by providing two case studies on a robotic scenario inspired by industrial and healthcare applications.

**Originality and Relevance.** To the best of our knowledge, DT-STARK is the only existing tool for the analysis of effectiveness, and formal verification, of DT. Indeed, while the novel feedback mechanism is key to encode the communication between twins, the smoothness and efficiency of the analysis are due to the attributes of STARK. The unique features of RobTL and evolution sequences, that make (DT-)STARK so versatile and widely applicable, play a significant role also in the application context of DT.

*RobTL.* RobTL allows us to express properties of distances between nominal and perturbed evolution sequences over a finite time horizon, by also providing the means to specify the perturbations and the distances. To the best of our knowledge, RobTL is the only existing temporal logic that compares the behaviour of *two systems* to evaluate the validity of formulae. Usually, in the quantitative setting, we have that properties are specified over *a single trajectory* of a system, or across trajectories *of the same* system. The former case corresponds to the classic approach of PCTL, probabilistic LTL, and their variants [5, 33, 46], and to those in [19, 22], where a single trajectory of a single system is compared to the set of the behaviours that satisfy a given property (specified in a suitable temporal logic, like, e.g., STL). The latter case is the *hyperproperty* [18] approach of, e.g., HyperPCTL [1] or HPSTL [4], where one can express quantitative dependencies, in the form of bounds on probabilistic weights, between *different independent trajectories* of the system. With RobTL we can specify properties based on the comparison of *all possible trajectories* of *two different systems*. This is a key feature in the evaluation of the effectiveness of a DT. By means of RobTL, and thus of DT-STARK, we can easily measure the distance between the intended behaviour of the system (i.e., that of the DT) and that of the PT when it is subject to perturbations while under control of the DT. If the distance is below a desired threshold, then we can establish that the feedback mechanism, i.e., the interaction with the DT, is effective.

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

The reason why we chose this model over classical and more established ones, like Labelled Markov Chains and Stochastic Hybrid Systems [10, 30], is purely technical. The most prominent consequence of the design choices in this model is that the behaviour of the system is not given by a set of traces/trajectories, but by the combination of their effects. This means a property of an evolution sequence takes into account

all potential behaviours of the system. This is fundamental in the validation and verification of DT [37], since even the slightest modification induced by uncertainty on behaviour is taken into account.

## 2 Case Studies

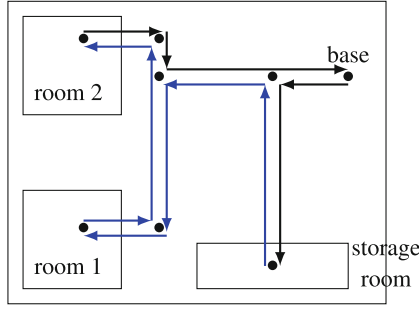
To explain how to apply our approach, and showcase its functionalities, we consider two instances of a simple robotic scenario, in which a single TURTLE-like robot [38] has to follow a determined path inside a working area, by going from one waypoint to another. The working area can be an industrial plant, or a warehouse, or even a hospital. We use waypoints to model both, the fact that the robot has to follow a determined path (e.g., to reduce the risk of collisions in a fully automated plant), and that it has to reach specific locations (e.g., to receive/deliver materials, goods, medicines, etc.).

The scripts with the experiments conducted on the case studies (Sect. 5 below) are available at [https://github.com/quasylab/jspear/blob/digital\\_twins/examples/turtle/](https://github.com/quasylab/jspear/blob/digital_twins/examples/turtle/).

**The Robot.** Our focus in this paper is not to determine how a DT takes its decisions, but rather to show how to use formal verification methods to measure the effectiveness of the interaction between the two twins. For this reason, we abstract from several implementation details of the robot, and opt for a simple model of its behaviour. In particular, we chose to use the TURTLE robot, developed by Tech United Eindhoven at the Eindhoven University of Technology, as a reference, due to its dimensions (making it suitable for both, moving around without disrupting other operations, and transporting some material), and its swerve drive platform [32] that allows it to rotate around its vertical axis, and to move on uneven surfaces with a lower risk of getting stuck or slipping away with respect to other platforms. These features will allow us to further simplify the mathematical modelling in the various scenarios, thus favouring a clearer analysis of behaviour. For instance, we do not implement the selection procedure of waypoints, and we assume that obstacle avoidance has already been taken into account in that: the path between two waypoints is always clear. Hence, the controller of the robot will be responsible for managing the acceleration actuator, to set the moving speed, and the wheels actuator, to set the direction. Specifically, when in a waypoint location, the robot first sets its direction towards the next waypoint, and rotates accordingly. At the next time step, the robot starts moving: the controller decides, with a certain frequency, whether to accelerate or to brake based on its current (evaluated) position and the required braking distance, given the current position and (sensed) speed.

**Industrial Plant.** The first case study concerns the use of the robot to transport some material inside an industrial plant. We will make use of this case study as a running example to highlight all the features of DT-STARK. Hence, we will give a detailed description of it in the examples throughout the paper, and we limit ourselves to remark here that the main challenges for the DT will consist in:

1. Ensuring that the robot actually stops in proximity of each waypoint. In this way, it can correctly receive or deliver the material.
2. Counter the effects of perturbations on the speed sensor of the robot.



**Fig. 1.** Sketch of the hospital. In blue the path to be followed when loaded. (Color figure online)

**Smart Hospital.** The recent pandemic led to the development of intelligent systems, like autonomous robots, that can safely traverse physical spaces hosting patients to sanitise a room, or to deliver medicines, instruments, etc. [40].

Our second case study is inspired by this setting, and it is sketched in Fig. 1. The robot starts from its base, and has to reach the storage room to receive some medicines and instruments that have to be delivered in two rooms. To do so, the robot has to follow a given path along the corridors of the hospital. We will use perturbations to mimic the fact that the robot will have to modify its current direction in order to avoid obstacles, like people walking in the same corridor. Hence, the DT will have the following tasks:

1. Ensure that the robot reaches the desired waypoints.
2. Ensure that the medicines and instruments that are transported are not damaged by changes in the direction made at “high” speed.
3. Ensure that the delivery is completed.

### 3 The STARK Tool

In this section we describe the evolution sequence model and RobTL.

#### 3.1 The Evolution Sequence Model

We recall the main elements of the evolution sequence model [13]. Systems consist of a set of *agents*, modelling the cyber component, and an *environment*, modelling the physical component, whose interaction produces changes on a *data space*  $\mathcal{D}$ , containing the values assumed by *variables* from a *finite* set  $\mathcal{V}$ , representing: (i) physical quantities, (ii) sensors, (iii) actuators, and (iv) internal variables of agents.

For each  $x \in \mathcal{V}$ , the domain  $\mathcal{D}_x \subseteq \mathbb{R}$  is either *finite*, or a *compact* subset of  $\mathbb{R}$  (and thus Polish), and equipped with the Borel  $\sigma$ -algebra  $\mathcal{B}_x$ . Then,  $\mathcal{D} = \prod_{x \in \mathcal{V}} \mathcal{D}_x$  and we equip it with the product  $\sigma$ -algebra  $\mathcal{B}_{\mathcal{D}} = \prod_{x \in \mathcal{V}} \mathcal{B}_x$  [8]. Let  $\Pi(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$  be the set of distributions over  $(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$ . We call *data state* the current state of the data space, and represent it by a mapping  $\mathbf{d}: \mathcal{V} \rightarrow \mathbb{R}$ , with  $\mathbf{d}(x) \in \mathcal{D}_x$  for all  $x \in \mathcal{V}$ .

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 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, we model the changes induced at each step as a distribution on the attainable data states. We can assume two measurable functions  $\text{ag}, \text{env}: \mathcal{D} \rightarrow \Pi(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$  defining, respectively, the effects on data states of the agents and of the environment (see [13] for the details). In particular,  $\text{ag}$  maps each data state onto a discrete distribution, whereas  $\text{env}$  maps them onto continuous distributions. The behaviour of the system is then expressed by its *evolution sequence*, i.e., the sequence of the distributions obtained at each step by combining the effects of  $\text{ag}$  with those of  $\text{env}$ . In [13, Prop. 3.15] we proved that the function  $\text{env}(\text{ag}(\mathbf{d}))(\mathbb{D}) = \sum_{\mathbf{d}' \in \text{supp}(\text{ag}(\mathbf{d}))} \text{ag}(\mathbf{d})(\mathbf{d}') \cdot \text{env}(\mathbf{d}')(\mathbb{D})$  is a well defined *Markov kernel*, and the evolution sequence is the *Markov process* generated by it:  $\text{env}(\text{ag}(\mathbf{d}))(\mathbb{D})$  expresses the probability to reach a data state in  $\mathbb{D}$  from  $\mathbf{d}$  in one computation step. Indeed, each system is characterised by a kernel  $\text{env}(\text{ag})$  starting from an *initial distribution* over  $\mathcal{D}$ .

**Definition 1.** *The evolution sequence of a system  $s$  having  $\mu$  as initial distribution is a countable sequence  $\mathcal{S}_{\mu} = \mathcal{S}_{\mu}^0, \mathcal{S}_{\mu}^1, \dots$  of distributions in  $\Pi(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$  s.t., for all  $\mathbb{D} \in \mathcal{B}_{\mathcal{D}}$ :*

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

*Example 1.* We briefly describe the behaviour of the controller of the robot, and of the environment in the industrial case study. We assume that the robot is initially still, i.e., its physical and sensed speed (respectively, `p_speed` and `s_speed`) are set to 0.0 m/s, and has to set its direction  $\theta$ , i.e., the value of the wheels actuators, towards the first waypoint. The robot turns during the first time step, and will start moving afterwards. To do so, the controller has to set the value of the acceleration actuator `acc`. For simplicity, we assume that the robot can either accelerate, with a constant positive acceleration  $A \text{ m/s}^2$ , or brake, with a constant negative acceleration  $-B \text{ m/s}^2$ . The decision of whether to accelerate or brake, is taken by the controller every `TIMER` steps, on the basis of both the speed of the robot and its `gap` from the next waypoint. The `gap` is given by the difference between the physical distance `p_distance` from the waypoint, and the braking distance of the robot, i.e., the space required to stop if the robot starts braking immediately. Since the objective of the robot is to stop as close as possible to the established waypoint, it starts braking as soon as the `gap` is negative. Once the waypoint is reached, the procedure described above is repeated to reach the next waypoint. The position  $(x, y)$  of the robot, its speed, and the `gap` from the next waypoint are updated by the environment, at each time step, according to the motion laws and the values of the actuators.

### 3.2 RobTL

*Robustness Temporal Logic (RobTL)* [12, 16] is the only temporal logic allowing one to express temporal properties of distances over systems behaviours. It uses atomic

propositions of the form  $\Delta(\text{exp}, \mathbf{p}) \bowtie \eta$  to evaluate, at a given time step, the *distance*, specified by an *expression*  $\text{exp}$ , between a given evolution sequence and its perturbed version, obtained by some *perturbation*  $\mathbf{p}$ , and to compare it with the threshold  $\eta$ . Atomic propositions are then combined with classic Boolean and temporal operators, in order to extend and compare these evaluations over the chosen time horizon. Hence, there are three main components constituting RobTL formulae: 1. A language  $\text{Exp}$  to specify *distance expressions*; 2. A language  $\text{P}$  to specify *perturbations*; 3. Classic Boolean and temporal operators to specify requirements on the *evolution of distances in time*.

**Distance Expressions.** We use expressions in  $\text{Exp}$  to define distances over evolution sequences. The idea is to introduce a *distance over distributions on data states* measuring their differences with respect to a *given target*, and then use the operators of the logic to extend it to the evolution sequences, while possibly taking into account *different targets and perturbations over time*. Following [11], to capture a particular task, we use a *rank function*  $\rho: \mathcal{D} \rightarrow [0, 1]$  assigning to each data state  $\mathbf{d}$  a rank in  $[0, 1]$ . These ranks are used to define a ground distance on data states, expressing how much they differ with respect to the point of view identified by the rank. Then we use  $\rho$  to obtain a *distance on data states*, i.e., the 1-bounded *metric*  $m_\rho$  defined for all  $\mathbf{d}_1, \mathbf{d}_2 \in \mathcal{D}$  by:

$$m_\rho(\mathbf{d}_1, \mathbf{d}_2) = |\rho(\mathbf{d}_2) - \rho(\mathbf{d}_1)|$$

Note that  $m_\rho(\mathbf{d}_1, \mathbf{d}_2)$  expresses the distance between  $\mathbf{d}_2$  and  $\mathbf{d}_1$  according to  $\rho$ . Then, we need to lift the metric  $m_\rho$  to a metric over  $\Pi(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$ . To this end, we make use of the *Wasserstein lifting* [52]: for any two distributions  $\mu, \nu$  on  $(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$ , the Wasserstein lifting of  $m_\rho$  to a distance between  $\mu$  and  $\nu$  is defined by

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

where  $\mathfrak{W}(\mu, \nu)$  is the set of the couplings of  $\mu$  and  $\nu$ , 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  $\mu$  and  $\nu$  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  $\mathbb{D} \in \mathcal{B}(\mathcal{D})$ .

**Definition 2 (Distance expressions).** *Expressions in  $\text{Exp}$  are defined as follows:*

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

where  $\rho$  ranges over rank functions,  $I$  is an interval,  $K$  is a finite set of indexes,  $w_k \in (0, 1]$  for each  $k \in K$ ,  $\sum_{k \in K} w_k = 1$ , and  $\zeta \in [0, 1]$ .

Distance expressions are evaluated over a pair of evolution sequences and a time step: given two evolution sequences  $\mathcal{S}_1, \mathcal{S}_2$  and a time step  $\tau$ , the evaluation of expressions in the triple  $\mathcal{S}_1, \mathcal{S}_2, \tau$  is given by function  $\llbracket \cdot \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^\tau: \text{Exp} \rightarrow [0, 1]$  which is defined inductively over expressions as follows:

- $\llbracket \langle \rho \rangle \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^\tau = \mathbf{W}(m_\rho^\tau)(\mathcal{S}_1^\tau, \mathcal{S}_2^\tau)$ ;
- $\llbracket \mathbf{F}^I \text{exp} \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^\tau = \min_{t \in I + \tau} \llbracket \text{exp} \rrbracket_{\mathcal{S}_1, \mathcal{S}_2}^t$ ;

- $\llbracket G^I \text{exp} \rrbracket_{S_1, S_2}^\tau = \max_{t \in I + \tau} \llbracket \text{exp} \rrbracket_{S_1, S_2}^t$ ;
- $\llbracket \text{exp}_1 U^I \text{exp}_2 \rrbracket_{S_1, S_2}^\tau = \min_{t \in I + \tau} \max \left\{ \llbracket \text{exp}_2 \rrbracket_{S_1, S_2}^t, \max_{t' \in I + \tau, t' < t} \llbracket \text{exp}_1 \rrbracket_{S_1, S_2}^{t'} \right\}$ ;
- $\llbracket \min(\text{exp}_1, \text{exp}_2) \rrbracket_{S_1, S_2}^\tau = \min \{ \llbracket \text{exp}_1 \rrbracket_{S_1, S_2}^\tau, \llbracket \text{exp}_2 \rrbracket_{S_1, S_2}^\tau \}$ ;
- $\llbracket \max(\text{exp}_1, \text{exp}_2) \rrbracket_{S_1, S_2}^\tau = \max \{ \llbracket \text{exp}_1 \rrbracket_{S_1, S_2}^\tau, \llbracket \text{exp}_2 \rrbracket_{S_1, S_2}^\tau \}$ ;
- $\llbracket \sum_{k \in K} w_k \text{exp}_k \rrbracket_{S_1, S_2}^\tau = \sum_{k \in K} w_k \cdot \llbracket \text{exp}_k \rrbracket_{S_1, S_2}^\tau$ ;
- $\llbracket \sigma(\text{exp}, \bowtie \zeta) \rrbracket_{S_1, S_2}^\tau = \begin{cases} 0 & \text{if } \llbracket \text{exp} \rrbracket_{S_1, S_2}^\tau \bowtie \zeta, \\ 1 & \text{otherwise.} \end{cases}$

*Example 2.* The rank function  $\rho_E(\mathbf{d}) = \sqrt{(x - \text{wp}_x)^2 + (y - \text{wp}_y)^2} / \max_E$  assigns to a data state  $\mathbf{d}$  of the robot a rank in  $[0, 1]$  corresponding to the normalised value of its distance from the current waypoint  $(\text{wp}_x, \text{wp}_y)$ . Here,  $\max_E$  is the maximal distance from the current waypoint that can be observed in the executions of the system. This value is estimated by employing the simulation module of DT-STARK. Intuitively, given two data states  $\mathbf{d}_1$  and  $\mathbf{d}_2$  reached at the same step in two different evolutions,  $|\rho_E(\mathbf{d}_1) - \rho_E(\mathbf{d}_2)|$  measures how much one of the two evolution is closer to the next target with respect to the other. Based on  $\rho_E$  we define the atomic distance  $\prec^{\rho_E}$  mapping two distributions to the Wasserstein distance [52] between their ranks, and the distance expression  $\text{dMax}(u, v) = G^{[u, v]} \prec^{\rho_E}$  returning the maximum of  $\prec^{\rho_E}$  over the interval  $[u, v]$ . Notice that, as the ranks, both  $\prec^{\rho_E}$  and  $\text{dMax}(u, v)$  are reals in  $[0, 1]$ .

**Perturbations.** A perturbation is the effect of unpredictable events, that can be repeated or different in time, on the current state of the system. In [12, 16], a perturbation is therefore modelled as a time-dependent function that maps a data state into a distribution over data states. Specifically, a perturbation  $\mathbf{p}$  is a list of mappings in which the  $i$ -th element describes the effects of  $\mathbf{p}$  at time  $i$ , and that is specified in the following language:

**Definition 3 (Perturbations).** *Perturbations in  $\mathcal{P}$  are defined as follows:*

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

where  $\mathbf{p}$  ranges over  $\mathcal{P}$ ,  $n$  and  $\tau$  are finite natural numbers, and:

- $\text{nil}$  is the perturbation with no effects;
- $\mathbf{p} @ \tau$  is an atomic perturbation, i.e., a function  $\mathbf{p}: \mathcal{D} \rightarrow \Pi(\mathcal{D}, \mathcal{B}_{\mathcal{D}})$  such that the mapping  $\mathbf{d} \mapsto \mathbf{p}(\mathbf{d})(\mathbb{D})$  is  $\mathcal{B}_{\mathcal{D}}$ -measurable for all  $\mathbb{D} \in \mathcal{B}_{\mathcal{D}}$ , and that is applied after  $\tau$  time steps from the current instant;
- $\mathbf{p}_1 ; \mathbf{p}_2$  is a sequential perturbation, i.e., perturbation  $\mathbf{p}_2$  is applied at the time step subsequent to the (final) application of  $\mathbf{p}_1$ ;
- $\mathbf{p}^n$  is an iterated perturbation, i.e., perturbation  $\mathbf{p}$  is applied for a total of  $n$  times;
- $\mathbf{p}^\infty$  is a persistent perturbation, i.e.,  $\mathbf{p}$  is applied along the whole evolution.

The semantics of perturbations is defined by means of two auxiliary functions:  $\text{effect}(\mathbf{p})$ , that describes the effect of  $\mathbf{p}$  at the current step, and  $\text{next}(\mathbf{p})$ , that identifies the perturbation that will be applied at the next step. These two functions are defined

inductively over the structure of  $\mathbf{p}$ . Due to space limitations, we omit their formal definition (which is similar to that of functions  $\text{changes}$  and  $\text{succ}$  in Sect. 4.1 below), and refer the interested reader to [12]. We make use of  $\text{effect}$  and  $\text{next}$  to define the mapping  $\langle\langle \cdot \rangle\rangle: \mathbf{P} \rightarrow (\mathcal{D} \times \mathbb{N} \rightarrow \Pi(\mathcal{D}, \mathcal{B}_{\mathcal{D}}))$  such that, for all  $\mathbf{d} \in \mathcal{D}$  and  $i \in \mathbb{N}$ :

$$\langle\langle \mathbf{p} \rangle\rangle(\mathbf{d}, i) = \text{effect}(\text{next}^i(\mathbf{p}))(\mathbf{d}),$$

where  $\text{next}^0(\mathbf{p}) = \mathbf{p}$  and  $\text{next}^i(\mathbf{p}) = \text{next}(\text{next}^{i-1}(\mathbf{p}))$ , for all  $i > 0$ .

Now we can define the perturbation of an evolution sequence.

**Definition 4 (Perturbation of an evolution sequence).** *Given an evolution sequence  $\mathcal{S}_\mu$ , with  $\mu$  as initial distribution, and a perturbation  $\mathbf{p}$ , we define the perturbation of  $\mathcal{S}_\mu$  via  $\mathbf{p}$  as the evolution sequence  $\mathcal{S}_\mu^{\mathbf{p}}$  obtained as follows:*

$$\begin{aligned} \mathcal{S}_\mu^{\mathbf{p},0}(\mathbb{D}) &= \int_{\mathcal{D}} \langle\langle \mathbf{p} \rangle\rangle(\mathbf{d}, 0)(\mathbb{D}) \mathfrak{d} \mu(\mathbf{d}) \\ \mathcal{S}_\mu^{\mathbf{p},i+1}(\mathbb{D}) &= \int_{\mathcal{D}} \left( \int_{\mathcal{D}} \langle\langle \mathbf{p} \rangle\rangle(\mathbf{d}', i+1)(\mathbb{D}) \mathfrak{d} \text{env}(\text{ag}(\mathbf{d}))(\mathbf{d}') \right) \mathfrak{d} \mathcal{S}_\mu^{\mathbf{p},i}(\mathbf{d}), \end{aligned}$$

where function  $\text{env}(\text{ag})$  is the Markov kernel that generates  $\mathcal{S}_\mu$ .

*Example 3.* We consider the perturbation  $\mathbf{p}_{\text{speed},o} = (\mathbf{p}_{\text{speed},o} \text{@}0)^\infty$ , where  $\mathbf{p}_{\text{speed},o}(\mathbf{d})$  is the distribution of the random variable  $O(o, \mathbf{d})$ , for  $o$  uniformly distributed in  $[0, 1]$ , defined for all  $\mathbf{d} \in \mathcal{D}$  by  $O(o, \mathbf{d}) = \mathbf{d}'$ , where  $\mathbf{d}'(\text{s\_speed}) = \mathbf{d}(\text{p\_speed}) - o * \text{MAXOFFSET}$ , for a constant  $\text{MAXOFFSET}$ , and  $\mathbf{d}'(x) = \mathbf{d}(x)$  for all other variables. Note that  $\mathbf{p}_{\text{speed},o}$  is a persistent perturbation. At each step  $\mathbf{p}_{\text{speed},o}$  is applied immediately (this is given by  $\_ \text{@}0$ ) and tricks the controller of the robot by letting it sense a speed which is lower than the real one. This may lead the robot to skip the next waypoint.

**RobTL Formulae.** We use formulae in RobTL for the specification and analysis of distances between nominal and perturbed evolution sequences over a finite time horizon.

**Definition 5 (RobTL).** *RobTL consists in the set of formulae  $\mathbf{L}$  defined by:*

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

where  $\varphi$  ranges over  $\mathbf{L}$ ,  $\text{exp}$  ranges over expressions in  $\text{Exp}$ ,  $\mathbf{p}$  ranges over perturbations in  $\mathbf{P}$ ,  $\bowtie \in \{<, \leq, \geq, >\}$ ,  $\eta \in [0, 1]$ , and  $I \subseteq [0, \mathfrak{h}]$  is a bounded time interval.

Formulae are evaluated in an evolution sequence and a time instant. The semantics of classic Boolean and temporal operators is standard, and it is based on the evaluation of the atomic formulae, which is defined as follows:

$$\mathcal{S}, \tau \models \Delta(\text{exp}, \mathbf{p}) \bowtie \eta \text{ iff } \llbracket \text{exp} \rrbracket_{\mathcal{S}, \mathcal{S}_{|\langle\langle \mathbf{p} \rangle\rangle}, \tau}^\tau \bowtie \eta,$$

where the evolution sequence  $\mathcal{S}_{|\langle\langle \mathbf{p} \rangle\rangle}, \tau$  is defined as:

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

*Example 4.* Consider the distance  $d\text{Max}(u, v)$  in Example 2 and the perturbation  $p_{\text{speed},o}$  in Example 3. A RobTL atomic formula  $\varphi_{(u,v,o,\eta)}$  expressing that the distance  $d\text{Max}(u, v)$  between the nominal behaviour and its perturbation via  $p_{\text{speed},o}$  is below a threshold  $\eta$  can be defined as follows:

$$\varphi_{(u,v,o,\eta)} = \Delta(d\text{Max}(u, v), p_{\text{speed},o}) \leq \eta \quad (1)$$

## 4 DT-STARK

Our objective is to provide the means to measure the effectiveness of a DT when the PT is operating in an unpredictable environment and with uncertain data.

One, naive, approach would consist in a direct encoding of the two twins in STARK: we model the controller of the DT as an agent, and then we simulate its behaviour in a given environment. The behaviour of the DT corresponds, thus, to the nominal evolution sequence. Conversely, to obtain the behaviour of the PT we let the same agent interact with a perturbed environment. We can then use distance expressions, and RobTL formulae, to check how close the behaviour of the two twins is. While this approach gives us a measure of the robustness of the DT against perturbations, it does not allow us to encode the monitoring function of the DT over the physical one: the DT receives data from the PT and, based on simulations and predictions, sends commands to it in order to ensure that its behaviour is as intended.

At the moment, STARK does not offer a similar communication mechanism between systems. Hence, we propose an extension of it, DT-STARK available at [https://github.com/quasylab/jspear/tree/digital\\_twins](https://github.com/quasylab/jspear/tree/digital_twins), that enriches STARK with a special mechanism, henceforth called *feedback*, allowing us to simulate the communication between the two twins, and, thus, to implement the monitoring activity of the DT. We can exploit the features of RobTL to measure the effectiveness of the DT.

### 4.1 The Feedback Mechanism

We use the feedback mechanism to abstract from the implementation details of the communication between the two twins: the feedback has access to the data of the PT and to the behaviour of the DT so that it can modify, up to a certain extent, the behaviour of the former to comply with the one of the latter.

For instance, consider a self-driven vehicle that has to turn. The idea is that the feedback checks for the status of the steering wheel and of the accelerator, and it compares them to their values in the corresponding state of the DT. If the speed of the vehicle is too high given the steering angle, or the angle is not correct, the mechanism will adjust the behaviour of the vehicle by either imposing to decelerate, or to change the steering angle. In DT-STARK, this is obtained by applying the feedback to the perturbed system (modelling the PT), so that it can have access to its data state  $\mathbf{d}$ . The mechanism takes as input the evolution sequence of the nominal system (modelling the DT), and determines whether to take action or not by comparing the values of the variables to be monitored (corresponding to actuators) in the two systems, at the current time step. Indeed, the magnitude of the modifications imposed by the feedback is limited by physical and safety constraints: the speed of the vehicle can only be reduced by a certain

amount per second, depending on the braking force of the vehicle and the conditions of the road; the steering angle cannot be changed abruptly, to limit the risk of a collision or of spinning out. Hence, each variable will only be modified up to a certain percentage.

Following the purely data-driven approach of the evolution sequence model, the communication between the two twins is abstracted by implementing the procedure sketched above as a mapping  $f_S$  over data states: the data state  $f_S(\mathbf{d})$  is obtained by applying to  $\mathbf{d}$  the modifications imposed by the feedback as formalised in  $f_S$ . Specifically, to better encode the communication, the feedback is applied to the data states obtained from an application of the controller  $ag$  of the PT: the DT checks for the current data and the decisions taken by the physical counterpart (given by  $ag(\mathbf{d})$ ) and it reacts to them by sending modification commands. These are abstracted in the application of the feedback to the data states, given by  $f_S(ag(\mathbf{d}))$ . We remark that  $f_S$  can be defined arbitrarily by the user, that can therefore autonomously decide how to use the behaviour  $S$  to establish the modifications. Moreover, we allow for choosing the time step at which the feedback will be applied: we use  $f_S@_\tau$  to denote that  $f_S$  will be applied (to the data states reached) after  $\tau$  steps from the current one. This feature allows us to establish the frequency of the feedback.

Function  $f_S@_\tau$  defines an *atomic feedback*. Since we may wish to apply the same feedback iteratively, or different modifications in time, we also offer the possibility to define *iterated* and *sequential feedback*.

The informal discussion above is formalised in the following definition.

**Definition 6 (Feedback).** *Feedback in  $F$  are defined as follows:*

$$\mathbf{f} ::= \text{id} \mid f_S@_\tau \mid \mathbf{f}_1; \mathbf{f}_2 \mid \mathbf{f}^n \mid \mathbf{f}^\infty$$

where  $\mathbf{f}$  ranges over  $F$ ,  $n, \tau$  are finite natural numbers,  $S$  is an evolution sequence, and:

- $\text{id}$  is the feedback with no effects, i.e., at each time step it behaves like the identity function  $\text{id}: \mathcal{D} \rightarrow \mathcal{D}$  such that  $\text{id}(\mathbf{d}) = \mathbf{d}$  for all  $\mathbf{d} \in \mathcal{D}$ ;
- $f_S@_\tau$  is an atomic feedback, i.e., a function  $f_S: \mathcal{D} \rightarrow \mathcal{D}$  that is applied after  $\tau$  time steps from the current instant;
- $\mathbf{f}_1; \mathbf{f}_2$  is a sequential feedback, i.e., feedback  $\mathbf{f}_2$  is applied at the time step subsequent to the (final) application of  $\mathbf{f}_1$ ;
- $\mathbf{f}^n$  is an iterated feedback, i.e., feedback  $\mathbf{f}$  is applied for a total of  $n$  times;
- $\mathbf{f}^\infty$  is a persistent feedback, i.e., feedback  $\mathbf{f}$  is applied along the whole evolution.

The semantics of feedback is then defined by means of two auxiliary functions:  $\text{changes}(\mathbf{f})$ , that describes the effect of  $\mathbf{f}$  at the current step, and  $\text{succ}(\mathbf{f})$ , that identifies the feedback that will be applied at the next step. Both functions are defined inductively over the structure of  $\mathbf{f}$  as follows:

$$\begin{aligned}
\text{changes}(\text{id}) &= \text{id} & \text{succ}(\text{id}) &= \text{id} \\
\text{changes}(f_S @ \tau) &= \begin{cases} \text{id} & \text{if } \tau > 0 \\ f_S & \text{if } \tau = 0 \end{cases} & \text{succ}(f_S @ \tau) &= \begin{cases} f_S @ (\tau - 1) & \text{if } \tau > 0 \\ \text{id} & \text{otherwise} \end{cases} \\
\text{changes}(f^n) &= \text{changes}(f) & \text{succ}(f^n) &= \begin{cases} \text{succ}(f); f^{n-1} & \text{if } n > 0 \\ \text{id} & \text{otherwise} \end{cases} \\
\text{changes}(f_1; f_2) &= \text{changes}(f_1) & \text{succ}(f_1; f_2) &= \begin{cases} \text{succ}(f_1); f_2 & \text{if } \text{succ}(f_1) \neq \text{id} \\ f_2 & \text{otherwise} \end{cases} \\
\text{changes}(f^\infty) &= \text{changes}(f) & \text{succ}(f^\infty) &= \text{succ}(f); f^\infty.
\end{aligned}$$

As for perturbations, we can identify the semantics of a feedback  $\mathbf{f}$  with the list of its effects in time. Hence, we make use of  $\text{changes}$  and  $\text{succ}$  to define the mapping  $\ulcorner \cdot \urcorner: \mathbf{F} \rightarrow (\mathcal{D} \times \mathbb{N} \rightarrow \mathcal{D})$  such that, for all  $\mathbf{d} \in \mathcal{D}$  and  $i \in \mathbb{N}$ :

$$\ulcorner \mathbf{f} \urcorner(\mathbf{d}, i) = \text{changes}(\text{succ}^i(\mathbf{f}))(\mathbf{d}),$$

where  $\text{succ}^0(\mathbf{f}) = \mathbf{f}$  and  $\text{succ}^i(\mathbf{f}) = \text{succ}(\text{succ}^{i-1}(\mathbf{f}))$ , for all  $i > 0$ .

As briefly discussed above, to model the communication between the two twins, the effects of the feedback are applied to the data states *after* the effects of the agents, and *before* those of the environment. This allows us to simulate the reaction of the DT to the decisions taken by the PT. We abstract this procedure by letting the agent (the PT) set its actuators according to the data at its disposal; the feedback (the DT) compares those values to those assumed in the nominal system, and, if necessary, it modifies them to make the behaviour compliant to the expected one. This double modification is done within the same time step, before the environment reacts to the activity of the agent. Moreover, as a consequence, the feedback has no effect on the initial distribution of an evolution sequence.

**Definition 7 (Feedback of an evolution sequence).** *Given an evolution sequence  $S_\mu$ , with  $\mu$  as initial distribution, and a feedback  $\mathbf{f}$ , we define the feedback of  $S_\mu$  via  $\mathbf{f}$  as the evolution sequence  $S_\mu^{\mathbf{f}}$  obtained as follows:*

$$\begin{aligned}
S_\mu^{\mathbf{f},0}(\mathbb{D}) &= \mu(\mathbb{D}) \\
S_\mu^{\mathbf{f},i+1}(\mathbb{D}) &= \int_{\mathcal{D}} \text{env}(\ulcorner \mathbf{f} \urcorner(\text{ag}(\mathbf{d}), i + 1))(\mathbb{D}) \delta S_\mu^{\mathbf{f},i}(\mathbf{d}),
\end{aligned}$$

where  $\text{env}(\ulcorner \mathbf{f} \urcorner(\text{ag}(\mathbf{d}), i))(\mathbb{D})$  stands for  $\sum_{\mathbf{d}' \in \text{supp}(\text{ag}(\mathbf{d}))} \text{ag}(\mathbf{d})(\mathbf{d}') \cdot \text{env}(\ulcorner \mathbf{f} \urcorner(\mathbf{d}', i))(\mathbb{D})$ .

Thanks to the modular design of the back-end of STARK, the feedback mechanism can be smoothly integrated in it. The main difference between DT-STARK and the original tool is in the module for the simulation of evolution sequences, which is modified to allow for the application of the effects of the feedback. All other modules, i.e., the ones for the application of perturbations, the evaluation of distances, and the statistical model checker for RobTL formulae, then automatically interact with the new simulation method, without the need for further modifications.

*Example 5.* We define the persistent feedback  $\mathbf{f}_{speed\&dir} = (\mathbf{f}_{s\&d} \textcircled{0})^\infty$ , where function  $\mathbf{f}_{s\&d}$  is applied immediately at each step (this is given by  $\textcircled{0}$ ) and is defined as  $\mathbf{f}_{s\&d}(\mathbf{d}) = \mathbf{d}'$ , with:

$$\begin{aligned} - \mathbf{d}'(\text{acc}) &= \begin{cases} -B & \text{if } \mathbf{d}(\text{s\_speed}) > \text{m\_s\_speed} + \text{SpeedD} \\ \mathbf{d}(\text{acc}) & \text{otherwise} \end{cases} \\ - \mathbf{d}'(\text{wp}_x) &= \begin{cases} \text{m\_wp}_x & \text{if } \mathbf{d}(\text{cur\_wp}) < \text{m\_cur\_wp} \\ \mathbf{d}(\text{wp}_x) & \text{otherwise} \end{cases} \\ - \mathbf{d}'(\text{wp}_y) &= \begin{cases} \text{m\_wp}_y & \text{if } \mathbf{d}(\text{cur\_wp}) < \text{m\_cur\_wp} \\ \mathbf{d}(\text{wp}_y) & \text{otherwise} \end{cases} \\ - \mathbf{d}'(\theta) &= \begin{cases} \text{m\_}\theta & \text{if } \text{cur\_wp} < \text{m\_cur\_wp} \\ \mathbf{d}(\theta) & \text{otherwise} \end{cases} \end{aligned}$$

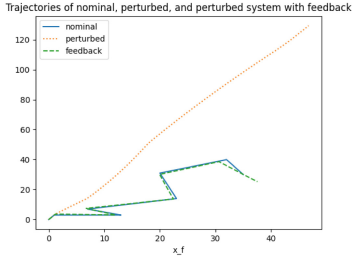
and  $\mathbf{d}'(x) = \mathbf{d}(x)$  for all other variables. Here,  $\text{acc}$  is the actuator of the accelerator,  $(\text{wp}_x, \text{wp}_y)$  are the coordinates of the current waypoint,  $\text{cur\_wp}$  identifies the position of the current waypoint in the list of the waypoints, and  $\theta$  is the steering angle. For each variable  $x$ , the average value assumed by  $x$  in the nominal evolution of DT at the same step is denoted with  $\text{m}_x$ . Intuitively, this feedback forces the robot to slow down if it is too fast, and to focus on the next waypoint if the previous one has been already passed.

## 5 Experiments

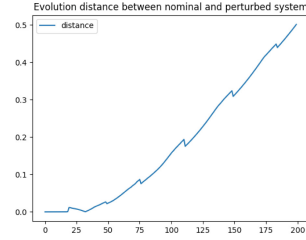
**Industrial Plant.** Consider the distances  $\langle^{\rho^E}$  and  $\text{dMax}(u, v)$  in Example 2, the perturbation  $\mathbf{p}_{speed,o}$  in Example 3, the formula  $\varphi(u, v, o, \eta)$  in Example 4 and the feedback  $\mathbf{f}_{speed\&dir}$  in Example 5. In Fig. 2a we showcase the paths followed by the robot in three scenarios: nominal evolution, perturbed evolution without feedback and perturbed evolution with feedback. The waypoints list is  $(1, 3), (13, 3), (7, 7), (23, 14), (20, 31), (32, 40), (35, 30)$ . The plot shows that the perturbation affects the behaviour of the robot, but the feedback is able to neutralise it. Pointwise evaluations of distance  $\langle^{\rho^E}$  between the nominal and the perturbed behaviour in the scenarios without and with feedback are showed in Fig. 2b and Fig. 2c, respectively. These plots confirms the effectiveness of the DT. Finally, Fig. 2d shows the evaluation of  $\varphi_{(100,200,1.0,\eta)}$  for several values of  $\eta$ .

**Smart Hospital.** Due to space restrictions, we give only a bird's-eye view on the results that we obtained. We refer the interested reader directly to the script at <https://github.com/quasylab/jspear/blob/digital.twins/examples/turtle/> for more details, including the (initial) values of variables and parameters.

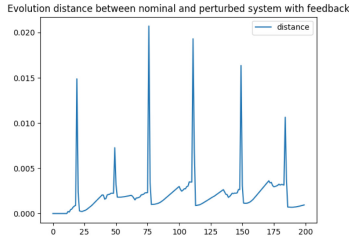
In Fig. 3a we report the path followed by the robot in the nominal system, i.e., the one established by the DT in the absence of perturbations, and the one followed by the physical robot. The latter is subject to the persistent perturbation  $\mathbf{p}_{obs} = (\mathbf{p}_{obs,\delta} \textcircled{0})^\infty$  that simulates continuous changes in the direction due to obstacle avoidance. In fact, function  $\mathbf{p}_{obs,\delta}$  applies an offset  $\delta$  uniformly distributed in  $[-\theta_{\text{OFF}}/2, \theta_{\text{OFF}}/2]$ , for a



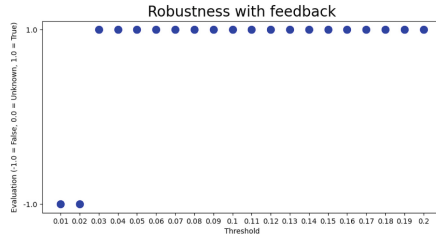
(a) Paths



(b) Distance between nominal and perturbed path



(c) Distance in the feedback case



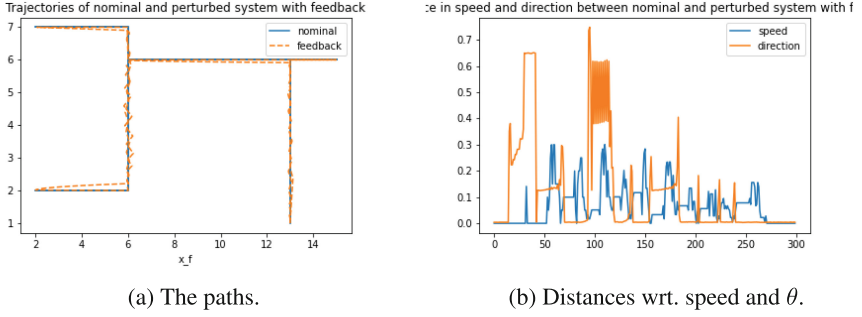
(d) Evaluation of formula in Eq. 1

**Fig. 2.** Analysis of the industrial plant case study.

chosen parameter  $\theta_{\text{OFF}}$ , to the current direction. Indeed, while it is unlikely for the robot to incur in continuous disruptions, we can use this heavy perturbation to show the effectiveness of the feedback mechanism. We recall that, besides guaranteeing that the robot reaches the designed waypoints, the feedback has also to guarantee that the medicines and instruments that it carries are not damaged on the way. Hence, we define the feedback as  $\mathbf{f} = (f_S @ 0)^\infty$  where function  $f_S$  is such that:

1. Whenever the robot is carrying something, encoded by `get_medicine = 1`, then the robot cannot travel at a speed higher than `MAX_SPEED_WITH_MED`. Hence, function  $f_S$  forces the robot to stop accelerating, and thus to gently decelerate, whenever its speed is close to the threshold.
2. Whenever the current direction of the robot deviates too much from the one in the nominal system, the feedback acts on the wheels actuator to point it towards the correct waypoint. Also in this case, the feedback determines the magnitude of the change according to whether the robot is carrying something and its current speed.
3.  $f_S$  behaves like  $f_{s\&d}$  with respect to waypoint selection.

In Fig. 3b we report the evaluation of the atomic distance between the speed, and the direction assumed in the two systems. We remark that the peaks in the difference between the directions are due to temporal delays in reaching the same waypoint.



**Fig. 3.** Analysis of the smart hospital case study.

## 6 Related Work

To the best of our knowledge, statistical model checking, and more broadly, model checking, are utilised for verifying correctness properties, such as safety and security (see e.g. [34]), of digital models without their physical counterparts. Consequently, interactions and feedback are not considered. While some of these works share similarities with our study by examining safety under a degree of uncertainty and noise, they focus solely on evaluating the robustness of the digital model, rather than the effectiveness of the digital twin. For instance, in [3], the authors verify the planned manoeuvres of an automated car on the road using reachable sets. This approach enables the verification of potential collisions with other traffic participants taking in account sensor noise, uncertain friction coefficient, and uncertain initial states.

In the literature, instead of model checking, other techniques for evaluating the discrepancy in valuation of digital twins, such as Bayesian inference, have been studied. These techniques often rely on validation techniques [44, 45], model calibration [43], and conformance checking approaches [39]. For a comprehensive overview, an exhaustive survey can be found in [42]. It's worth noting that recently, [37] introduced the new concept of online validation of digital twins. The techniques proposed in this research enable the continuous updating of a digital model in terms of both inputs and logic, facilitating short-term decision-making. The proposed methodology is subject to several limitations, as, for instance, it is designed for validating digital models with limited data and it is focused on manufacturing system data.

In the literature, we can find several tools for (statistical) model checking and formal verification, like HyTECH [28], PRISM [29], Storm [27], UPPAAL [7], etc. While these tools have been successfully applied to the verification of Cyber-Physical Systems, we could find fewer applications in the context of DT. For instance, in [47], PRISM is used for the analysis of security properties of DT, modelled as Markov Decision Processes. In [36], DT are modelled as time automata networks whose properties are verified in UPPAAL. However, in the former case, only the behaviour of the DT is taken into account. In the latter case, as the paper concerns building DT, the interaction with the PT is considered, but there is no analysis of behaviour in the presence of uncertainty and perturbations.

## 7 Concluding Remarks

We presented DT-STARK, that extends the STARK tool with a feedback mechanism that allows us to encode the communication between the digital and the physical twin. Thanks to it, we can use RobTL formulae to evaluate the effectiveness of the DT, namely how much it is able to direct the PT that is operating under the effect of uncertainty and perturbations. To the best of our knowledge, DT-STARK is the only formal tool that allows for this kind of analysis.

Given the high level of abstraction in the current implementation, the next step in the development of DT-STARK will consist in switching from discrete step modelling to a time-point modelling, by refining the techniques we introduced in *Bio-STARK* [15] for the robustness analysis of biological systems. This will allow us not only to check for discrepancies in time in the behaviour of the two twins, but also to account for potential delays in the communication between them, and thus in the application of feedback, and their consequences. Moreover, for a more accurate simulation of behaviour, we plan to extend (DT-)STARK with ODE modelling.

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

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

## References

1. Ábrahám, E., Bonakdarpour, B.: HyperPCTL: a temporal logic for probabilistic hyperproperties. In: McIver, A., Horvath, A. (eds.) QEST 2018. LNCS, vol. 11024, pp. 20–35. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-99154-2\\_2](https://doi.org/10.1007/978-3-319-99154-2_2)
2. Agha, G., Palmkog, K.: A survey of statistical model checking. *ACM Trans. Model. Comput. Simul.* **28**(1), 6:1–6:39 (2018). <https://doi.org/10.1145/3158668>
3. Althoff, M., Dolan, J.M.: Reachability computation of low-order models for the safety verification of high-order road vehicle models. In: American Control Conference (ACC) 2012, pp. 3559–3566. IEEE (2012). <https://doi.org/10.1109/ACC.2012.6314777>
4. Arora, S., Hansen, R.R., Larsen, K.G., Legay, A., Poulsen, D.B.: Statistical model checking for probabilistic hyperproperties of real-valued signals. In: Legunsen, O., Rosu, G. (eds.) SPIN 2022. LNCS, vol. 13255, pp. 61–78. Springer, Cham (2022). [https://doi.org/10.1007/978-3-031-15077-7\\_4](https://doi.org/10.1007/978-3-031-15077-7_4)
5. Baier, C.: Probabilistic model checking. In: Dependable Software Systems Engineering, NATO Science for Peace and Security Series - D: Information and Communication Security, vol. 45, pp. 1–23. IOS Press (2016). <https://doi.org/10.3233/978-1-61499-627-9-1>
6. Banks, J.: Handbook of Simulation - Principles, Methodology, Advances, Applications, and Practice. Wiley, Hoboken (1998)
7. Behrmann, G., David, A., Larsen, K.G.: A tutorial on UPPAAL. In: Bernardo, M., Corradini, F. (eds.) SFM-RT 2004. LNCS, vol. 3185, pp. 200–236. Springer, Cham (2004). [https://doi.org/10.1007/978-3-540-30080-9\\_7](https://doi.org/10.1007/978-3-540-30080-9_7)
8. Bogachev, V.I.: Measure Theory, vol. 2. Springer, Heidelberg (2007). <https://doi.org/10.1007/978-3-540-34514-5>

9. Brosinsky, C., Westermann, D., Krebs, R.: Recent and prospective developments in power system control centers: adapting the digital twin technology for application in power system control centers. In: 2018 IEEE International Energy Conference (ENERGYCON), pp. 1–6 (2018). <https://doi.org/10.1109/ENERGYCON.2018.8398846>
10. Cassandras, C.G., Lygeros, J. (eds.): Stochastic Hybrid Systems. No. 24 in Control Engineering, 1st edn. CRC Press, Boca Raton (2007). <https://doi.org/10.1201/9781315221625>
11. Castiglioni, V., Loreti, M., Tini, S.: How adaptive and reliable is your program? In: Peters, K., Willemse, T.A.C. (eds.) FORTE 2021. LNCS, vol. 12719, pp. 60–79. Springer, Cham (2021). [https://doi.org/10.1007/978-3-030-78089-0\\_4](https://doi.org/10.1007/978-3-030-78089-0_4)
12. Castiglioni, V., Loreti, M., Tini, S.: RobTL: a temporal logic for the robustness of cyber-physical systems. CoRR abs/2212.11158 (2022). <https://doi.org/10.48550/arXiv.2212.11158>
13. Castiglioni, V., Loreti, M., Tini, S.: A framework to measure the robustness of programs in the unpredictable environment. Log. Methods Comput. Sci. **19**(3) (2023). [https://doi.org/10.46298/LMCS-19\(3:2\)2023](https://doi.org/10.46298/LMCS-19(3:2)2023)
14. Castiglioni, V., Loreti, M., Tini, S.: STARK: a software tool for the analysis of robustness in the unknown environment. In: Jongmans, S.S., Lopes, A. (eds.) COORDINATION 2023. LNCS, vol. 13908, pp. 115–132. Springer, Cham (2023). [https://doi.org/10.1007/978-3-031-35361-1\\_6](https://doi.org/10.1007/978-3-031-35361-1_6)
15. Castiglioni, V., Loreti, M., Tini, S.: Bio-STARK: a tool for the time-point robustness analysis of biological systems. In: Proceedings of CMSB 2024. LNCS. Springer (2024, to appear)
16. Castiglioni, V., Loreti, M., Tini, S.: RobTL: robustness temporal logic for CPS. In: Proceedings of CONCUR 2024. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2024, to appear)
17. Castiglioni, V., Loreti, M., Tini, S.: STARK: a tool for the analysis of CPSs robustness. Sci. Comput. Program. **236**, 103134 (2024). <https://doi.org/10.1016/j.scico.2024.103134>
18. Clarkson, M.R., Schneider, F.B.: Hyperproperties. J. Comput. Secur. **18**(6), 1157–1210 (2010). <https://doi.org/10.3233/JCS-2009-0393>
19. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: Chatterjee, K., Henzinger, T.A. (eds.) FORMATS 2010. LNCS, vol. 6246, pp. 92–106. Springer, Heidelberg (2010). [https://doi.org/10.1007/978-3-642-15297-9\\_9](https://doi.org/10.1007/978-3-642-15297-9_9)
20. Esterle, L., Gomes, C., Frasheri, M., Ejersbo, H., Tomforde, S., Larsen, P.G.: Digital twins for collaboration and self-integration. In: 2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C), pp. 172–177 (2021). <https://doi.org/10.1109/ACSOS-C52956.2021.00040>
21. Esterle, L., Porter, B., Woodcock, J.: Verification and uncertainties in self-integrating system. In: 2nd IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion, ACSOS-C 2021, pp. 220–225. IEEE (2021). <https://doi.org/10.1109/ACSOS-C52956.2021.00040>
22. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications for continuous-time signals. Theor. Comput. Sci. **410**(42), 4262–4291 (2009). <https://doi.org/10.1016/j.tcs.2009.06.021>
23. Frasheri, M., et al.: Addressing time discrepancy between digital and physical twins. Robot. Auton. Syst. **161**, 104347 (2023). <https://doi.org/10.1016/j.robot.2022.104347>
24. Fuller, A., Fan, Z., Day, C., Barlow, C.: Digital twin: enabling technologies, challenges and open research. IEEE Access **8**, 108952–108971 (2020). <https://doi.org/10.1109/ACCESS.2020.2998358>
25. Gahlot, S., Reddy, S.R.N., Kumar, D.: Review of smart health monitoring approaches with survey analysis and proposed framework. IEEE Internet Things J. **6**, 2116–2127 (2019). <https://doi.org/10.1109/JIOT.2018.2872389>

26. Grieves, M., Vickers, J.: Digital twin: mitigating unpredictable, undesirable emergent behavior in complex systems. In: Kahlen, J., Flumerfelt, S., Alves, A. (eds.) *Transdisciplinary Perspectives on Complex Systems*, pp. 85–113. Springer, Cham (2017). [https://doi.org/10.1007/978-3-319-38756-7\\_4](https://doi.org/10.1007/978-3-319-38756-7_4)
27. Hensel, C., Junges, S., Katoen, J.P., Quatmann, T., Volk, M.: The probabilistic model checker STORM. *Int. J. Softw. Tools Technol. Transfer* **24**(4), 589–610 (2022). <https://doi.org/10.1007/s10009-021-00633-z>
28. Henzinger, T.A., Ho, P.H., Wong-Toi, H.: HYTECH: a model checker for hybrid systems. In: Grumberg, O. (ed.) *CAV 1997*. LNCS, vol. 1254, pp. 460–463. Springer, Cham (1997). [https://doi.org/10.1007/3-540-63166-6\\_48](https://doi.org/10.1007/3-540-63166-6_48)
29. Hinton, A., Kwiatkowska, M., Norman, G., Parker, D.: PRISM: a tool for automatic verification of probabilistic systems. In: Hermanns, H., Palsberg, J. (eds.) *TACAS 2006*. LNCS, vol. 3920, pp. 441–444. Springer, Heidelberg (2006). [https://doi.org/10.1007/11691372\\_29](https://doi.org/10.1007/11691372_29)
30. Hu, J., Lygeros, J., Sastry, S.: Towards a theory of stochastic hybrid systems. In: Lynch, N., Krogh, B.H. (eds.) *HSCC 2000*. LNCS, vol. 1790, pp. 160–173. Springer, Heidelberg (2000). [https://doi.org/10.1007/3-540-46430-1\\_16](https://doi.org/10.1007/3-540-46430-1_16)
31. Ibrahim, M., Rassölkin, A., Vaimann, T., Kallaste, A.: Overview on digital twin for autonomous electrical vehicles propulsion drive system. *Sustainability* **14**(2) (2022). <https://doi.org/10.3390/su14020601>
32. Kempers, S.T., et al.: Tech united Eindhoven middle size league winner 2022. In: Eguchi, A., Lau, N., Paetzel-Prüsmann, M., Wanichanon, T. (eds.) *RoboCup 2022*. LNCS, vol. 13561, pp. 337–348. Springer, Cham (2023). [https://doi.org/10.1007/978-3-031-28469-4\\_28](https://doi.org/10.1007/978-3-031-28469-4_28)
33. Kwiatkowska, M.Z., Norman, G., Parker, D.: Stochastic model checking. In: Bernardo, M., Hillston, J. (eds.) *SFM 2007*. LNCS, vol. 4486, pp. 220–270. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-72522-0\\_6](https://doi.org/10.1007/978-3-540-72522-0_6)
34. Lanotte, R., Merro, M., Zannone, N.: Impact analysis of coordinated cyber-physical attacks via statistical model checking: a case study. In: Huisman, M., Ravara, A. (eds.) *FORTE 2023*. LNCS, vol. 13910, pp. 75–94. Springer, Cham (2023). [https://doi.org/10.1007/978-3-031-35355-0\\_6](https://doi.org/10.1007/978-3-031-35355-0_6)
35. Legay, A., Lukina, A., Traonouez, L., Yang, J., Smolka, S.A., Grosu, R.: Statistical model checking. In: Steffen, B., Woeginger, G. (eds.) *Computing and Software Science - State of the Art and Perspectives*. LNCS, vol. 10000, pp. 478–504. Springer, Cham (2019). [https://doi.org/10.1007/978-3-319-91908-9\\_23](https://doi.org/10.1007/978-3-319-91908-9_23)
36. Li, S., Yang, Q., Xing, J., Chen, W., Zou, R.: A foundation model for building digital twins: a case study of a chiller. *Buildings* **12**(8) (2022). <https://doi.org/10.3390/buildings12081079>
37. Lugaresi, G., Gangemi, S., Gazzoni, G., Matta, A.: Online validation of digital twins for manufacturing systems. *Comput. Ind.* **150**, 103942 (2023). <https://doi.org/10.1016/j.compind.2023.103942>
38. Martinez, C.L., et al.: Tech united Eindhoven team description. Technical report, Eindhoven University of Technology (2014). <https://www.techunited.nl/media/files/TDP2014.pdf>
39. Naderifar, V., Sahran, S., Shukur, Z.: A review on conformance checking technique for the evaluation of process mining algorithms. *TEM J.* **8**(4), 1232 (2019). <https://doi.org/10.18421/TEM84-18>
40. Pincirolì, R., Trubiani, C.: Model-based performance analysis for architecting cyber-physical dynamic spaces. In: *Proceedings of ICSA 2021*, pp. 104–114. IEEE (2021). <https://doi.org/10.1109/ICSA51549.2021.00018>
41. Qi, Q., Tao, F.: Digital twin and big data towards smart manufacturing and industry 4.0: 360 degree comparison. *IEEE Access* **6**, 3585–3593 (2018). <https://doi.org/10.1109/access.2018.2793265>

42. Riedmaier, S., Danquah, B., Schick, B., Diermeyer, F.: Unified framework and survey for model verification, validation and uncertainty quantification. *Arch. Comput. Methods Eng.* **28**, 2655–2688 (2021). <https://doi.org/10.1007/s11831-020-09473-7>
43. Sankararaman, S., Mahadevan, S.: Integration of model verification, validation, and calibration for uncertainty quantification in engineering systems. *Reliab. Eng. Syst. Saf.* **138**, 194–209 (2015). <https://doi.org/10.1016/j.ress.2015.01.023>
44. Sargent, R.G.: Verification and validation of simulation models. In: *Proceedings of the 2010 Winter Simulation Conference*, pp. 166–183 (2010). <https://doi.org/10.1109/WSC.2010.5679166>
45. Sargent, R.G.: Verification and validation of simulation models. *J. Simul.* **7**, 12–24 (2013). <https://doi.org/10.1057/JOS.2012.20>
46. Sen, K., Viswanathan, M., Agha, G.: On statistical model checking of stochastic systems. In: Etesami, K., Rajamani, S.K. (eds.) *CAV 2005*. LNCS, vol. 3576, pp. 266–280. Springer, Heidelberg (2005). [https://doi.org/10.1007/11513988\\_26](https://doi.org/10.1007/11513988_26)
47. Shaikh, E., Al-Ali, A., Muhammad, S., Mohammad, N., Aloul, F.A.: Security analysis of a digital twin framework using probabilistic model checking. *IEEE Access* **11**, 26358–26374 (2023). <https://doi.org/10.1109/ACCESS.2023.3257171>
48. Sharma, A., Kosasih, E., Zhang, J., Brintrup, A., Calinescu, A.: Digital twins: state of the art theory and practice, challenges, and open research questions. *J. Ind. Inf. Integr.* **30**, 100383 (2022). <https://doi.org/10.1016/j.jii.2022.100383>
49. Sifakis, J., Harel, D.: Trustworthy autonomous system development. *ACM Trans. Embed. Comput. Syst.* **22**(3) (2023). <https://doi.org/10.1145/3545178>
50. Soe, R.M.: FINEST twins: platform for cross-border smart city solutions. In: *Proceedings of the 18th Annual International Conference on Digital Government Research*, pp. 352–357. Association for Computing Machinery (2017). <https://doi.org/10.1145/3085228.3085287>
51. Umeda, Y., et al.: Development of an education program for digital manufacturing system engineers based on ‘digital triplet’ concept. *Procedia Manuf.* **31**, 363–369 (2019). <https://doi.org/10.1016/j.promfg.2019.03.057>
52. Vaserstein, L.N.: Markovian processes on countable space product describing large systems of automata. *Probl. Peredachi Inf.* **5**(3), 64–72 (1969)