Formally Verified Self-adaptation of an Incubator Digital Twin

Thomas Wright[1(✉)], Cláudio Gomes[2], and Jim Woodcock[1,2]

[1] Department of Computer Science, University of York, Heslington, UK
{thomas.d.wright,jim}@york.ac.uk
[2] DIGIT, Department of Engineering, Aarhus University, Aarhus, Denmark
claudio.gomes@ece.au.dk

Abstract. The performance and reliability of Cyber-Physical Systems are increasingly aided through the use of digital twins, which mirror the static and dynamic behaviour of a Cyber-Physical System (CPS) in software. Digital twins enable the development of self-adaptive CPSs which reconfigure their behaviour in response to novel environments. It is crucial that these self-adaptations are formally verified at runtime, to avoid expensive re-certification of the reconfigured CPS. In this paper, we demonstrate formally verified self-adaptation in a digital twinning system, by constructing a non-deterministic model which captures the uncertainties in the system behaviour after a self-adaptation. We use Signal Temporal Logic to specify the safety requirements the system must satisfy after reconfiguration and employ formal methods based on verified monitoring over Flow* flowpipes to check these properties at runtime. This gives us a framework to predictively detect and mitigate unsafe self-adaptations before they can lead to unsafe states in the physical system.

Keywords: Digital twin · Self-adaptation · Reachability analysis · Signal temporal logic · Optimization · Cyber-physical system

1 Introduction

A Cyber-Physical System (CPS) consists of a digital component controlling a physical asset within some operating environment. Cyber-Physical Systems design poses significant engineering challenges, whilst scalably verifying that CPSs meet their requirements has long been a central problem in formal methods research [2,5,29,51]. Moreover, a CPS must cope with significant uncertainty and change during its operations. This motivates the need for *self-adaptive* cyber-physical systems which dynamically reconfigure their behaviour in response to anomalous situations. However, the dynamic nature of these reconfigurations induces significant additional design and verification challenges, demanding new methods for engineering safe self-adaptive CPSs.

One approach to the challenges of CPS engineering comes through the use of digital twins. A *digital twin* is a computational replica of a CPS, which we refer to

T. Margaria and B. Steffen (Eds.): ISoLA 2022, LNCS 13704, pp. 89–109, 2022.
https://doi.org/10.1007/978-3-031-19762-8_7

as the *physical twin*. The digital twin is constructed from heterogeneous models of the physical system, including its hardware components, control software, and physical environment. The digital twin synchronises with the physical twin by monitoring its behaviour in order to update the state of these models.

Whilst a digital twin for a simple CPS can be implemented directly based on sensory data, it is difficult to get a comprehensive view of the state of a more complex CPS. This requires a combination of state estimators, data-fusion algorithms, and numerical simulation, which may still leave discrepancies between the digital and physical twins. Such discrepancies can also arise due to unexpected shifts in the CPS's operating environment, causing the model parameters of the digital twin to become out of date. Hence, we must continually monitor the conformance of the physical twin behaviour to the models in the digital twin to detect these anomalies and recalibrate the digital twin parameters based on the data from the physical twin. The digital twin may also be used to alter the behaviour of the CPS when conformance is violated (see Kritzinger et al. [33] and Tao et al. [54]).

Because the digital twin changes the behaviour of the physical twin, it is crucial that these changes are formally verified to be safe. This challenge has been well discussed in [57] and [28] where the application of formal methods is surveyed in the context of self-adaptive systems. This is relevant since a digital twin enables self-adaptation of its physical twin. However, digital twins place a higher emphasis on physical systems, which means that traditional formal methods must be adapted. Indeed, in practical systems, self-adaptation cannot be deployed, since each system reconfiguration requires re-certification of the equipment, leading to long potential downtimes. Nevertheless, it is our vision that re-certification can be sped up with the application of formal methods.

We will explore many of these challenges through a model incubator system [25] in which a digital controller regulates the temperature inside an incubator box by controlling a heat-bed inside the box. A digital twin of the incubator can measure the temperature within the box through digital temperature sensors placed at different locations within the box; these temperature readings can be used to calibrate the parameters of the digital twin models. If effectively calibrated, these models can be used to predict the future values of the box temperature or to synthesise optimal control policies for the heat-bed. However, we must handle discrepancies in these predictions arising from a number of uncertainties inherent in the calibration process: (i) temperature sensors at different locations in the box may give inconsistent readings; (ii) the sensor data represents delayed discrete samples of the system; (iii) sensors have noisy readings and actuators are inaccurate; (iv) the digital twin models only approximate the physical twin; (v) there are processing delays in the digital twin; (vi) the incubator's operating environment is uncertain and changeable.

Contribution. In this paper, we demonstrate formally verified self-adaptation in the context of an incubator digital twin system. To this end, we construct a second non-deterministic model that predicts the behaviour of the physical twin after a self-adaptation whilst we perform *uncertainty calibration* to measure and account

for the uncertainties introduced during the self-adaptation process. This enables us to apply exact formal verification, leveraging Flow* verified integration [13] to perform verified monitoring [59] of the non-deterministic model against high-level safety requirements specified in the Signal Temporal Logic (STL) [39]. Verification is performed inside the self-adaptive loop to predict future violations after each self-adaptation. This is in contrast with most (offline or online) STL monitoring approaches that use data from the physical system to detect violations which have already occurred. Thus we may perform online monitoring of self-adaptations, which predictively identifies unsafe self-adaptations, or active enforcement, enabling the system to take evasive action to avert unsafety.

Related Work. Woodcock et al. [58] demonstrated how safety violations of CPSs in uncertain environments may be detected based on statistical analysis of digital twin cosimulations. Formally verified self-adaptation can be seen as an alternative approach to handling environmental uncertainty, with the non-deterministic model assuring that safety is maintained.

A variety of works have considered formal verification of self-adaptive software systems. Of these, our approach is particularly related to [8,9,21] which develop predictive monitoring of non-functional requirements expressed in the QCTL [4] temporal logic. It should also be compared to SimCA* [52] which is able to give formal guarantees for control-theoretic requirements under environmental uncertainties. On the other hand, most of the work on self-adaptive CPSs has focused on self-adaptation at the architectural or software levels [41]. These also include applications of concurrency-theoretic formalisms [7,55] to verify self-adaptations which reconfigure the network topology of a CPS. However, none of this work has considered formal verification of the controlled continuous dynamics of the system, which is the main focus of this paper. These challenges are related to Fault Detection, Isolation, and Reconfiguration (FDIR) problems [30], which have been considered in the control theory community, although the focus of these works is quite different than our temporal-logic based approach.

Another distinctive feature of our approach is the application of predictive STL model checking at runtime. Outside of the context of self-adaptive systems, this is related to the Clairvoyant monitoring approach of Qin and Deshmukh [44] which fits statistical models to traces in order to predict the probability that a STL property will be satisfied by future extension of the trace and to the approach of Ma et al. [37] which makes predictions based on Bayesian Recurrent Neural Networks with calibrated uncertainty estimation. In contrast, our approach expands a system's digital twin into an uncertain dynamical system model, which is used to predict its future behaviour. This is worth comparing to the model-bounded monitoring approaches of Waga, André, and Hasuo [56] and of Ghosh and André [27] which both use uncertain linear dynamical systems to interpolate between sparsely sampled time series data. Also relevant is the closely related problem of model predictive synthesis of controllers satisfying STL specifications [20,22,43,45,46,48–50] including the recent reachability-based methods [11,16,53].

A number of works [1,15,17,35,60] have also applied reachability analysis to predict future safety violations at runtime. Zhang et al. [62] have also

demonstrated online repair based on control synthesis. However, the only one of these methods which moves beyond reach-avoidance properties to a full range of STL properties is that of Yu et al. [61], which targets discrete rather than continuous time dynamical systems.

2 Background

In this section we introduce some background material on the incubator system and on our verified monitoring approach for STL specifications.

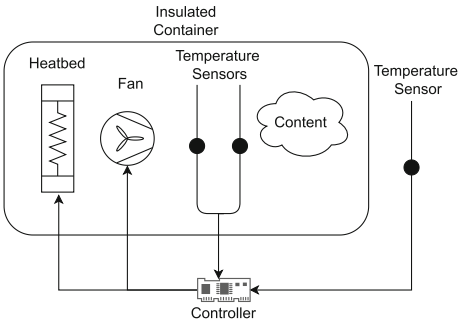
2.1 Notation

Firstly, we introduce some mathematical notation which we will use throughout the paper. We will frequently work with the real numbers $\mathbb{R}$, including the space of non-negative real numbers $\mathbb{R}_{\geq 0} = [0, \infty)$ and the space of n-dimensional real vectors $\mathbb{R}^n$. We use boldface to distinguish the names of vectors $\mathbf{x}$ from scalars x, and write a specific n-dimensional vector with real entries $x_1, \ldots, x_n \in \mathbb{R}$ as $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$. We rely upon interval arithmetic [40], which represents uncertain quantities as closed real intervals $I = [a, b] \in \mathbb{IR}$ and defines over-approximate arithmetic operations based on the endpoints of intervals so that, for example, $[a, b] + [c, d] = [a + c, b + d]$. We also work with interval vectors $\mathbf{I} = (I_1, \ldots, I_n) \in \mathbb{IR}^n$ which consist of interval entries $I_1, \ldots, I_n \in \mathbb{IR}$ and support all of the standard vector operations. We define the interval vector $[\mathbf{x}, \mathbf{y}] = ([x_1, y_1], \ldots, [x_n, y_n])$ ranging between two real vectors $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$ (assuming $x_i \leq y_i$ for all i). Finally, we define the width of an interval $[a, b]$, $\text{width}([a, b]) = b - a$, and the distance of a point x from an interval $[a, b]$,

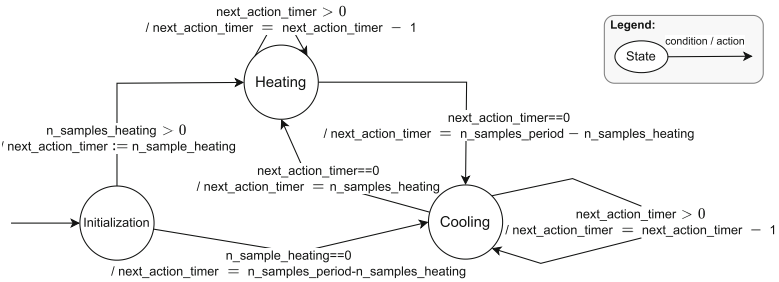
$$\text{dist}(x, [a, b]) \triangleq \begin{cases} x - b & \text{if } x > b \\ a - x & \text{if } x < a \\ 0 & \text{otherwise} \end{cases}.$$

2.2 Incubator

The incubator system, detailed in [25], consists of a styrofoam box and a digital controller. An overview of the incubator and its control logic is shown in Fig. 1. It consists of a heat-bed (that radiates heat when turned on) and a fan (that ensures uniform temperature distribution inside the box). The temperature can be sensed and sent to the controller from two different spots inside the box, and a spot outside the box. The duty cycle of the controller regulates the steady state temperature inside the box. In this paper we will consider an open-loop controller (shown in Fig. 1b) which operates independently of the temperature measurements, but is periodically reconfigured based on the temperature measurements and a digital twin of the incubator system.



(a) Schematic overview of the incubator.



(b) Statechart of the open-loop controller.

Fig. 1. Overview of the incubator system.

As shown in [25, Section 3], we can model the incubator using the following system of equations which describe the evolution of the temperature of the air inside the box T_A and of the heat-bed T_H, in degrees Celsius:

$$\begin{aligned} \frac{\mathrm{d}T_H}{\mathrm{d}t} &= \frac{1}{C_H}\big(VI - G_H\big(T_H - T_A\big)\big) \\ \frac{\mathrm{d}T_A}{\mathrm{d}t} &= \frac{1}{C_A}\big(G_H\big(T_H - T_A\big) - G_B\big(T_A - T_R\big)\big) \end{aligned} \tag{1}$$

where:

- V and I denote voltage and current, respectively, and the product VI represents power (rate of energy produced at the heat-bed);
- G_H represents the rate of energy transfer between the surface of the heat-bed and the surrounding air;
- C_H encapsulates both the heat capacity of the heat-bed as well as its mass;
- C_A encapsulates the heat capacity and mass of the air inside the box;
- G_B represents the rate of energy transfer between the air inside the box and the air outside the box (e.g. the lid being opened is equivalent to increasing this value by an order of magnitude).

Equation (1) represents the plant without any control action. We also need to include the control signal which turns the heat-bed on and off. Therefore the controlled equations have the form

$$\begin{aligned}\frac{\mathrm{d}T_H}{\mathrm{d}t} &= \frac{1}{C_H}\big(c(t)VI - G_H\big(T_H - T_A\big)\big)\\ \frac{\mathrm{d}T_A}{\mathrm{d}t} &= \frac{1}{C_A}\big(G_H\big(T_H - T_A\big) - G_B\big(T_A - T_R\big)\big)\end{aligned} \tag{2}$$

where the input signal $c : \mathbb{R}_{\geq 0} \to \{0, 1\}$ determines the control state of the heater at a given instant in time. In particular, in the incubator system this control signal takes the form of a piecewise constant periodic signal $c = c_{k,l}$ which (after an initialisation period) alternates between heating for k duty cycles ($c_{k,l}(t) = 1$) and cooling for l duty cycles ($c_{k,l}(t) = 0$) in line with Fig. 1b.

2.3 Flow* Verified Integration

The majority of simulation and analysis of mathematical models such as digital twins is carried out using numerical methods. Whilst the flexibility and performance of these methods makes them invaluable, a major limitation is their approximate nature, which means they are unable to definitively prove properties of the system as their results can be unreliable for sensitive or chaotic systems [10]. An even larger practical limitation is their inability to represent and account for uncertainties in the system, preventing us from providing verification results which are robust to noise or mismatches between a digital twin and a physical system.

Verified integration applies exact formal methods in order to move beyond the approximate simulation results produced by classical numerical methods, to computing a verified enclosure of all possible trajectories of a system over time. One leading such method is the Flow* verified integrator [14] which applies a variety of Taylor model-based [6] methods to tightly enclose the dynamics of continuous and hybrid dynamical systems featuring complex non-linear dynamics and large uncertainties in initial conditions and model parameters.

In particular, Flow* verified integration is able to handle uncertain parametric continuous systems of form,

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, \mathbf{p}, \mathbf{c}(t), t) \tag{3}$$

whose dynamics are specified by a Lipschitz continuous function $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^q \times \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ (i.e. a vector of n coupled non-linear ODEs) subject to a vector $\mathbf{p}$ of system parameters and a predetermined open-loop control policy $\mathbf{c} : \mathbb{R}_{\geq 0} \to \mathbb{R}^m$ which we assume to be a piecewise constant. We are able to introduce uncertainties in this class of models both through an interval initial value constraint $\mathbf{x}(0) \in \mathbf{I}$ which states that the system must start inside the n-dimensional box $\mathbf{I} \in \mathbb{IR}^n$ of initial conditions and the interval parameter constraint $\mathbf{p} \in \mathbf{U}$ which constrains the parameters of the system to the

m-dimensional box $\mathbf{U} \in \mathbb{IR}^m$. These uncertain parameters are assumed to be *time-invariant* (so that they have fixed real values; we just do not know what they are); whilst such uncertain parameters have been tackled explicitly by verified-integration methods such that of Lin and Stadtherr [36], we handle them by implicitly re-encoding them as uncertain initial conditions for additional derivative-zero variables of the model[1]. We denote an uncertain parametric system of the above form as $\mathcal{M}(\mathbf{I}, \mathbf{U}, \mathbf{c})$ where $\mathbf{I}$ and $\mathbf{U}$ record the interval initial condition and parameter constraints of the system and $\mathbf{c}$ records the control policy.

In order to enclose all possible behaviours of the system under uncertainty, Flow* moves from approximating a single trajectory $\mathbf{x} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ of the system, to computing a *flowpipe* enclosing all possible trajectories of the system. Whilst internally Flow* uses a complex symbolic flowpipe representation based on preconditioned Taylor models [38] to give the tightest possible bounds on system dynamics, we can view these flowpipes as interval vector functions $\mathbf{g} : \mathbb{IR} \to \mathbb{IR}^n$ which map interval regions $T \in \mathbb{IR}$ of the time domain to n-dimensional regions $\mathbf{g}(T) \in \mathbb{IR}^n$ of the system state space. These flowpipes are then guaranteed to form an *interval extension* of every possible trajectory of the underlying system in the sense that, for every trajectory $\mathbf{x} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$, every time point $t \in \mathbb{R}_{\geq 0}$, and every time interval $T \in \mathbb{IR}$ such that $t \in T$ we are guaranteed that $\mathbf{x}(t) \in \mathbf{g}(T)$. This means a Flow* flowpipe computed for a given uncertain model, which we henceforth denote flowpipe$(\mathcal{M}(\mathbf{I}, \mathbf{U}, \mathbf{c}))$, is guaranteed to soundly enclose all possible behaviours of a system regardless of any uncertainties in the initial conditions $\mathbf{I}$ or parameters $\mathbf{U}$ of the system.

Example 1. We can view the incubator model Eq. (1) as an uncertain model of form Eq. (3) if we assume uncertain knowledge of the initial state of the system, given by the interval initial value constraints,

$$T_A(0) \in I_A = \big[25.0, 25.1\big], \qquad T_H(0) \in I_H = \big[20.59, 21.60\big],$$

uncertain knowledge of the system parameters C_A, G_B given by the interval constraints,

$$C_A \in U_A = \big[68.20, 68.71\big], \qquad C_H \in U_H = \big[0.73, 0.79\big],$$

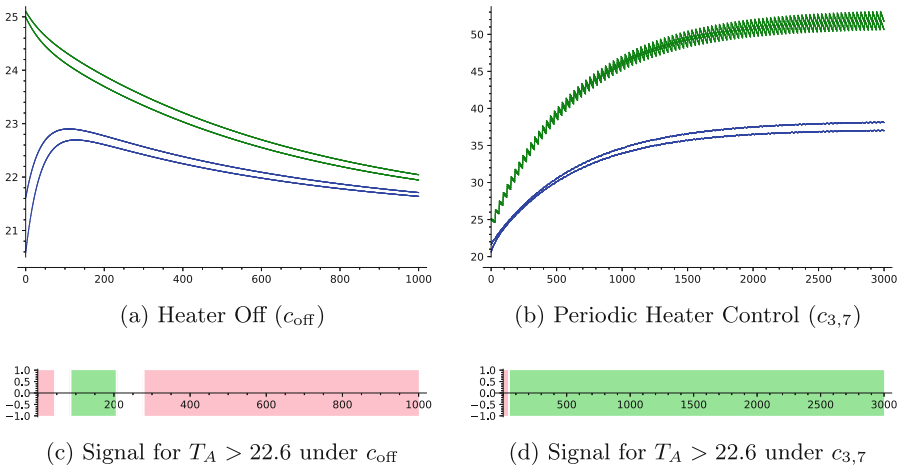
and the following fixed values of the remaining system parameters,

$$V = 12.00, \qquad I = 10.45, \qquad T_R = 21.25, \qquad G_H = 0.87095429.$$

Following the notation of the previous section, we denote the overall uncertain incubator model as $\mathcal{M}\big((I_A, I_H), (U_A, U_H), c\big)$.

If we apply a constant control policy $c_{\text{off}}(t) \equiv 0$ in which the heater is always off, then applying Flow* gives the flowpipes shown in Fig. 2a. This demonstrates

[1] Flow* also has native support for time-varying interval uncertain parameters [12, Section 3.5], which may vary throughout the simulation leading to much greater uncertainty in the overall behaviour of the system over time.



(a) Heater Off (c_{off})

(b) Periodic Heater Control ($c_{3,7}$)

(c) Signal for $T_A > 22.6$ under c_{off}

(d) Signal for $T_A > 22.6$ under $c_{3,7}$

Fig. 2. Verified integration and monitoring results for the incubator under uncertainty.

that the system absorbs the uncertainties introduced by the uncertain initial conditions and hence the long-term behaviour of the system is robust under these variations, as demonstrated in [26].

We can also apply a periodic open-loop control policy $c_{3,7}(t)$ in which the heater alternates between switching on for periods of 3 duty cycles and cooling down for periods of 7 duty cycles. This results in the flowpipes shown in Fig. 2, which demonstrates that this control policy results in an initial rise of the box and heater temperature before eventual stabilisation.

2.4 Verified Monitoring

Whilst models and simulation techniques provide a powerful way to analyse the behaviour of systems, their application typically depends on human insight to interpret the results and determine whether it is consistent with the expected safe behaviour of the system. Since self-adaptive systems are designed to operate without human intervention, we propose the use of specification languages such as temporal logics to capture the safety requirements of the system, and offline or online monitoring techniques to check whether a given system behaviour is consistent with these requirements.

Signal Temporal Logic (STL) [39] has emerged as a popular specification language for the behaviour of Cyber-Physical Systems [5]. STL formulae are defined according to the following grammar,

$$\phi, \psi ::= \rho \mid \phi \wedge \psi \mid \phi \vee \psi \mid \neg\phi \mid \mathcal{F}_{[a,b]}\,\phi \mid \mathcal{G}_{[a,b]}\,\phi \mid \phi\,\mathcal{U}_{[a,b]}\,\psi\,,$$

and incorporate as atomic propositions inequalities $\rho \triangleq f(\mathbf{x}) \geq 0$ featuring functions $f(\mathbf{x})$ of the system variables alongside complex propositions with the

logical operators *not* $\neg\phi$, *and* $\phi \wedge \psi$, and *or* $\phi \vee \psi$, as well as the temporal operators $\mathcal{F}_{[a,b]}\,\phi$ or *eventually* ϕ (which states that the STL property ϕ should hold at *some* time point between a and b time units in the future), $\mathcal{G}_{[a,b]}\,\phi$ or *globally* ϕ (which states that the STL property ϕ should hold at *all* time points between a and b time units in the future); as well as the *until operator* $\phi\,\mathcal{U}_{[a,b]}\,\psi$ (which states that there is some time point t between a and b time units in the future such that ψ is true at t and that ϕ is true at every time point t' between now and then).

STL monitoring has traditionally been applied to a single *numerical signal* $\mathbf{x} : \mathbb{K} \to \mathbb{R}^n$ (with either bounded real-time domain $\mathbb{K} = [0, T]$ or discrete time domain $\mathbb{K} = \{t_1, \ldots, t_N\} \subseteq [0, T]$ based on time-sampling points $t_1 < t_2 < \ldots < t_N$) either from a running system or a numerical simulation of its behaviour using bottom-up monitoring algorithms [39] that recursively compute *Boolean signals* $s : [0, T] \to \{\text{True}, \text{False}\}$ which record the truth of a STL given property at each time point $t \in [0, T]$. In contrast, verified methods such as Flow* have traditionally focused on *reachability analysis* to enclose all possible behaviours of the system without supporting the rich timed specifications which STL allows. However, recently a number of methods [11,31,34,47,59] have emerged applying verified reachability analysis as a basis for *verified monitoring* of STL properties. In particular we apply the method of Wright and Stark [59] which implements verified monitoring of STL properties over Flow* flowpipes by computing three-valued signals $s : \mathbb{R}_{\geq 0} \to \{\text{True}, \text{Unknown}, \text{False}\}$ which uses a third truth value, Unknown, to record when the uncertainty in the flowpipe is too great to either verify or refute a property at a given time point $t \in \mathbb{R}_{\geq 0}$. Thus, the signal produced by verified monitoring of a STL property over a flowpipe $\mathcal{M}(\mathbf{I}, \mathbf{U}, \mathbf{c})$ for a given uncertain model gives formal guarantees, with True and False values guaranteeing the truth or falsehood of the property for all possible initial conditions $\mathbf{x} \in \mathbf{I}$ and parameters $\mathbf{p} \in \mathbf{U}$ of the model. We are therefore able to apply verified STL monitoring in order to capture many interesting properties of the incubator system.

Example 2. As a simple example, we can monitor properties of system state variables such as $T_A \geq 22.6$ which asserts that the air temperature within the incubator is at least 22.6 °C. A three-valued signal for this property when the heater is off is shown in Fig. 2c illustrating that the property is true at both ends of the time period of truth at the peak of the temperature graph and periods of uncertainty in between. Similarly, a three-valued signal for the same property with a periodic heater control policy $c_{3,7}$ is shown in Fig. 2d.

We are also able to apply the full monitoring algorithm to verify properties of the overall timed behaviour of the incubator. For example, under the periodic control policy $c_{3,7}$ we are able to verify the STL property

$$\mathcal{F}_{[0,2000]}\,\mathcal{G}_{[0,100]}(T_A \geq 33 \wedge T_A \leq 36)$$

which states that the air temperature eventually (within 2000 seconds) reaches and (for at least 100 seconds) remains within the interval range [33, 36]. This corresponds to the control-theoretic requirement that the air temperature stabilises close to 34.5 °C, however, the use of STL allows us to strengthen this to require that the temperature stabilises within a timely manner (within 2000 s).

3 Formally Verified Self-Adaptation

This section lays out our approach for introducing formally verified self-adaptation into an incubator digital twin system, through the use of a self-adaptation loop, and formal verification of a non-deterministic model capturing the behaviour of the incubator after each self-adaptation.

3.1 Incubator Self-Adaptation Loop

The digital twin of the incubator, originally introduced in [24], has been extended with a self-adaptation loop in [23], that reconfigures the duty cycle of the open loop controller whenever an external disturbance undermines the predictive power of the model in Eq. (1). The most common example disturbance is when the lid of the incubator is opened.

A disturbance is detected by comparing the output from a Kalman filter (that uses Eq. (1)) with the actual temperature measurements. After some time, the disturbance is confirmed, and the following steps, based on the MAPE-K loop [32], are taken:

Gather Data —The system is left to operate normally for some more time steps, in order to gather sufficient data for the next step.

Recalibrate Model —The data gathered since the time the anomaly was detected is used to re-estimate a real-valued vector $\mathbf{p}^* = (C_A^*, G_H^*)$ of parameters for the incubator model in Eq. (1) by repeatedly running simulations and comparing them to the data, while adjusting the parameters to make the simulation match the data (we use a non-linear optimisation package which is part of SciPy[2]).

Recompute Control Policy —Use the newly found parameters to inform an optimisation problem where the new controller parameters are derived to determine an updated control policy c^*. Repeated simulations of the controlled incubator equations (Eq. (2)) are performed with different control duty cycles, to find the optimal one.

Update Control Parameter Finally, the new parameters are uploaded to the controller.

For more details on this self-adaptation loop, we refer the reader to [23].

[2] https://docs.scipy.org/doc/scipy/index.html.

Figure 3a shows an example of self adaptation of the incubator. Initially the duty cycle of the controller has been optimised to keep the temperature at 41 °C. After the lid is opened, an anomaly is detected (shown by the discrepancy between the orange signal and the blue line, at time 500 s). Shortly after, the duty cycle is changed to maximum power, to try to compensate for the loss of energy in the system. The same process happens in reverse when the lid is closed.

3.2 Verified Monitoring Architecture for Safe Self-Adaptation

Whilst self-adaptation through the use of a numerical digital twin such as that described in the previous section provides an effective way of detecting and responding to anomalies, one can question whether the resulting adaptive control policies lead to long-term safe behaviour for the overall system. Indeed, the deployment of such a self-adaptive loop in a safety-critical setting typically requires both offline verification that a safe configuration exists and that the self-adaptive procedure is able to identify it. This is often not possible in practice given the uncertain and unpredicatable system contexts which self-adaptation seeks to address.

We propose an alternative approach in which verified monitoring is deployed online in order to verify the safety of the system after self-adaptation. To this end, we propose to modify the self-adaptation loop architecture introduced in the previous section so that after each anomaly we construct a non-deterministic model of the system denoted as $\mathcal{M}(\mathbf{I}^*, \mathbf{U}, \mathbf{c}^*)$ which aims to over-approximate all possible behaviours of the physical twin after the anomaly, based on the data collected during the **Gather Data** phase. We can then apply verified STL monitoring to $\mathcal{M}(\mathbf{I}^*, \mathbf{U}, \mathbf{c}^*)$ in order to verify a set $\mathcal{S}$ of STL properties representing the safety requirements of the system. Thus, whilst we cannot guarentee that self-adaptation will always succeed, we can use the verified STL monitoring results to guarentee that any unsafe self-adaptations can be detected, enabling the safety of the deployed system to be ensured by other means (such as human intervention or an automated safe-shutdown procedure).

3.3 Uncertainty Calibration

A key stage of the verified monitoring procedure proposed in the previous section is constructing $\mathcal{M}(\mathbf{I}^*, \mathbf{U}, \mathbf{c}^*)$ which attempts to over-approximate the behaviour of the physical incubator system after the anomaly based on the data gathered during the **Gather Data** stage.

To accomplish this we propose to perform an *uncertainty calibration* process in which we start off with a model $\mathcal{M}(\mathbf{x}^*, \mathbf{p}^*, \mathbf{c}) = \mathcal{M}((T_A^*, T_H^*), (C_A^*, G_H^*), c)$ based on the digital twin state $\mathbf{x}^* = (T_A^*, T_H^*)$ at the start of the calibration period, the vector $\mathbf{p}^* = (C_A^*, G_H^*)$ of real-valued parameters determined in the **Recalibrate Model** stage, and the old control policy $\mathbf{c}$. We then expand these real parameters into interval parameters achieving a minimal enclosure of the plant data signal $\mathbf{y} : \{t_1, \ldots, t_N\} \to \mathbb{R}^n$ over the calibration period. To this end we first define the *inflated model*

$$\mathcal{M}_{\boldsymbol{\delta}}^{\boldsymbol{\varepsilon}}(\mathbf{x}^*, \mathbf{p}^*, \mathbf{c}) \triangleq \mathcal{M}\Big(\mathbf{x}^* + [-\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}], \mathbf{p}^* + [-\boldsymbol{\delta}, \boldsymbol{\delta}], \mathbf{c}\Big)$$

of the model $\mathcal{M}(\mathbf{x}^*, \mathbf{p}^*, \mathbf{c})$ by the inflation parameter vectors $\boldsymbol{\varepsilon} = (\varepsilon_A, \varepsilon_H)$ and $\boldsymbol{\delta} = (\delta_A, \delta_H)$. This inflated model encloses each initial condition $\mathbf{x}_i^*$ or parameter $\mathbf{p}_i^*$ of the calibrated digital twin in a radius ε_i or δ_i interval of uncertainty respectively. In order to fit these intervals of uncertainty to the plant data, we define the *non-conformity*[3] of a plant signal $\mathbf{y}$ to a flowpipe $\mathbf{g}$ as

$$\text{non-conformity}(\mathbf{y}, \mathbf{g}) = \sum_{i=1}^{N} \sqrt{\sum_{j=1}^{m} \text{dist}(\mathbf{y}(t_i)_j, \mathbf{g}(t_i)_j)^2}$$

and the *uncertainty* of a n-dimensional interval vector $\mathbf{I} \in \mathbb{IR}^n$ as

$$\text{uncertainty}(\mathbf{I}) = \sqrt{\sum_{i=1}^{n} \text{width}(I_i)^2}.$$

Then suitable inflation parameters $\boldsymbol{\varepsilon}^* = (\varepsilon_A^*, \varepsilon_H^*)$ and $\boldsymbol{\delta} = (\delta_A^*, \delta_H^*)$ are found by applying an optimisation process to minimize,

$$\begin{aligned} &K \,\text{non-conformity}\left(\mathbf{y}, \text{flowpipe}(\mathcal{M}_{\boldsymbol{\delta}}^{\boldsymbol{\varepsilon}}(\mathbf{x}^*, \mathbf{p}^*, \mathbf{c}))\right) \\ &\quad + \text{uncertainty}\left(\text{flowpipe}(\mathcal{M}_{\boldsymbol{\delta}}^{\boldsymbol{\varepsilon}}(\mathbf{x}^*, \mathbf{p}^*, \mathbf{c}))(t_N)\right) \end{aligned}$$

thus jointly minimising the non-conformity of the plant signal to the flowpipe and the uncertainty of the flowpipe at the end of the calibration period (since this gives a measure of how strongly the overall uncertainty in the parameters feeds into the long term behaviour of the system). Here we use the weight parameter $K \gg 1$ to prioritise enclosure of the plant data over minimising the uncertainty in the parameters.

Provided we can find a solution with zero non-conformity, this then provides a non-deterministic model $\mathcal{M}(\mathbf{I}, \mathbf{U}, \mathbf{c}) = \mathcal{M}_{\boldsymbol{\delta}}^{\boldsymbol{\varepsilon}}(\mathbf{x}^*, \mathbf{p}^*, \mathbf{c})$ for the system over the calibration period. To predict the plant behaviour after the anomaly and calibration period have passed, we use a flowpipe from the non-deterministic model to predict the possible system state $\mathbf{I}^* = \text{flowpipe}\big(\mathcal{M}_{\boldsymbol{\delta}^*}^{\boldsymbol{\varepsilon}^*}(\mathbf{x}^*, \mathbf{p}^*, \mathbf{c})\big)(t_N)$ at the end of the calibration period. We can then obtain and utilize the non-deterministic model $\mathcal{M}(\mathbf{I}^*, \mathbf{U}, \mathbf{c}^*)$ where $\mathbf{c}^*$ is the new control policy produced by self-adaptation, as detailed next.

3.4 Self-adaptation Monitoring and Enforcement

We propose that the verified self-adaptation loop may use the non-deterministic model in two ways. Firstly, it may be used in a *monitoring mode* in which we apply verified STL monitoring to the uncertainty-calibrated non-deterministic model after each self-adaptation in order to validate the self-adaptation process itself against the safety requirements $\mathcal{S}$ of the system. Thus, the monitoring mode

[3] This notion is worth comparing this to notions of conformance between continuous and hybrid systems traces such as [19] and [3].

will report the durations of each self-adaptation period, alongside the calibrated intervals of uncertain model parameters for the non-deterministic model after the anomaly, and a record of whether each safety requirement is satisfied or violated after the anomaly. Secondly, it may be used in an *enforcing mode* in which the violation of a safety property blocked the application of the new control policy (deemed unsafe) and instead triggers a safe shutdown of the system, preventing potential harm when self-adaptation is insufficient to ensure safety. Whilst enforcement is necessary to ensure overall safety, in cases where it is not practical, we argue that monitoring provides an invaluable tool to identify design flaws in self-adaptation loops and alert human supervisors to potential issues when safe self-adaptation is not possible.

4 Incubator Self-adaptation Verification Results

In this section we examine some example executions of our formally verified self-adaptation loop for the incubator system in order to explore how the use of verified monitoring allows us to identify unsafe self-adaptations and to correct design flaws in the self-adaptation loop.

4.1 System Setup

We consider an instantiation of the incubator self-adaptation loop which simulates the full digital twinning setup *in silico* based on experimental data and the use another numerical model of the incubator in place of the physical twin. We configure the plant with an initial open-loop control policy $\mathbf{c} = c_{10,30}$ whilst the physical and digital twins are both initially configured with parameters $C_A \approx 177.63$, $G_B \approx 0.77$, $C_H \approx 239.61$, and $G_H \approx 2.32$, and the self-adaptation process is configured to optimize the control policy based on a desired incubator temperature of 41 °C, a data gathering period of 12 seconds, and a minimum period of 20 seconds between anomalies.

In order to simulate a situation requiring self-adaptation, we introduce discontinuous jumps which change the parameter G_B to 10 times its original value after $500s$ and then again after $1500s$. This simulates the box lid being opened at time 500 and closed again at time 1500. We have validated this procedure with the real system, to certify that our simulation results are representative of the real opening of the lid [23].

4.2 Safety Properties

We capture the desired safety properties for controller in the set $\mathcal{S} = \{\phi_1, \phi_2\}$ consisting of the two STL properties:

- $\phi_1 \triangleq \mathcal{F}_{[0,1000]}\,\mathcal{G}_{[0,100]}(T_A \geq 36 \wedge T_A \leq 46)$: the incubator air temperature should stabilise between 36 °C and 46 °C within 1000 seconds;
- $\phi_2 \triangleq \mathcal{G}_{[0,1000]}(T_A \leq 60)$: the incubator air temperature should never exceed 60 °C.

Of these we note that ϕ_2 is the most critical, given the potential for excessive air temperatures inside the box to rapidly harm its contents.

4.3 Self-adaptation Results

Figure 3a demonstrates the behaviour of the incubator self-adaptation loop as it adapts to the opening and closing of the box lid. The self-adaptation loop detects and responds to two anomalies: and anomaly a_1 when the incubator box lid is opened and an anomaly a_2 once the box lid is closed again.

We were able to apply our verified monitoring procedure to check the properties ϕ_1 and ϕ_2 after each self-adaptation. Uncertainty calibration produced non-deterministic models which tightly over-approximated the plant behaviour after reconfiguration; for example, the non-deterministic model flowpipe after a_2 is shown in Fig. 4. This allowed us to generate the monitoring results for each property, producing the results shown in Fig. 3d. From these we can see that ϕ_1 is true after a_1 showing that the system finds a control policy which achieves the desired temperature once the lid is opened. On the other hand, we get a monitoring result of False for ϕ_1 after a_2: whilst the control policy is eventually returning the air temperature to the desired range, it is not able to do so within the time limit stipulated in ϕ_1 (see Fig. 4). More seriously, the crucial property ϕ_2 fails after a_2. We can see that in trying to keep the box close to the desired temperature when the lid is open, the heat-bed temperature has risen to over 70 °C, placing the system in a state where overheating is unavoidable once the lid is closed again and the accumulated energy in the heat-bed dissipates to the air in the box.

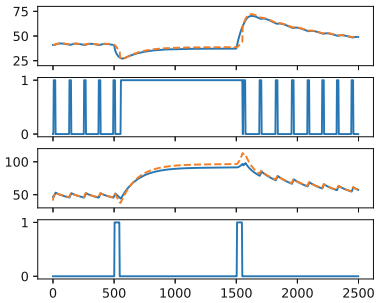
4.4 Repairing the Loop

We can also use verified monitoring as a means to understand and mitigate the potential for unsafe self-adaptations. The failure of ϕ_2 after the box lid was closed again demonstrates the danger of allowing unrestricted heater temperatures, motivating us to introduce an additional safety requirement

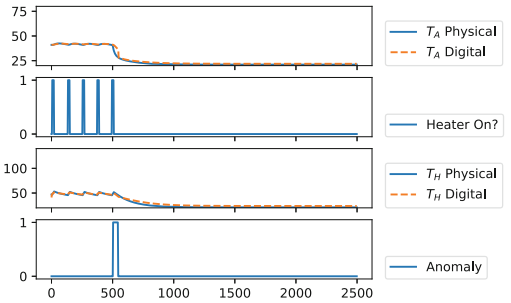
$$\phi_3 \triangleq \mathcal{G}_{[0,1000]}(T_H \leq 70)$$

which requires the heater temperature to be kept below 70 °C for at least 1000 seconds after every self-adaptation. From Fig. 3d this property was violated after each of the two anomalies a_1, a_2 demonstrating the ability of the verified monitor to predict future failures after a self-adaptation. This could allow us to notify a human supervisor or to use the verified monitor in enforcing mode, initiating a safe-shutdown by turning off the heater before an unsafe self-adaptation can lead to directly detectable damage (as shown in Fig. 3b).

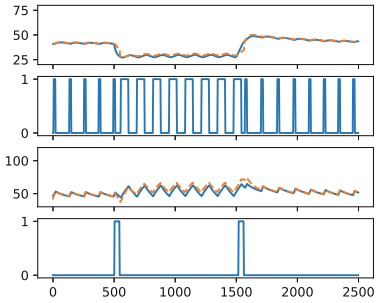
We are also able to use our new understanding of this potential hazard to improve the design of the underlying self-adaptation loop. For example, we can limit the overall heater temperature by modifying the **Recompute Control Policy** stage to select control policies which prevent the heater temperature from



(a) Original Behaviour



(b) Enforcing ϕ_3



(c) Restricted Heater Temperature

Property	Anomaly			
	a_1	a_2	b_1	b_2
ϕ_1	True	False	False	True
ϕ_2	True	False	True	True
ϕ_3	False	False	True	True

(d) Verified Monitoring Results

Fig. 3. Self-adaptation experiment results.

Fig. 4. A demonstration of enclosure of the physical plant signals for T_A and T_H (in yellow and cyan resp.) by the non-deterministic model flowpipes after anomaly a_2. The uncertainty calibration data points for T_A and T_H are highlighted in red and purple respectively. (Color figure online)

exceeding some thresholds (e.g. 60 °C). Repeating the self-adaptation experiment with this modified self-adaptation loop results in the improved behaviour shown in Fig. 3c. We can see in Fig. 3d that all of our safety properties are satisfied after each of the two anomalies b_1, b_2 of the modified system, with one exception: the property ϕ_1 cannot be maintained when the incubator lid is open. This is, however, an understandable trade-off since in an uncontrolled environment there are cases in which self-adaptation is genuinely not possible (given the limitations of the system's physical components) and so we must prioritise between the different design requirements to minimize harm.

5 Conclusion

In this paper we have shown how verified STL monitoring over Flow* flowpipes may be applied to achieve formally-verified self-adaptation in a digital twinning system. We demonstrated the power of this approach to predictively identify unsafe self-adaptations before they lead to a safety violation and to apply active enforcement to prevent violations. Whilst for this paper we have focused on a relatively simple incubator case study, we hope to develop our methods further through a representative range of different of applications of digital twins. Our methods are also not restricted the open-loop control policies considered in this paper, but apply equally to more complex closed-loop control policies.

Limitations and Future work Our initial implementation of our monitoring approach has a number of limitations which we hope to address in future work. Firstly, the accuracy of our predictions is ultimately determined by the representitativeness of the digital twin and the sampled physical twin data from which the non-deterministic model is constructed. This is a limitation we share with most model-based formal methods, however, we can increase the conservativeness of our results by performing uncertainty calibration over sensor data streams which encapsulate the range of possible uncertainty in measurements (by, for example, taking temperature readings from multiple sensors at different locations within the box). We also need to contend with timing differences between the physical and digital twins which may arise from variable processing times and the computation delays between components; these may lead to excessive false positives. We propose to account for such time distortions of signals through the use of more sophisticated notions of conformance such as the Skorokhod metric [19] or Dynamic Time Warping [42].

A final limitation lies in the time required to perform Flow* verified integration to verify a self-adaptation. For the simple case of the incubator, this time was not an issue, but as the example becomes more complex, the verification stage will become a bottleneck in the self-adaptation process. We should be able to substantively overcome this limitation by precomputing the flowpipes offline, extending the approach of Chou, Yoon, and Sankaranarayanan [18], leaving only the moderate [59] cost of our verified monitoring algorithm at runtime. We can also explore combining our verified monitoring approach with other efficient online flowpipe computation methods [15]. Additionally, whilst our focus

thus far has been on control over relatively short timespans, it would also be worth exploring the applicability of our verified monitoring approach to different application domains for digital twins such as infrastructure, civil engineering, and biomedical engineering which feature control over much longer timespans. Whilst we expect our core approach should be equally applicable to models in these domains, the longer timespans of control offer scope for more extensive applications of formal verification inside of the control cycle.

Acknowledgements. Cláudio Gomes and Jim Woodcock are grateful to the Poul Due Jensen Foundation, which has supported the establishment of a new Centre for Digital Twin Technology at Aarhus University. Thomas Wright and Jim Woodcock gratefully acknowledge the support of the UK EPSRC for grant EP/V026801/1, UKRI Trustworthy Autonomous Systems Node in Verifiability. We also thank Jos Gibbons and Juliet Cooke for their feedback and suggestions on drafts of this paper as well as our anonymous reviewers for all of their valuable feedback which fed into the final version of the paper.

References

1. Althoff, M., Dolan, J.M.: Online verification of automated road vehicles using reachability analysis. IEEE Trans. Robot. **30**(4), 903–918 (2014)
2. Althoff, M., et al.: ARCH-COMP18 category report: continuous and hybrid systems with linear continuous dynamics. In: Frehse, G. (ed). ARCH18. 5th International Workshop on Applied Verification of Continuous and Hybrid Systems, vol. 54 of EPiC Series in Computing EasyChair, pp. 23–52 (2018)
3. Araujo, H., et al.: Sound conformance testing for cyber-physical systems: theory and implementation. Sci. Comput. Program. **162**, 35–54 (2018)
4. Aziz, A., Singhal, V., Balarin, F., Brayton, R.K., Sangiovanni-Vincentelli, A.L.: It usually works: the temporal logic of stochastic systems. In: Wolper, P. (ed.) CAV 1995. LNCS, vol. 939, pp. 155–165. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60045-0_48
5. Bartocci, E., et al.: Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications. In: Bartocci, E., Falcone, Y. (eds.) Lectures on Runtime Verification. LNCS, vol. 10457, pp. 135–175. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-75632-5_5
6. Berz, M., Makino, K.: Verified integration of odes and flows using differential algebraic methods on high-order taylor models. Reliab. Comput. **4**(4), 361–369 (1998)
7. Borda, A., Pasquale, L., Koutavas, V., Nuseibeh, B.: Compositional verification of self-adaptive cyber-physical systems. In: 2018 IEEE/ACM 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pp. 1–11. IEEE (2018)
8. Calinescu, R., Rafiq, Y., Johnson, K., Bakır, M.E.: Adaptive model learning for continual verification of non-functional properties. In: Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering, pp. 87–98 (2014)
9. Calinescu, R., Ghezzi, C., Kwiatkowska, M., Mirandola, R.: Self-adaptive software needs quantitative verification at runtime. Commun. ACM **55**(9), 69–77 (2012)
10. Cellier, F.E., Kofman, E.: Continuous System Simulation. Springer, New York (2006). https://doi.org/10.1007/0-387-30260-3

11. Chen, M., Tam, Q., Livingston, S.C., Pavone, M.: Signal temporal logic meets reachability: connections and applications. In: Morales, M., Tapia, L., Sánchez-Ante, G., Hutchinson, S. (eds.) WAFR 2018. SPAR, vol. 14, pp. 581–601. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44051-0_34
12. Chen, X.: Reachability Analysis of Non-Linear Hybrid Systems Using Taylor Models. PhD thesis, Fachgruppe Informatik, RWTH Aachen University (2015)
13. Chen, X., Abraham, E., Sankaranarayanan, S.: Taylor model flowpipe construction for non-linear hybrid systems. In: 2012 IEEE 33rd Real-Time Systems Symposium, pp. 183–192. IEEE (2012)
14. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: an analyzer for non-linear hybrid systems. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 258–263. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_18
15. Chen, X., Sankaranarayanan, S.: Model predictive real-time monitoring of linear systems. In: 2017 IEEE Real-Time Systems Symposium (RTSS), pp. 297–306. IEEE (2017)
16. Chen, Y., Anderson, J., Kalsi, K., Ames, A.D., Low, S.H.: Safety-critical control synthesis for network systems with control barrier functions and assume-guarantee contracts. IEEE Trans. Control Netw. Syst. **8**(1), 487–499 (2021)
17. Chou, Y., Yoon, H., Sankaranarayanan, S.: Predictive runtime monitoring of vehicle models using bayesian estimation and reachability analysis. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2111–2118, October 2020. ISSN: 2153–0866
18. Chou, Y., Yoon, H., Sankaranarayanan, S.: Predictive runtime monitoring of vehicle models using bayesian estimation and reachability analysis. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2111–2118. IEEE (2020)
19. Deshmukh, J.V., Majumdar, R., Prabhu, V.S.: Quantifying conformance using the skorokhod metric. Formal Methods in Sys. Des. 168–206 (2017). https://doi.org/10.1007/s10703-016-0261-8
20. Donzé, A., Raman, V., Frehse, G., Althoff, M.: BluSTL: controller synthesis from signal temporal logic specifications. ARCH@ CPSWeek **34**, 160–168 (2015)
21. Fang, X., et al.: Fast parametric model checking through model fragmentation. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), pp. 835–846. IEEE (2021)
22. Farahani, S.S., et al.: Formal controller synthesis for wastewater systems with signal temporal logic constraints: the Barcelona case study. J. Process Control **69**, 179–191 (2018)
23. Feng, H., et al.: Integration of the MAPE-K loop in digital twins. In: 2022 Annual Modeling and Simulation Conference (ANNSIM), San Diego, California, USA, IEEE (2022)
24. Feng, H., et al.: Introduction to digital twin engineering. In: 2021 Annual Modeling and Simulation Conference (ANNSIM), Fairfax, VA, USA, pp. 1–12. IEEE, July 2021
25. Feng, H., et al. The incubator case study for digital twin engineering. arXiv:2102.10390 [cs, eess], February 2021
26. Feng, H., Gomes, C., Sandberg, M., Macedo, H.D., Larsen, P.G.: Under what conditions does a digital shadow track a periodic linear physical system?. In Software Engineering and Formal Methods. SEFM 2021 Collocated Workshops. SEFM 2021. Lecture Notes in Computer Science, vol. 13230. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-12429-7_11

27. Ghosh, B., Étienne, A.: Offline and online monitoring of scattered uncertain logs using uncertain linear dynamical systems. Technical Report. arXiv:2204.11505. [cs, eess] April 2022
28. Hachicha, M., Halima, R.B., Kacem, A.H.: Formal verification approaches of self-adaptive systems: a survey. Procedia Comput. Sci. **159**, 1853–1862 (2019)
29. 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, Heidelberg (1997). https://doi.org/10.1007/3-540-63166-6_48
30. Hwang, I., et al.: A survey of fault detection, isolation, and reconfiguration methods. In: IEEE Transactions on Control Systems Technology, Conference Name: IEEE Transactions on Control Systems Technology, vol. 18 no. 3, pp. 636–653, May 2010
31. Ishii, D., Yonezaki, N., Goldsztejn, A.: Monitoring temporal properties using interval analysis. IEICE Trans. Fund. Electron. Commun. Comput. Sci. **99**(2), 442–453 (2016)
32. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer **36**(1), 41–50 (2003)
33. Kritzinger, W., et al.: Digital Twin in manufacturing: a categorical literature review and classification. IFAC-PapersOnLine **51**, 1016–1022 (2018)
34. Lee, J., Yu, G., Bae, K.: Efficient SMT-based model checking for signal temporal logic. In: 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 343–354. IEEE (2021)
35. Lin, Q., et al.: Reachflow: an online safety assurance framework for waypoint-following of self-driving cars. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 6627–6632 (2020)
36. Lin, Y., Stadtherr, M.A.: Validated solutions of initial value problems for parametric odes. Appl. Numer. Math. **57**(10), 1145–1162 (2007)
37. Meiyi, M., et al.: Predictive monitoring with logic-calibrated uncertainty for cyber-physical systems. ACM Trans. Embed. Comput. Syst. **20**(5s), 101:1–101:25 (2021)
38. Makino, K., Berz, M.: Suppression of the wrapping effect by taylor model-based verified integrators: long-term stabilization by preconditioning. Int. J. Diff. Equat. Appl. **10**(4), 353–384 (2011)
39. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: Lakhnech, Y., Yovine, S. (eds.) FORMATS/FTRTFT -2004. LNCS, vol. 3253, pp. 152–166. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30206-3_12
40. Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to Interval Analysis, vol. 110. Siam, Philadelphia (2009)
41. Muccini, H., Sharaf, M., Weyns, D.: Self-adaptation for cyber-physical systems: a systematic literature review. In: Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2016, New York, pp. 75–81. Association for Computing Machinery, May 2016
42. Warping, D.T.: In: Meinard, M. (ed.), Information Retrieval for Music and Motion, pp. 69–84. Springer, Berlin (2007). https://doi.org/10.1007/978-3-540-74048-3_4
43. Pant, Y.V., Abbas, H., Mangharam, R.: Smooth operator: control using the smooth robustness of temporal logic. In: 2017 IEEE Conference on Control Technology and Applications (CCTA), pp. 1235–1240, August 2017
44. Qin, X., Deshmukh, J.V.: Clairvoyant monitoring for signal temporal logic. In: Bertrand, N., Jansen, N. (eds.) FORMATS 2020. LNCS, vol. 12288, pp. 178–195. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-57628-8_11

45. Raman, V., et al.: Model predictive control with signal temporal logic specifications. In: 53rd IEEE Conference on Decision and Control, pp. 81–87, December 2014. ISSN: 0191–2216
46. Raman, V., et al.: Reactive synthesis from signal temporal logic specifications. In: Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control, HSCC 2015, New York, pp. 239–248. Association for Computing Machinery, April 2015
47. Roehm, H., Oehlerking, J., Heinz, T., Althoff, M.: STL model checking of continuous and hybrid systems. In: Artho, C., Legay, A., Peled, D. (eds.) ATVA 2016. LNCS, vol. 9938, pp. 412–427. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46520-3_26
48. Sadigh, D., Ashish, K.: Safe control under uncertainty. Technical Report, arXiv:1510.07313 [cs] type: article, arXiv, October 2015
49. Sadraddini, S., Belta, C.: Model predictive control of urban traffic networks with temporal logic constraints. In: 2016 American Control Conference (ACC), pp. 881–881, July 2016. ISSN: 2378–5861
50. Sahin, Y.E., Quirynen, R., Di Cairano, S.: Autonomous vehicle decision-making and monitoring based on signal temporal logic and mixed-integer programming. In: 2020 American Control Conference (ACC), pp. 454–459, July 2020. ISSN: 2378–5861
51. Sanwal, M.U., Hasan, O.: Formal verification of cyber-physical systems: coping with continuous elements. In: Murgante, B., et al. (eds.) ICCSA 2013. LNCS, vol. 7971, pp. 358–371. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39637-3_29
52. Shevtsov, S., Weyns, D., Maggio, M.: Simca*: a control-theoretic approach to handle uncertainty in self-adaptive systems with guarantees. ACM Trans. Auton. Adapt. Syst. **13**(4), 1–34 (2019)
53. da Silva, R.R., Kurtz, V., Lin, H.: Symbolic control of hybrid systems from signal temporal logic specifications. Guidance Navig. Control **01**(02), 2150008 (2021)
54. Tao, F., et al.: Digital twin in industry: state-of-the-art. IEEE Trans. Ind. Inf. **15**(4), 2405–2415 (2019)
55. Tsigkanos, C., et al.: On the interplay between cyber and physical spaces for adaptive security. IEEE Trans. Dependable Secur. Comput. **15**(3), 466–480 (2016)
56. Waga, M., et al.: Model-bounded monitoring of hybrid systems. In: Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems, pp. 21–32. Association for Computing Machinery, New York, May 2021
57. Weyns, D., et al.: A survey of formal methods in self-adaptive systems. In: Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering - C3S2E 2012, Montreal, Quebec, Canada, pp. 67–79. ACM Press (2012)
58. Woodcock, J., Gomes, C., Macedo, H.D., Larsen, P.G.: Uncertainty quantification and runtime monitoring using environment-aware digital twins. In: Margaria, T., Steffen, B. (eds.) ISoLA 2020. LNCS, vol. 12479, pp. 72–87. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-83723-5_6
59. Wright, T., Stark, I.: Property-directed verified monitoring of signal temporal logic. In: Deshmukh, J., Ničković, D. (eds.) RV 2020. LNCS, vol. 12399, pp. 339–358. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60508-7_19
60. Yoon, H., Chou, Y., Chen, X., Frew, E., Sankaranarayanan, S.: Predictive runtime monitoring for linear stochastic systems and applications to geofence enforcement for UAVs. In: Finkbeiner, B., Mariani, L. (eds.) RV 2019. LNCS, vol. 11757, pp. 349–367. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32079-9_20

61. Yu, X., et al.: Online monitoring of dynamic systems for signal temporal logic specifications with model information. Technical Report. arXiv:2203.16267 [cs, eess] type: article, arXiv, March 2022
62. Zhang, L., Chen, X., Kong, F., Cardenas, A.A.: Real-time attack-recovery for cyber-physical systems using linear approximations. In: 2020 IEEE Real-Time Systems Symposium (RTSS), pp. 205–217, December 2020. ISSN: 2576-3172