Analysis and Design of Cyber-Physical Systems: A Hybrid Control Systems Approach

Ricardo G. Sanfelice
Department of Computer Engineering
University of California
Santa Cruz, CA 95064

Abstract

Cyber-physical systems combine digital and analog devices, interfaces, networks, computer systems, and the like with the natural and man-made physical world. The inherent interconnected and heterogeneous combination of behaviors in these systems makes their analysis and design a challenging task. Safety and reliability specifications imposed in cyber-physical applications, which are typically translated into stringent robustness standards, aggravate the matter. Unfortunately, state-of-the-art tools for system analysis and design cannot cope with the intrinsic complexity in cyber-physical systems. Tools suitable for analysis and design of cyber-physical systems must allow a combination of physical (or continuous dynamics) and the cyber (or computational components), as well as handle a variety of types of perturbations, such as exogenous disturbances, time delays, and system failures. This article proposes a hybrid control systems approach to analysis and design of cyber-physical systems. Due to the capability of capturing continuous and discrete dynamics, hybrid system models [1] are a very natural framework for the study of cyber-physical systems. This article provides an overview of methods from the literature of hybrid control theory that are suitable for modeling, analysis, and design of cyber-physical systems. The ideas are illustrated in several examples throughout the article.

Index Terms

Modeling, invariance, stability, temporal logic, robustness, simulation, cyber-physical systems

I. Introduction

A *cyber-physical system* is a system combining physical and computer or cyber components. The physical components consist of systems existing in nature, such as biological entities as well as those developed by humans, such as transportation and energy-producing systems. This component exists, operates, and interacts with its environment in continuous or ordinary time. The computational component consists of systems and entities involved in processing, communicating, and controlling information via computational means. These include algorithms implemented in software and digital systems, interfaced to the physical components via analog-to-digital and digital-to-analog converters and digital communication networks. They are man-made systems that operate in discrete time or in an event-driven fashion. Evidently, the complexity of component

integration in cyber-physical systems stems from the fact that the computational component is distributed throughout the system, and tightly coupled with the physical component. As a consequence, cyber-physical systems are highly interconnected systems combining continuous and discrete dynamics.

This article considers cyber-physical systems that are given by physical components, cyber components, and the systems needed to interconnect them. The physical components include the systems existing in the physical world, e.g., a process to be monitored or controlled. The cyber components capture the computer algorithms in the system, e.g., a communication or a control algorithm. The subsystems used to interconnect them include interfaces, converters, signal conditioners, networks, among others. Models from the theory of hybrid systems are employed to capture the mixed continuous and discrete behavior of some of these subsystems and of the entire interconnection. More specifically, dynamical models of physical components, cyber components, and systems used to interface them are proposed within the framework for hybrid dynamical systems in [1]. In this framework, hybrid dynamical systems are given by the combination of differential inclusions and difference inclusions with constraints, namely, *hybrid inclusions*. Such a combination leads to dynamical models with variables that may change continuously as determined by the differential inclusions while, at times, may also change discretely according to the difference inclusions. These capabilities make hybrid inclusions very suitable for mathematical modeling of cyber-physical systems. Furthermore, the stability and robustness of hybrid inclusion models of cyber-physical systems can be systematically analyzed using the theory recently developed in [1] for hybrid dynamical systems without inputs.

Motivated by the recent results on modeling, analysis, and design of hybrid inclusions, this article compiles a collection of methods for hybrid inclusions that are suitable for the study of the class of cyber-physical systems of interest here. A mathematical model for the physical components in terms of differential inclusions with set-based enabling conditions on the state and input is given in Section II-A. This model captures the continuous dynamics of a hybrid system with inputs, which not only extends the model in [1] but also is required to be able to define an interconnection between the physical components and the other subsystems; see also the models in [2]. Several mathematical models of cyber components are given in Section II-B in terms of difference inclusions that are enabled when certain conditions on the state and input are satisfied. First, following the classical definitions in [3] and the models in [4], models of finite state machines (FSMs) in terms of difference inclusions are presented to capture the evolution of deterministic and nondeterministic FSMs (pure and with conditional structures). The model proposed for FSMs with conditional structures implements such conditions in terms of set conditions involving the state, input, and output. Next, models of digital computations and discrete-time algorithms are introduced to capture one-shot and iterative computations, as well as the algorithms obtained when discretizing the dynamics of a continuous-time system. The examples in Section II-B illustrate how these models can be used to describe operations performed by the algorithms implemented in the software of cyber-physical systems. The models of the systems used to interconnect physical and cyber components are introduced in Section II-C. Due to being at the interface between physical and cyber components, these models involve both differential and difference equations. A model of the system performing the conversion of analog signals into digital signals is first proposed. This model uses a timer to trigger the sampling events and a memory state to store the samples. A similar model is proposed to capture the operation of digital-to-analog converters of zero-order hold type. In addition to models of converters, a model of a digital network in which transmission of data occurs with a rate that is within a pre-specified

range is proposed. The model is nondeterministic due to using a difference inclusion to model all of the possible transmission rates within the allowed range. The combination of the models in Section II-A-II-C leads to a hybrid inclusion model. Section II-D includes examples of such combinations and motivates the hybrid inclusion modeling approach for cyber-physical systems that is advocated in this article.

With a hybrid inclusion model for the class of cyber-physical of interest, Section III summarizes tools for analysis and design. Following the ideas in [1], a notion of time and execution for these systems is first introduced in Section III-A. In this notion, time is hybrid in the sense that a time instant uniquely determines the amount of time that variables have evolved continuously as well as discretely. Then, executions (or solutions) are given by a state and input pair that is parameterized by the said notion of hybrid time. Solutions are classified according to the properties of their domain of definition, which are given by sets called hybrid time domains. Conditions guaranteeing some of the properties of their hybrid time domains are given at the end of Section III-A. Using this notion of solution, a definition of an *invariant* for the proposed model of a cyber-physical system is proposed in Section III-B. Invariants are sets of particular interest, for example, when studying reachability and safety. Sufficient conditions useful for determining the invariants of a system are given. Section III-C pertains to stability and attractivity, which are properties of particular interest to cyber-physical systems with a feedback topology. Following [1], a Lyapunov stability notion for sets is introduced and a sufficient condition in terms of Lyapunov functions is presented. Attractivity of a set is also defined and an invariance principle for hybrid inclusions from [5] is presented as a tool to determine the set of points to which solutions converge. In addition to methods for the study of invariants and stability, conditions to determine whether a function of the state is true or not is proposed following the theory of temporal logic. Operators and associated semantics are defined for solutions to hybrid inclusions. Conditions on their solutions for the satisfaction of specific formulae are given in Section III-D. Further conditions under which the properties of a cyber-physical system modeled as a hybrid inclusion are robust to small perturbations are given in Section III-E. Robustness to small perturbations is mandatory for cyber-physical systems since several of the perturbations introduced during the implementation of the cyber components, while cannot always be neglected, can be assumed to be small, e.g., quantization and discretization effects. Finally, a brief overview of a tool for simulating hybrid inclusions is given in Section III-F.

II. Dynamical Models of Cyber-physical Systems

In this article, the temporal evolution of the variables of a cyber-physical system are captured using dynamical models. The state of the physical components is typically determined by variables that change according to physical time (or ordinary time) and take values from a dense set (e.g., real numbers). On the other hand, the state of the cyber components is usually defined by variables that change within the code, which is executed at discrete-time events, and that take values from discrete sets. Unavoidably, this heterogeneous combination of variables and notions of time requires dynamical models that combine continuous and discrete variables as well as notions of time.

In this paper, we advocate that hybrid dynamical system models can be employed to capture the behavior of cyber-physical systems. More precisely, the evolution of the continuous variables is captured by *differential inclusions* while the evolution of the discrete variables is captured by *difference inclusions*. These inclusions (or equations) are typically nonlinear due to the complexity of the dynamics of those variables. Furthermore, conditions determining the

change of the continuous and discrete variables according to the said equations/inclusions can be conveniently captured by functions of the variables, inputs, and outputs. A particular advantage provided by hybrid models of cyber-physical systems is that a notion of time is automatically imposed on the cyber component, due to the fact that a dynamical model of a cyber-physical system will naturally have a concept of solution attached to it. In fact, a solution to such models will be parameterized by a notion of time so as to determine the change of the continuous variables, according to the physical components, as well as the discrete variables, according to the cyber components. In the next sections, these models are introduced in detail and illustrated in examples.

A. Models of physical components

The physical components of a cyber-physical system include the analog elements, physical systems, and the environment. Among the many possible models available, we will capture the dynamics of the physical components by differential equations or inclusions. This type of semantic is very common in the modeling of the so-called *environment* in embedded systems [6], the system to study in dynamical systems theory [7], and the plant to control in control theory [8]. The proposed model consists of a continuous-time system, in which typically the time variable $t \in \mathbb{R}_{\geq 0}$ parameterizes the variables of the system, which are called *states*. For example, differential equations arise when studying the physics of systems, such as analog devices, electro-mechanical systems, chemical systems, etc. Differential inclusion models allow for the derivative of the state of a system to take values from a set (rather than being assigned by a single-valued function). Such models are useful to model systems that exhibit nondeterminism, perhaps due to uncertainty on its dynamics or parameters, to capture families of trajectories with a single dynamical model, and to model regularizations of nonsmooth systems.

We denote the state variable of the physical components by z with state space given by the Euclidean space $\mathbb{R}^{n_P}$. Its dynamics are defined by a differential inclusion with right-hand side defined by the set-valued map F_P. We let $u \in \mathbb{R}^{m_P}$ denote the input signals affecting the physical components and $y \in \mathbb{R}^{r_P}$ to be the output defined by the output function h which is a function of the state z and of the input u. With these definitions, the mathematical description of the physical components is given by

$$\dot{z} \in F_P(z, u), \quad y \;=\; h(z, u) \tag{1}$$

where F_P and h are functions from $\mathbb{R}^{n_P} \times \mathbb{R}^{m_P}$ mapping into $\mathbb{R}^{n_P}$ and $\mathbb{R}^{r_P}$, respectively. The notation $\dot{z}$ represents the derivative of $t \mapsto z(t)$ with respect to $t \in [0, \infty) =: \mathbb{R}_{\geq 0}$, i.e., $\dot{z} = \frac{d}{dt} z$. The inclusion in (1) indicates that, at each (z, u), the time derivative of z is taken from the set $F_P(z, u)$. If F_P is single valued then (1) reduces to a differential equation.

In certain cases, it would be needed to impose restrictions on the state and inputs to the physical component. Such is the case when it is needed to limit the range of their allowed values (e.g., the state belongs to a manifold or the input is constrained), or to impose conditions that they should satisfy so as to evolve continuously according to (1). Such conditions can be modeled using sets, namely,

$$(z, u) \in C_P \subset \mathbb{R}^{n_P} \times \mathbb{R}^{m_P} \tag{2}$$

The model of the physical components is given by (1)-(2).

The following two examples illustrate the model proposed in (1). While the examples omit such features, the model in (1) can easily incorporate the dynamics of analog filters, sensors, and actuators.

Example 2.1: A linear time-invariant model of the physical component is defined by

$$F_P(z,u) = A_P z + B_P u, \qquad h(z,u) = M_P z + N_P u$$

where A_P, B_P, M_P, and N_P are matrices of appropriate dimensions. State and input constraints can directly be embedded into the set C_P. For example, the constraint that z has all of its components nonnegative and that u has its components with norm less or equal than one is captured by

$$\begin{aligned} C_P \;\; = \;\; & \{(z,u) \in \mathbb{R}^{n_P} \times \mathbb{R}^{m_P} \,:\, z_i \geq 0 \;\forall i \in \{1,2,\ldots,n_P\}\} \\ & \cap \{(z,u) \in \mathbb{R}^{n_P} \times \mathbb{R}^{m_P} \,:\, |u_i| \leq 1 \;\forall i \in \{1,2,\ldots,m_P\}\} \end{aligned}$$

For example, the evolution of the temperature of a room with a heater can be modeled by a linear-time invariant system with state z denoting the temperature of the room and with input $u = (u_1, u_2)$, where u_1 denotes whether the heater is turned on ($u_1 = 1$) or turned off ($u_1 = 0$) while u_2 denotes the temperature outside the room. The evolution of the temperature is given by

$$\dot{z} \;\; = \;\; -z + \begin{bmatrix} z_\Delta & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad \text{when } (z,u) \in C_P = \big\{(z,u) \in \mathbb{R} \times \mathbb{R}^2 \,:\, u_1 \in \{0,1\}\big\} \quad (3)$$

where z_Δ is a constant representing the heater capacity. △

Example 2.2: A widely used kinematic model of a ground vehicle moving on the plane is the so-called Dubins vehicle. The vehicle is assumed to be a particle on the plane that describes smooth paths and has the capability of making turns satisfying a minimum turning radius constraint (similar to a car). Denoting its position by $(z_1, z_2) \in \mathbb{R}^2$ and its orientation by $z_3 \in \mathbb{R}$ (with respect to the vertical axis), the dynamics of the particle defining a Dubins vehicle are given by

$$\dot{z}_1 = \nu \sin z_3, \qquad \dot{z}_2 = \nu \cos z_3, \qquad \dot{z}_3 = u$$

where ν is the velocity of the vehicle, $u \in [-\overline{u}, \overline{u}]$ is the angular velocity input, $\overline{u} = \frac{\nu}{\rho}$, and ρ is the minimum turning radius constraint. Assuming, for simplicity, that ν is a constant, this mathematical model can be captured as in (1)-(2) with

$$F_P(z,u) = \begin{bmatrix} \nu \sin z_3 \\ \nu \cos z_3 \\ u \end{bmatrix}, \qquad C_P = \big\{(z,u) \in \mathbb{R}^3 \times \mathbb{R} \,:\, u \in [-\overline{u}, \overline{u}]\big\}$$

△

The model in (1)-(2) captures the continuous dynamics of a hybrid system with inputs, which extends the model in [1] and is required to be able to define an interconnection between the physical components and the other subsystems; see also the models in [2].

B. Models of cyber components

The cyber components of a cyber-physical system include those in charge of performing computations, implementing algorithms, and transmitting digital data over networks. The tasks performed by the code (at the software level) and the logic-based mechanisms (at the circuit level) involve variables that only change at discrete events, not necessarily periodically. Furthermore, unlike most of the quantities involved in the physical components, such variables may take value

from discrete sets rather than from a continuum. Due to these unique features, models of cyber components have state variables, potentially discrete valued, that are updated at discrete events. In this paper, we capture the dynamics of such variables using differential equations/inclusions

We denote the state variable of the cyber components by $\eta \in \Upsilon$, where $\Upsilon \subset \mathbb{R}^{n_C}$ is the state space. The dynamics of η are defined by a difference inclusion with right-hand side defined by the set-valued map G_C. We let $v \in \mathcal{V} \subset \mathbb{R}^{m_C}$ denote the input signals affecting the cyber components and $\zeta \in \mathbb{R}^{r_C}$ to be the output defined by the output function κ, which is a function of the state η and of the input v. With these definitions, the general mathematical description of the cyber component is

$$\eta^+ \in G_C(\eta, v), \quad \zeta \;=\; \kappa(\eta, v) \tag{4}$$

In certain cases, it would be needed to impose restrictions on the state and inputs to the cyber component. Such conditions can be modeled imposing that η and v belong to a subset of their state space, namely,

$$(\eta, v) \in D_C \subset \Upsilon \times \mathcal{V} \tag{5}$$

The model of the cyber components is given by (4)-(5). Next, we provide specific constructions of models of cyber components.

1) Pure Finite State Machines: A finite state machine (FSM) or deterministic finite automaton (DFA) is a system with inputs, states, and outputs taking values from discrete sets that are updated at discrete transitions (or jumps) triggered by its inputs. At every jump, the states and the outputs of the finite state machine are updated. Let v denote the inputs, q denote the states (or mode), and r denote the outputs of the FSM. Following the definition in [3], a finite state machine consists of the following objects:

- An input alphabet Σ where v takes values from;
- A finite set of states Q where q takes values from;
- A set of output symbols Δ where r takes values from;
- An output function $\kappa : Q \to \Delta$; and
- A transition function $\delta : Q \times \Sigma \to Q$.

The initial state of the FSM is denoted q_0 while, at times, a set of final states is imposed and denoted as $Q_\infty \subset Q$. The output function is defined for the current value of the state and input, which, in particular, permits modeling Mealy machines (one way to model Moore machines is to include in the model an additional variable capturing the output and updating it at transitions using the value of the state and input before the transitions). The transition function is defined for each state $q \in Q$ and each input $v \in \Sigma$; moreover, by convention, $\delta(q, \emptyset) = q$, which is known as the basis condition, and $\delta(q, ab) = \delta(\delta(q, a), b)$ for each $a, b \in \Sigma$, which implies that δ can be evaluated for an input string, i.e., δ satisfies the properties of an *extended transition function*. Note that the model is deterministic since the transition and output functions are uniquely defined for each value of their arguments. Also, while functions δ and κ of the FSM are assumed to be defined for each element in $Q \times \Sigma$, i.e., they are *total functions*, a FSM with functions not defined for all points, i.e., *partial functions*, can be modeled similarly.

Then, given a FSM and an initial state $q_0 \in Q$, a transition to a state $q_1 = \delta(q_0, v)$ is performed when an input $v \in \Sigma$ is applied to it. After the transition, the output of the FSM is updated to $\kappa(q_1)$. This mechanism can be captured by the difference equation

$$q^+ \;=\; \delta(q, v) \qquad \zeta \;=\; \kappa(q) \qquad (q, v) \in Q \times \Sigma \tag{6}$$

This model corresponds to the model of the cyber components in (4)-(5) with

$$\eta = q, \qquad \Upsilon = Q, \qquad \mathcal{V} = \Sigma, \qquad G_C = \delta, \qquad D_C = \Upsilon \times \mathcal{V}$$

Note that there is no notion of time associated with the FSM model above.

Example 2.3: The finite state machine given in Figure 1 has two modes and one input. Its output is equal to the current mode. It is given by the difference equation in (6) with

$$Q = \{A, B\}, \quad \Sigma = \{0, 1\}, \quad \delta(q, v) = \begin{cases} A & \text{if } v = 1 \\ B & \text{if } v = 0 \end{cases}, \quad \kappa(q) = q$$

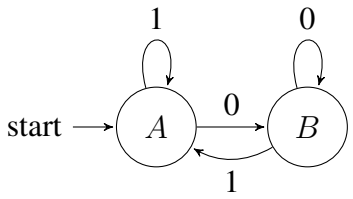


Fig. 1. A finite state machine with two modes and one input.

Note that this FSM does not terminate. △

The nondeterministic case of a FSM can be treated similarly by using a set-valued transition function $\delta : Q \times \Sigma \rightrightarrows Q \times \Delta$. Note that in this case

$$\delta(q, ab) = \bigcup_{q' \in \delta(q,a)} \delta(q', b)$$

for each $a, b \in \Sigma$.

2) Finite State Machines with Conditional Structures as Guards: In many applications, it is desired that the jumps of the FSM are triggered based on conditional structures, e.g., perform a transition when $v < 0$. To model conditional structures within a pure FSM, the input alphabet Σ has to be of infinite size. Conditional structures can be added to a pure FSM by allowing for an infinite alphabet and including the conditional structure as a *guard*, i.e., a boolean-valued expression that evaluates to *true* when the transition is enabled, and to *false* otherwise. As pointed out in [4], the dataflow model of computation is appropriate to trigger transitions in a FSM according to conditional structures; e.g., the comparison $u_c < 0$ amounts to externally compute (in a dataflow model) the conditional statement and based on its result, either true or false, trigger a jump of the FSM.

To define a FSM with transitions according to conditional structures, let the function $\ell : Q \times \Sigma \times \Delta \to \mathbb{R}$ be a testing function for the condition on the transition for each mode $q \in Q$. Assume that the value of $\ell(q, v, \zeta)$ is larger than zero if the conditional condition to implement is not satisfied for the current values of (q, v, ζ), and that is less or equal than zero if it is satisfied. Then, a FSM with transitions triggered by the conditional structure modeled by ℓ is given by

$$q^+ \;=\; \delta(q, v) \qquad \zeta \;=\; \kappa(q) \qquad\qquad \ell(q, v, \zeta) \leq 0, \;\; (q, v) \in Q \times \Sigma \tag{7}$$

This model corresponds to the model of the cyber components in (4)-(5) with

$$\eta = q, \qquad \Upsilon = Q, \qquad \mathcal{V} = \Sigma, \qquad G_C = \delta, \qquad D_C = \{(q, v) \in Q \times \mathcal{V} \,:\, \ell(q, v, \kappa(q)) \leq 0\}$$

The finite state machine with data flow is a powerful model as it can be used to describe a control data flow graph (CDFG). CDFGs are a common intermediate representation in software compilation. Many programming languages are translated into a CDFG during their translation to assembly code. The CDFG is also a common model in hardware compilation; it is frequently used as an entry model into behavioral/high-level synthesis.

Note that as in the pure FSM model in Section II-B.1, there is no notion of time associated with the model in (7).

Finally, similar mathematical models can be derived for other machines used in practice, such as Turing machines, as well as Büchi and pushdown automata.

3) Models of Computer Computations and Discrete-time Algorithms: Computations performed on a computer can be modeled as a purely discrete system that, after one or a series of steps, provides the outcome of the computations. The computation to be performed may require inputs v and the result of the computation could be used to determine the output ζ of the model. Computations that can be performed in one step of the discrete system can be modeled by the static system

$$\zeta = \widetilde{\kappa}(v) \tag{8}$$

where the function $\widetilde{\kappa}$ models the computations being performed. This model corresponds to the model of the cyber components in (4)-(5) with

$$\eta = \emptyset, \qquad \Upsilon = \emptyset, \qquad \mathcal{V} = \Sigma, \qquad G_C = \emptyset, \qquad D_C = \mathcal{V}, \qquad \kappa = \widetilde{\kappa}$$

Example 2.4: The check $v < 0$ in Section II-B.2 can be modeled by

$$\widetilde{\kappa}(v) := \begin{cases} 1 & v < 0 \\ 0 & \text{otherwise} \end{cases}$$

△

Example 2.5: The computation of the factorial of an integer $a > 0$ is given by the evaluation at $v = a$ of the function[1]

$$\widetilde{\kappa}(v) := v(v-1)(v-2)\dots(v-(v-2))1 \tag{9}$$

△

Iterative implementations of computations require a number of steps to reach an answer and, at times, additional variables. Denoting the additional variables as $m \in \mathbb{R}^{n_C-1}$ and the counter as $k \in \{0, 1, 2, \dots, k^*\}$, $k^* \in \{0, 1, 2, \dots\} =: \mathbb{N}$, we define a discrete system that performs k^* iterations to provide the final outcome of the computations. Denoting $\eta = [m^\top k]^\top$ as the state, v as the input, and $\widetilde{\kappa}$ as the routine performing the computations at each iteration, the model is given by

$$\eta^+ = \begin{bmatrix} \widetilde{\kappa}(m, k, v) \\ k+1 \end{bmatrix} \qquad \zeta = m \qquad k \in \{0, 1, 2, \dots, k^*-1\}, m \in \mathbb{R}^{n_C-1}, v \in \mathcal{V} \tag{10}$$

Example 2.6: The one-step computation of the factorial of the integer a in (9) can be performed iteratively by initializing m to a and k to one, setting $k^* = a$, and defining

$$\widetilde{\kappa}(m, k, v) = m(m-k)$$

[1]This operation can be performed recursively by defining the function $\text{fact}(v) = v\,\text{fact}(v-1)$ when $v > 1$, and $\text{fact}(1) = 1$, and using $\widetilde{\kappa}(v) = \text{fact}(v)$.

Note that for $k < k^*$, ζ is equal to the intermediate result $a! - (a-k)!$. $\triangle$

The model in (10) corresponds to the model of the cyber components in (4)-(5) with

$$\eta = \begin{bmatrix} m \\ k \end{bmatrix}, \qquad \Upsilon = \mathbb{R}^{n_C - 1} \times \{0, 1, 2, \ldots, k^*\}, \qquad \mathcal{V} = \Sigma, \qquad G_C = \begin{bmatrix} \widetilde{\kappa}(m, k, v) \\ k+1 \end{bmatrix}$$

and

$$D_C = \mathbb{R}^{n_C - 1} \times \{0, 1, 2, \ldots, k^* - 1\}, \qquad \kappa(\eta) = m \quad \forall \eta \in \Upsilon$$

Discrete-time algorithms are naturally modeled using difference equations. For instance, feedback controllers in difference equation form are obtained after a discretization of a controller designed using continuous-time control design tools, or from directly designing a controller for a discretized model of the plant. Such discrete-time algorithms can be written as

$$\eta^+ \;=\; G_C(\eta, v) \qquad \zeta \;=\; \kappa(\eta) \tag{11}$$

where G_C is obtained from discretizing the control algorithm.

Example 2.7: If the original controller is given by the system in state-space form

$$\dot{\xi} \;=\; A_K \xi + B_K v, \qquad \zeta_K \;=\; M_K \xi + N_K v$$

with $\xi \in \mathbb{R}^{n_C}$ and $v \in \mathbb{R}^{m_C}$, then its discretization with period T^* is given by the discrete-time algorithm

$$\eta^+ \;=\; A_K^d \eta + B_K^d v, \qquad \zeta \;=\; M_K^d \eta + N_K^d v$$

where

$$A_K^d = \exp(A_K T^*), \qquad B_K^d = \left(\int_0^{T^*} \exp(A_K s) ds \right) B_K, \qquad M_K^d = M_K, \qquad N_K^d = N_K$$

which is already in the form of (11). Also, this model is already in the form of the model of the cyber components in (4)-(5), in which $\Upsilon = \mathbb{R}^{n_C}$ and $\mathcal{V} = \mathbb{R}^{m_C}$. Note that while this model does not have a notion of time associated to it, at times, like for the discretized controller, the model should be executed at specific time instants. $\triangle$

The examples in this section illustrate how these models can be used to describe operations performed by the algorithms implemented in the software of cyber-physical systems. Next, we consider the models of the systems enabling the interconnection between the physical and cyber components.

C. Models of systems at the interface between physical and cyber components

The models describing the behavior of the physical and the cyber components have significantly different dynamics. Due to this, their interconnection requires interfaces that condition and convert the signals appropriately. Below, we propose mathematical models for some of the most widely used interfaces. In the section to follow, these models will be used to define a complete model of a cyber-physical system.

1) Analog-to-Digital Converters: Analog-to-digital converters (ADCs), or simply sampling devices, are commonly used to provide measurements of the physical systems to the cyber components. Their main function is to sample their input, which is usually the output of the sensors measuring the output y, at a given periodic rate T_s^* and to make these samples available to the embedded computer. A basic model for a sampling device consists of a timer state and a sample state. When the timer reaches the value of the sampling time T_s^*, the timer is reset to zero and the sample state is updated with the inputs to the sampling device.

The model for the sampling device we propose has both continuous and discrete dynamics. If the timer state has not reached T_s^*, then the dynamics are such that the timer state increases continuously with a constant, unitary rate. When T_s^* is reached, the timer state is reset to zero and the sample state is mapped to the inputs of the sampling device. To implement this mechanism, we employ a timer state $\tau_s \in \mathbb{R}_{\geq 0}$ and a sample state $m_s \in \mathbb{R}^{r_P}$. The input to the sampling device is denoted by $v_s \in \mathbb{R}^{r_p}$. The model of the sampling devices is

$$\dot{\tau}_s = 1, \quad \dot{m}_s = 0 \qquad \text{when} \quad \tau_s \in [0, T_s^*] \tag{12}$$

$$\tau_s^+ = 0, \quad m_s^+ = v_s \qquad \text{when} \quad \tau_s \geq T_s^* \tag{13}$$

In practice, there exists a time, usually called *the ADC acquisition time*, between the triggering of the ADC with the sampling device and the update of its output. Such a delay limits the number of samples per second that the ADC can provide. Additionally, an ADC can only store in the sample state finite-length digital words, which causes quantization. The model above omits effects such as acquisition delays and quantization effects, but those can be incorporated if needed. In particular, quantization effects can be added to the model in (12)-(13) by replacing the update law for m_s to $m_s^+ = \text{round}(v_s)$, where the function round is such that $\text{round}(v_s)$ is the closest number to v_s that the machine precision can represent.

2) Digital-to-Analog Converters: The digital signals in the cyber components need to be converted to analog signals for their use in the physical world. Digital-to-analog converters (DACs) perform such a task by converging digital signals into analog equivalents. One of the most common models for a DAC is the zero-order hold model (ZOH). In simple terms, a ZOH converts a digital signal at its input into an analog signal at its output. Its output is updated at discrete time instants, typically periodically, and held constant in between updates, until new information is available at the next sampling time. We will model DACs as ZOH devices with dynamics similar to (12)-(13). Let $\tau_h \in \mathbb{R}_{\geq 0}$ be the timer state, $m_h \in \mathbb{R}^{r_C}$ be the sample state (note that the value of h indicates the number of DACs in the interface), and $v_h \in \mathbb{R}^{r_C}$ be the inputs of the DAC. Its operation is as follows. When $\tau_h \geq T_h^*$, the timer state is reset to zero and the sample state is updated with v_h (usually the output of the embedded computer). A model that captures this mechanism is given by

$$\dot{\tau}_h = 1, \quad \dot{m}_h = 0 \qquad \text{when} \quad \tau_h \in [0, T_h^*] \tag{14}$$

$$\tau_h^+ = 0, \quad m_h^+ = v_h \qquad \text{when} \quad \tau_h \geq T_h^* \tag{15}$$

3) Digital Networks: The information transfer between the physical and cyber components, or between subsystems within the cyber components, might occur over a digital communication network. The communication links bridging each of these components are not capable of continuously transmitting information, but rather, they can only transmit sampled (and quantized) information at discrete time instants. Combining the ideas in the models of the converters in the

previous sections, we propose a model of a digital network link that has a variable that triggers the transfer of information provided at its input, and that stores that information until new information arrives. We assume that the transmission of information occurs at instants $\{t_i\}_{i=1}^{i^*}$, $i^* \in \mathbb{N} \cup \{\infty\}$, satisfying

$$T_N^{*\min} \leq t_{i+1} - t_i \leq T_N^{*\max} \qquad \forall i \in \{1, 2, \ldots, i^* - 1\}$$

where $T_N^{*\min}$ and $T_N^{*\max}$ are constants satisfying

$$T_N^{*\min}, T_N^{*\max} \in [0, \infty]$$

and

$$T_N^{*\min} \leq T_N^{*\max}$$

and i^* is the number of transmission events, which might be finite or infinite. The constant $T_N^{*\min}$ determines the minimum possible time in between transmissions while the constant $T_N^{*\max}$ defines the maximum amount of time elapsed between transmissions. In this way, a communication channel that allows transmission events at a high rate would have $T_N^{*\min}$ small (zero for infinitely fast transmissions), while one with slow data rate would have $T_N^{*\min}$ large. The constant $T_N^{*\max}$ determines how often transmissions may take place. Note that the constants $T_N^{*\min}$ and $T_N^{*\max}$ can be generalized to functions so as to change according to other states or inputs.

At every t_i, the information at the input v_N of the communication link is used to update the internal variable m_N, which is accessible at the output end of the network and remains constant between communication events. This internal variable acts as an information buffer, which can contain not only the latest piece of information transmitted but also previously transmitted information. A mathematical model capturing the said mechanism is given by

$$\dot{\tau}_N = -1, \quad \dot{m}_N = 0 \qquad \text{when} \quad \tau_N \in [0, T_N^{*\max}] \tag{16}$$

$$\tau_N^+ \in [T_N^{*\min}, T_N^{*\max}], \quad m_N^+ = v_N \qquad \text{when} \quad \tau_N \leq 0 \tag{17}$$

Note that the update law for τ_N at jumps is given in terms of a difference inclusion, which implies that the new value of τ_N is taken from the set $[T_N^{*\min}, T_N^{*\max}]$. The dimension of the states and the input would depend on the type of components that connect to and from it, and also the size of data transmitted and buffered. Similar to the models proposed for conversion, the model of the digital link (16)-(17) does not include delays nor quantization, but such effects can be incorporated if needed.

Note that the models in this section are given in terms of differential and difference equations/inclusions. The reason for this is that they are at the interface between physical and cyber components. These type of models are referred to as *hybrid inclusions* and will be the class of models we will employ to study cyber-physical systems in the remainder of this article. In general, the interface will be modeled as a hybrid inclusion with state λ, input w, output ψ, and dynamics

$$\dot{\lambda} \in F_I(\lambda, w) \qquad \text{when } (\lambda, w) \in C_I \tag{18}$$

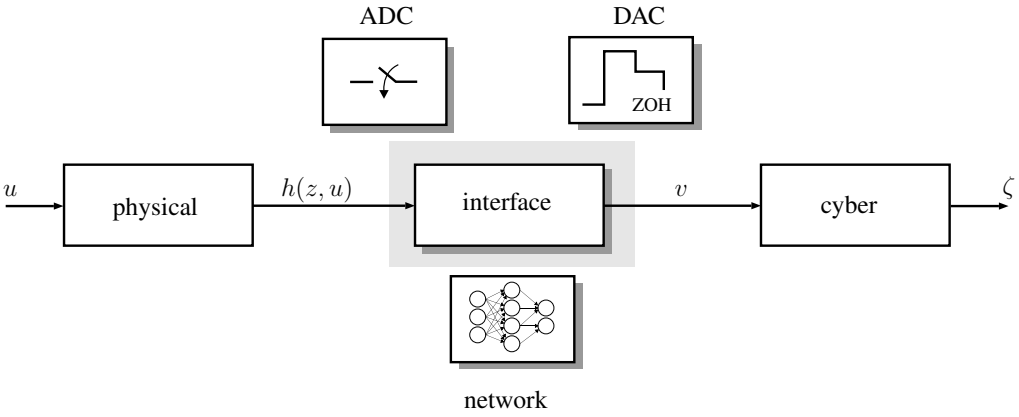
$$\lambda^+ \in G_I(\lambda, w) \qquad \text{when } (\lambda, w) \in D_I \tag{19}$$

$$\psi = \varphi(\lambda) \tag{20}$$

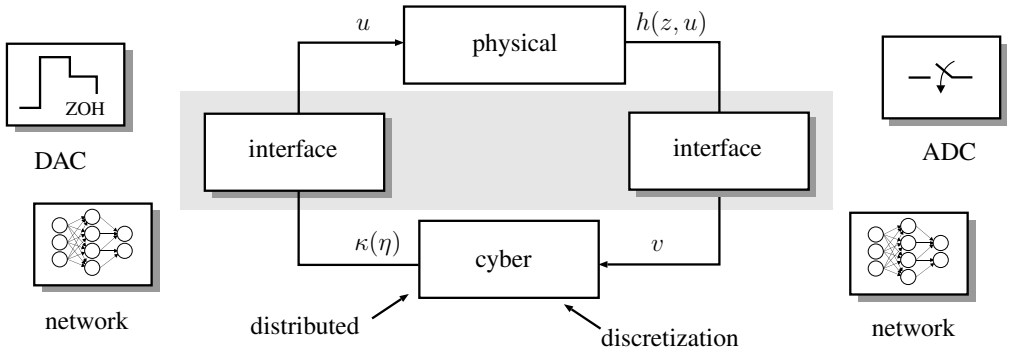
where F_I defines the continuous dynamics on C_I and G_I the discrete dynamics on D_I of the interface.

D. Combining models of physical and cyber components

Mathematical models of certain types of cyber-physical systems can be obtained by appropriately interconnecting the models of physical and cyber components given in Section II-A and Section II-B, respectively, and of the interfaces in Section II-C. Figure 2 depicts two specific interconnections of such models, a cascade interconnection in Figure 2(a) and a feedback interconnection in Figure 2(b). The individual models of the components are used to define a model of the entire cyber-physical system, which, due to the combination of differential and difference equations/inclusions, is in the hybrid inclusion form.



(a) Series topology



(b) Feedback topology

Fig. 2. **Cyber-physical systems:** series and parallel interconnections between a plant (part of the physical component), controller (part of the cyber component), and interfaces/converters/signal conditioners (part of both the physical and cyber components).

The following examples exercise the models proposed in the previous section for modeling specific cyber-physical systems.

Example 2.8 (Implementation of a FSM): Transitions of the state of a FSM are triggered by changes of its input v. When the input to the FSM is an external signal, a digital implementation of a FSM would require a conversion of the input, from analog to digital. Due to the conversion occurring at isolated time instants t_i, state transitions of the FSM will only take place at those instants. The resulting system can be modeled as a cascade of two systems, in which an analog-

to-digital converter drives the FSM. The analog-to-digital converter is modeled as the hybrid inclusion model (12)-(13) and the FSM as the difference equation in (6). Assuming that the input of the converter is transferred to its output at every conversion event of the converter, and that the output of the converter ψ feeds the input v of the FSM, we do not need to include a memory state m_s in the converter. In this way, when the timer τ_s is within $[0, T_s^*)$, the timer in the converter evolves according to

$$\dot{\tau}_s = 1$$

and the FSM variable q remains constant, which can be written as the trivial differential equation

$$\dot{q} = 0$$

When the timer τ_s reaches T_s^*, then the timer is updated via the difference equation

$$\tau_s^+ = 0$$

and the state of the FSM is updated according to the transition function evaluated at the current input, that is, q is updated according to

$$q^+ = \delta(q, w)$$

The above model can be summarized as follows:

$$\left.\begin{array}{rcl} \dot{\tau}_s & = & 1 \\ \dot{q} & = & 0 \end{array}\right\} \quad \text{when} \quad (q, v) \in Q \times \Sigma, \tau_s \in [0, T_s^*]$$

$$\left.\begin{array}{rcl} \tau_s^+ & = & 0 \\ q^+ & = & \delta(q, w) \end{array}\right\} \quad \text{when} \quad (q, v) \in Q \times \Sigma, \tau_s \geq T_s^*$$

△

Example 2.9 (Estimation Over a Network): Consider a physical process given in terms of the state-space model

$$\dot{z} = Az, \qquad y = Cz, \qquad z \in \mathbb{R}^{n_P}, y \in \mathbb{R}^{r_P} \tag{21}$$

where z is the state and y is the measured output. The output is digitally transmitted through a network. At the other end of the network, a computer receives the information and runs an algorithm that takes the measurements of y to estimate the state z of the physical process. We consider an estimation algorithm with a state variable $\hat{z} \in \mathbb{R}^{n_P}$, which denotes the estimate of z, that is appropriately reset to a new value involving the information received. More precisely, denoting the transmission times by t_i and L a constant matrix to be designed, the estimation algorithm updates its state as follows

$$\hat{z}^+ = \hat{z} + L(y - C\hat{z}) \tag{22}$$

at every instant information is received. In between such events, the algorithm updates its state continuously so as to match the evolution of the state of the physical process, that is, via

$$\dot{\hat{z}} = A\hat{z} \tag{23}$$

Modeling the network as in Section II-C.3, which, in particular, assumes zero transmission delay, the state variables of the entire system are z, $\tau_N \in \mathbb{R}$, $m_N \in \mathbb{R}^{r_P}$, and $\hat{z}$. Then, transmissions occur when $\tau_N \leq 0$, at which events the state of the network is updated via

$$\tau_N^+ \in [T_N^{*\min}, T_N^{*\max}], \qquad m_N^+ = y$$

and the state of the algorithm is updated via (22). Note that since the state of the physical process does not change at such events, we can use the following trivial difference equation to update it at the network events:

$$z^+ = z$$

In between events, the state of the network is updated as

$$\dot{\tau}_N = -1, \qquad \dot{m}_N = 0$$

the state of the algorithm changes continuously according to (23), and the state of the physical process changes according to (21). Combining the equations above, we arrive to the following model of the system:

$$\left.\begin{array}{rcl} \dot{z} & = & Az \\ \dot{\tau}_N & = & -1 \\ \dot{m}_N & = & 0 \\ \dot{\hat{z}} & = & A\hat{z} \end{array}\right\} \quad \text{when} \quad z \in \mathbb{R}^{n_P},\ \tau_N \in [0, T_N^{*\max}],\ m_N \in \mathbb{R}^{r_P},\ \hat{z} \in \mathbb{R}^{n_P}$$

$$\left.\begin{array}{rcl} z^+ & = & z \\ \tau_N^+ & \in & [T_N^{*\min}, T_N^{*\max}] \\ m_N^+ & = & y \\ \hat{z}^+ & = & \hat{z} + L(y - C\hat{z}) \end{array}\right\} \quad \text{when} \quad z \in \mathbb{R}^{n_P},\ \tau_N \leq 0,\ m_N \in \mathbb{R}^{r_P},\ \hat{z} \in \mathbb{R}^{n_P}$$

As suggested above, a digital implementation of this law would lead to a discrete-time system as in Example 2.7, which when implemented in a computer would requite an additional timer and a memory state.

Also, note that the algorithm uses the current measurement of y instead of the latest value held in m_N. This is due to assuming that there is zero transmission delay in the network. $\triangle$

Example 2.10 (Sample-and-hold Feedback Control): Consider a physical process modeled as in Example 2.1 and a control algorithm given in terms of a finite state machine modeled as in (6). The algorithm uses measurements of its output and controls the input of the physical process with the goal of steering its state to zero. Suppose the sampling device is ideal and that the signals produced by the finite state machine are connected to the plant via a DAC modeled as in (14)-(15).

The interconnection between the models of the physical process, the sampling device, the finite state machine, and the DAC has the feedback topology shown in Figure 2(b). In particular, the output m_h of the DAC is connected to the input u of the physical process, while the input v of the finite state machine is equal to the output y of the physical process at every sampling instant. The resulting interconnected system leads to the following hybrid inclusion model:

$$\left.\begin{array}{rcl} \dot{z} & = & Az + Bm_h \\ \dot{\tau}_s & = & 1 \\ \dot{q} & = & 0 \\ \dot{\tau}_h & = & 1 \\ \dot{m}_h & = & 0 \end{array}\right\} \quad \text{when} \quad z \in \mathbb{R}^{n_P},\ \tau_s \in [0, T_s^*],\ q \in Q,\ \tau_h \in [0, T_h^*],\ m_N \in \mathbb{R}^{r_P}$$

$$\left.\begin{array}{rcl} z^+ & = & z \\ \tau_s^+ & = & 0 \\ q^+ & = & \delta(q, y) \\ \tau_h^+ & = & \tau_h \\ m_h^+ & = & m_h \end{array}\right\} \quad \text{when} \quad z \in \mathbb{R}^{n_P},\ \tau_s \geq T_s^*,\ q \in Q,\ \tau_h \in [0, T_h^*),\ m_N \in \mathbb{R}^{r_P}$$

$$\left.\begin{array}{rcl} z^+ &=& z \\ \tau_s^+ &=& \tau_s \\ q^+ &=& q \\ \tau_h^+ &=& 0 \\ m_h^+ &=& \kappa(q) \end{array}\right\} \quad \text{when} \quad z \in \mathbb{R}^{n_P},\ \tau_s \in [0, T_s^*),\ q \in Q,\ \tau_h \geq T_h^*,\ m_N \in \mathbb{R}^{r_P}$$

$$\left.\begin{array}{rcl} z^+ &=& z \\ \tau_s^+ &=& 0 \\ q^+ &=& \delta(q, v) \\ \tau_h^+ &=& 0 \\ m_h^+ &=& \kappa(q) \end{array}\right\} \quad \text{when} \quad z \in \mathbb{R}^{n_P},\ \tau_s \geq T_s^*,\ q \in Q,\ \tau_h \geq T_h^*,\ m_N \in \mathbb{R}^{r_P}$$

Note that in this model the events can be triggered by three different conditions which are parsed independently: expiration of the timer τ_s used for sampling only, expiration of the timer τ_h used in the DAC only, and expiration of both timers (this list of events is in the same order as the difference equations in the model above).

There are numerous practical examples of systems that can be modeled within the general model for sample-and-hold feedback control defined above. For example, one "classical" example is the control of the temperature of a room by turning on and off a heater so as to keep the temperature within a desired range; see the model in (3). Another widely known example is the control of the level of a water tank. $\triangle$

In general, the models of the cyber-physical systems given above can be written as a hybrid inclusion with state $x = (z, \eta, \lambda)$, input $\gamma = (u, v, w)$, and dynamics

$$\dot{x} \in F(x, \gamma) \qquad (x, \gamma) \in C \tag{24}$$

$$x^+ \in G(x, \gamma) \qquad (x, \gamma) \in D \tag{25}$$

with continuous evolution according to F over C and discrete evolution according to G over D. The map F is the *flow map*, the set C is the *flow set*, the map G is the *jump map*, and the set D is the *jump set*. In fact, the model of the system in Example 2.10 results in a hybrid inclusion as in (18)-(19) with state

$$x = (z, \tau_s, q, \tau_h, m_h) \in \mathcal{X} = \mathbb{R}^{n_P} \times [0, T_s^*] \times Q \times [0, T_h^*] \times \mathbb{R}^{r_P}$$

with all of the inputs of its subsystems assigned (hence, we can omit γ), and with dynamics

$$\dot{x} = F(x) = \begin{bmatrix} Az + Bm_h \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad x \in C \tag{26}$$

$$x^+ \in G(x) = \begin{cases} G_s(x) & \text{if } x \in D_1 \setminus D_2 \\ G_h(x) & \text{if } x \in D_2 \setminus D_2 \\ G_{sh}(x) & \text{if } x \in D_1 \cap D_2 \end{cases} \qquad x \in D = D_1 \cup D_2 \tag{27}$$

where

$$D_1 = \{x \in \mathcal{X} : \tau_s = T_s^*\}, \quad D_2 = \{x \in \mathcal{X} : \tau_h = T_h^*\}$$

and, for each $x \in D$,

$$G_s(x) = \begin{bmatrix} z \\ 0 \\ \delta(q,y) \\ \tau_h \\ m_h \end{bmatrix}, \qquad G_h(x) = \begin{bmatrix} z \\ \tau_s \\ q \\ 0 \\ \kappa(q) \end{bmatrix}, \qquad G_{sh}(x) = \begin{bmatrix} z \\ 0 \\ \delta(q,y) \\ 0 \\ \kappa(q) \end{bmatrix}$$

The next section presents control theoretical tools for the analysis and design of cyber-physical systems modeled as in (24)-(25). The tools permit to study properties of interest for cyber-physical systems, such as asymptotic stability and robustness, which are usually the focus in control theory, as well as set invariants, temporal logic, and simulation.

III. Control theoretical tools for analysis and design of CPS

A. Basic concepts

The systems presented in the previous examples suggest that the physical components may have variables that change continuously as ordinary (or physical) time evolves. Some of the examples also indicate that the cyber components may include variables that only change at discrete events. As a consequence, a notion of time parameterizing the evolution of the variables of a cyber-physical system has to keep track of both continuously and discretely changing variables. It is very tempting to define a notion of time that parameterizes the evolution of the variables by a collection of ordinary time intervals in which the upper boundary value of (potentially, the closure of) each interval corresponds to an instant at which an event occurs. However, such a notion arbitrarily prioritizes the continuous evolution of the variables, does not explicitly determine the number of events after a specific amount of continuous evolution, and, to avoid ambiguity at each time instant, forces to use combinations of open to the right (or open to the left) interval as well as left-continuous (or right-continuous, respectively) functions defining solutions. These disadvantages and the implications on robustness of such a notion of time were discussed in detail in [9].

In this article, the evolution of the variables of cyber-physical systems are conveniently parameterized by ordinary time

$$t \in [0, \infty) (= \mathbb{R}_{\geq 0})$$

which is incremented continuously as continuous evolution of the variables occurs, and a counter

$$j \in \{0, 1, 2, \ldots\} (= \mathbb{N})$$

which is incremented at unitary steps when events occur. Then, pairs (t, j) define a hybrid time instant, which takes values on subsets of $\mathbb{R}_{\geq 0} \times \mathbb{N}$ called *hybrid time domains*. A set

$$E \subset \mathbb{R}_{\geq 0} \times \mathbb{N}$$

is a hybrid time domain if, for each $(T, J) \in E$, the set

$$E \ \cap \ ([0, T] \times \{0, 1, \ldots, J\})$$

can be written in the form

$$\bigcup_{j=0}^{J} ([t_j, t_{j+1}] \times \{j\})$$

for some finite sequence of times $0 = t_0 \leq t_1 \leq t_2 \ldots \leq t_{J+1}$. The set

$$E \ \cap \ ([0,T] \times \{0,1,\ldots,J\})$$

defines a compact hybrid time domain since it is bounded and closed. Then, solutions to a hybrid inclusion are given by functions $(t,j) \mapsto (\phi(t,j), \gamma(t,j))$ defined on hybrid time domains that, over intervals of flow, satisfy

$$\frac{d}{dt}\phi(t,j) \in F(\phi(t,j), \gamma(t,j))$$
$$(\phi(t,j), \gamma(t,j)) \in C$$

and, at jump times,

$$\phi(t,j+1) \in G(\phi(t,j), \gamma(t,j))$$
$$(\phi(t,j), \gamma(t,j)) \in D$$

These functions are such that ϕ is a hybrid arc and γ is a hybrid input. A hybrid arc ϕ is a function with domain $\operatorname{dom}\phi$, where $\operatorname{dom}\phi$ is a hybrid time domain, and that, for each fixed j, the function $t \mapsto \phi(t,j)$ has a derivative, at least for almost every point in $I_j = \{t : (t,j) \in \operatorname{dom}(\phi,\gamma)\}$. More specifically, a hybrid arc ϕ is such that, for each $j \in \mathbb{N}$, $t \mapsto \phi(t,j)$ is absolutely continuous on intervals of flow I_j with nonzero Lebesgue measure. Similarly, a hybrid input γ is a function on a hybrid time domain that, for each $j \in \mathbb{N}$, $t \mapsto \gamma(t,j)$ is Lebesgue measurable and locally essentially bounded on I_j.

The conditions determining whether a hybrid arc and a hybrid input define a solution to a hybrid inclusion as well as when it can flow or jump are determined by the data of the system, namely, (C,F,D,G). A solution to a hybrid system $\mathcal{H}$ is given by a pair (ϕ,u) with $\operatorname{dom}\phi = \operatorname{dom} u (= \operatorname{dom}(\phi,u))$ and satisfying the dynamics of the hybrid inclusion, where ϕ is a hybrid arc and u is a hybrid input. More precisely, a hybrid input $\gamma : \operatorname{dom}\gamma \to \mathbb{R}^m$ and a hybrid arc $\phi : \operatorname{dom}\phi \to \mathbb{R}^n$ define a *solution pair* (ϕ,γ) *to the hybrid inclusion* (24)-(25) if the following conditions hold:

(S0) $(\phi(0,0), \gamma(0,0)) \in \overline{C} \cup D$ and $\operatorname{dom}\phi = \operatorname{dom}\gamma$;

(S1) For each $j \in \mathbb{N}$ such that $I_j = \{t : (t,j) \in \operatorname{dom}(\phi,\gamma)\}$ has nonempty interior $\operatorname{int}(I_j)$, we have

$$(\phi(t,j), \gamma(t,j)) \in C \text{ for all } t \in \operatorname{int}(I_j)$$

and, for almost all $t \in I_j$, we have

$$\frac{d}{dt}\phi(t,j) \in F(\phi(t,j), \gamma(t,j))$$

(S2) For each $(t,j) \in \operatorname{dom}(\phi,\gamma)$ such that $(t,j+1) \in \operatorname{dom}(\phi,\gamma)$, we have

$$(\phi(t,j), \gamma(t,j)) \in D$$

and

$$\phi(t,j+1) \in G(\phi(t,j), \gamma(t,j))$$

Solutions can be classified in terms of their hybrid time domains. A solution (ϕ,γ) to the hybrid inclusion (24)-(25) is said to be

- *nontrivial* if $\mathrm{dom}(\phi, \gamma)$ has at least two points;
- *continuous* if $\mathrm{dom}(\phi, \gamma) \subset \mathbb{R}_{\geq 0} \times \{0\}$;
- *discrete* if $\mathrm{dom}(\phi, \gamma) \subset \{0\} \times \mathbb{N}$;
- *complete* if $\mathrm{dom}(\phi, \gamma)$ is unbounded;
- *Zeno* if it is complete and the projection of $\mathrm{dom}(\phi, \gamma)$ onto $\mathbb{R}_{\geq 0}$ is bounded;
- *dwell time* if there exists a positive constant c such that, for each $j > 0$ such that $(t, j) \in \mathrm{dom}\, \phi$ for some t, there exists $(t', j), (t'', j) \in \mathrm{dom}\, \phi$ with $t'' - t' \geq c$;
- *maximal* if there does not exist another pair $(\phi, \gamma)'$ such that (ϕ, γ) is a truncation of $(\phi, \gamma)'$ to some proper subset of $\mathrm{dom}(\phi, \gamma)'$.

For more details about solutions to hybrid inclusions as in (24)-(25), see [10].

Example 3.1: The FSM model in Section II-B.1 is a purely discrete system, which can be modeled as a hybrid inclusion with state $x = q$, input $\gamma = v$, and data

$$C = \emptyset, \qquad F = \star, \qquad D = Q \times \Sigma, \qquad G(x, \gamma) = \delta(x, \gamma)$$

where $\star$ indicates an arbitrary choice. Note that while the model in Section II-B.1 does not involve a notion of time, this hybrid inclusion model confers to it the hybrid time notion. Since δ is a function that maps from D into q, we have that after every jump, $G(x, \gamma)$ maps to points into q. Then, since v takes values from Σ, for every input γ we have that every solution (ϕ, γ) to the hybrid inclusion is discrete. Furthermore, every maximal solution is complete. $\triangle$

Example 3.1 motivates the search for conditions under which solutions to a general hybrid inclusion exist and fall under the particular types listed above. Such conditions necessarily involve the data of the system. The conditions given below, most of which are given in [1, Proposition 2.10 and Proposition 6.10], provide insight on the existence and the type of solutions for such systems. They apply to hybrid inclusions without input, in which case C and D are subsets of $\mathbb{R}^n$ and $F, G : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$.

1) Let $\xi \in \overline{C} \cup D$. If $\xi \in D$ or

   (VC) there exist $\varepsilon > 0$ and an absolutely continuous function $z : [0, \varepsilon] \to \mathbb{R}^n$ such that $z(0) = \xi, \dot{z}(t) \in F(z(t))$ for almost all $t \in [0, \varepsilon]$ and $z(t) \in C$ for all $t \in (0, \varepsilon]$,

   then there exists a nontrivial solution ϕ to the hybrid inclusion (24)-(25) with $\phi(0, 0) = \xi$.
2) If (VC) holds for every $\xi \in \overline{C} \setminus D$, then there exists a nontrivial solution to the hybrid inclusion (24)-(25) from every point of $\overline{C} \cup D$.
3) If C is nonempty and D is empty, then every solution is continuous.
4) If C is empty and D is nonempty, then every solution is discrete. Furthermore, if $G(D) \subset D$ then every maximal solution is complete.
5) If (VC) holds for every $\xi \in \overline{C} \setminus D$, then every maximal solution ϕ satisfies exactly one of the following:
   a) ϕ is complete;
   b) $\mathrm{dom}\, \phi$ is not complete and "ends with flow", with $(T, J) = \sup \mathrm{dom}\, \phi$, the interval I_J has nonempty interior; and either
      - I_J is closed, in which case $\phi(T, J) \in \overline{C} \setminus (C \cap D)$; or
      - I_J is open to the right, in which case $\phi(T, J) \notin \mathrm{dom}\, \phi$, and there does not exist an absolutely continuous function $z : \overline{I_J} \to \mathbb{R}^n$ satisfying $\dot{z}(t) \in F(z(t))$ for almost all $t \in I_J, z(t) \in C$ for all $t \in \mathrm{int}\, I_J$, and such that $z(t) = \phi(t, J)$ for all $t \in I_J$;
   c) $\mathrm{dom}\, \phi$ is not complete and "ends with a jump": for $(T, J) = \sup \mathrm{dom}\, \phi$, one has $\phi(T, J) \notin \overline{C} \cup D$.

Furthermore, if $G(D) \subset \overline{C} \cup D$, then 5(c) above does not occur. Moreover, if C is closed and F is a continuous single-valued map, then (VC) above is implied by the following property: for the given $\xi \in \overline{C} \setminus D$, there exists a neighborhood U of ξ such that

$$F(x) \cap T_C(x) \neq \emptyset \qquad \forall x \in U \cap C$$

where $T_C(x)$ is the tangent cone of C at x.

Example 3.2: A hybrid inclusion model for the sample-and-hold feedback control system in Example 2.10 is given in (26)-(27). Its state is given by $x = (z, \tau_s, q, \tau_h, m_h)$ and the sets C and D are closed. For each $x = (z, \tau_s, q, \tau_h, m_h) \in D$, we have that either $\tau_s = T_s^*$ or $\tau_h = T_h^*$. If only $\tau_s = T_s^*$, then $G(x) = G_s(x)$ which belongs to $C \setminus D$. If only $\tau_h = T_h^*$, then $G(x) = G_h(x)$ which also belongs to $C \setminus D$. Similarly, when both $\tau_s = T_s^*$ and $\tau_h = T_h^*$, G maps the state to a point in $C \setminus D$. Then, $G(D) \subset C \cup D$. For each $x \in C \setminus D$, we have that $\tau_s < T_s^*$ and $\tau_h < T_h^*$. Then, there exists $\varepsilon > 0$ and a solution to $\dot{x} = F(x)$ such that, over $[0, \varepsilon]$, the solution stays within C – this implies that (VC) holds. Item 2 above implies that there exists a nontrivial solution from every point in $C \cup D$. Furthermore, since $G(D) \subset C \cup D$, item 5(c) does not hold and due to the properties F, item 5(b) does not hold either. Then, item 5(a) implies that every maximal solution is complete. $\triangle$

B. Invariants

An *invariant* for a system is a set with the property that every solution that starts from it stays in it for all future time. The knowledge of the invariants of a system is key in its design since a wide range of design specifications require its variables to remain within a region. Characterizing the invariants is of paramount importance in the study of reachability, safety, and stability.

A definition of invariant for a hybrid inclusion should account for the fact that solutions may not be unique and that every maximal solution may not be complete. Nonuniqueness of solutions may arise due to F or G being set valued, or due to C and D overlapping. In particular, as pointed out in Section II-B.1, nondeterministic finite state machines lead to set-valued jump maps G. A set-valued jump map was also used in the model of a digital network in Section II-C.3. The sets C and D may overlap when both flows and jumps should be allowed. Such nondeterminism might be desired to model the behavior of a system or when, due to perturbations, it is not possible to determine whether the perturbed state is in C or D, which is typically the case nearby the boundaries of these sets. Noncompleteness of maximal solutions may be due to, in particular, the flow map F steering the state to points in the boundary of $C \setminus D$ from where flow within C is not possible, or due to the jump map G mapping the state to points away from $\overline{C} \cup D$.

Due to the nonuniqueness of solutions, notions of weak and strong invariants can be introduced for hybrid inclusions. Due to the noncompleteness of maximal solutions, notions of invariants that do and do not impose that every maximal solution is complete are plausible. In this article, we consider a strong notion that does not insist on every maximal solution being complete. More precisely, a set $K \subset \mathbb{R}^n$ is said to be an invariant for a hybrid inclusion if, for each $\phi_0 \in K$, each solution (ϕ, γ) with $\phi(0,0) = \phi_0$ satisfies

$$\phi(t, j) \in K \qquad \forall (t, j) \in \mathrm{dom}(\phi, \gamma)$$

In other words, over its hybrid time domain, each solution with component ϕ starting in K has to stay in K for any possible input γ. Note that the notion does not insist on solutions to even exist from K.

In general, it is very difficult to directly check that a set is an invariant from the definition above, as that would require to explicitly check each solution from the set under study. This motivates the development of solution-independent sufficient conditions for invariants.

The tangent cone to a set will be used in the conditions below. The tangent cone to a set $K \subset \mathbb{R}^n$ at a point $x \in \mathbb{R}^n$, denoted as $T_K(x)$, is the set of all vectors $\omega \in \mathbb{R}^n$ for which there exist sequences $x_i \in K$, $\tau_i > 0$ with $x_i \to x$, $\tau_i \searrow 0$ and

$$\omega = \lim_{i \to \infty} \frac{x_i - x}{\tau_i}.$$

See, e.g., [1, Definition 5.12]. We will also assume the following. The sets K and C are such that $(K \times \mathbb{R}^m) \cap C$ is closed. The map $F : \mathbb{R}^n \times \mathbb{R}^m \rightrightarrows \mathbb{R}^n$ is outer semicontinuous, locally bounded relative to $C, C \subset \operatorname{dom} F$, and $F(x, \gamma)$ is convex for every $(x, \gamma) \in C$. Then, the set K is an invariant if

1) $G((K \times \mathbb{R}^m) \cap D) \subset K$; and
2) For every $\xi \in (K \times \mathbb{R}^m) \cap C$, there exists a neighborhood U of ξ such that

$$F(x, \gamma) \subset T_{(K \times \mathbb{R}^m) \cap C}(x, \gamma)$$

for every $(x, \gamma) \in U$.

Example 3.3: Consider the linear-time invariant model of the temperature of a room in (3) and the finite state machine given in Figure 3, which can be modeled as in (6) with

$$Q = \{\text{ON}, \text{OFF}\}, \qquad \Sigma = \mathbb{R}$$

and, for each $(q, v) \in Q \times \Sigma$,

$$\delta(q, v) = \begin{cases} \text{ON} & \text{if } v \leq z_{\min} \text{ and } q = \text{OFF} \\ \text{OFF} & \text{if } v \geq z_{\max} \text{ and } q = \text{ON} \end{cases}, \qquad \kappa(q) = \begin{cases} 1 & \text{if } q = \text{ON} \\ 0 & \text{if } q = \text{OFF} \end{cases}$$

where $z_{\min} < z_{\max}$.

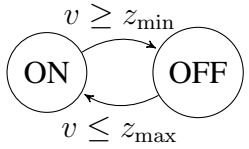


Fig. 3. A finite state machine to control the temperature of a room.

When $u_2 = z_{\text{out}}$ with $z_{\text{out}} \in (-\infty, z_{\max}]$ and $z_{\text{out}} + z_\Delta \in [z_{\min}, \infty)$, it can be shown that when an ideal implementation of this finite state machine is used to control the temperature of the room via the input u_1, every solution to the closed-loop system is such that z reaches the range $[z_{\min}, z_{\max}]$ in finite time and stays there for all future time. The closed-loop system is modeled as a hybrid inclusion with state $x = (z, q)$ and data

$$\begin{aligned} C &= \{x \in \mathbb{R} \times Q : z \geq z_{\min}, q = \text{OFF}\} \cup \{x \in \mathbb{R} \times Q : z \leq z_{\max}, q = \text{ON}\} \\ F(x) &= \begin{bmatrix} -z + \begin{bmatrix} z_\Delta & 1 \end{bmatrix} \begin{bmatrix} q \\ z_{\text{out}} \end{bmatrix} \\ 0 \end{bmatrix} \quad \forall x \in C \\ D &= \{x \in \mathbb{R} \times Q : z \leq z_{\min}, q = \text{OFF}\} \cup \{x \in \mathbb{R} \times Q : z \geq z_{\max}, q = \text{ON}\} \\ G(x) &= \begin{bmatrix} z \\ \delta(q, z) \end{bmatrix} \qquad \forall x \in D \end{aligned}$$

The latter property can be established by checking that the conditions proposed above for $K = [z_{\min}, z_{\max}] \times Q$ to be invariant hold. In fact, for each point $x \in K \cap D$ with $z \in [z_{\min}, z_{\max}]$, we have that after the jump, z is mapped back to $[z_{\min}, z_{\max}]$ and, in turn, x is mapped back to K. Furthermore, for every point in $\xi \in K \cap C$ the flow map F is such that the z component of the solutions from nearby ξ stay in $[z_{\min}, z_{\max}]$ for some small amount of time. $\triangle$

C. Stability and Attractivity

Stability is a notion of particular interest in the study of dynamical systems as it captures the property that solutions starting nearby a point (or region) stay nearby. This property is desired for the executions of a cyber-physical system as, in particular, it would guarantee that the evolution of the physical states and the variables in the code within the cyber components remain nearby steady-state values when initialized nearby. For models of such systems given in terms of hybrid inclusions as in (24)-(25) and without inputs, stability is defined as follows:

A closed set $\mathcal{A} \subset \mathbb{R}^n$ is said to be *Lyapunov stable* if for each $\varepsilon > 0$ there exists $\delta > 0$ such that each maximal solution ϕ to the hybrid inclusion in (24)-(25) without inputs and with

$$|\phi(0,0)|_{\mathcal{A}} \leq \delta$$

satisfies

$$|\phi(t,j)|_{\mathcal{A}} \leq \varepsilon$$

for all $(t,j) \in \operatorname{dom} \phi$.

The notion reduces to stability of a point when the set $\mathcal{A}$ is a singleton, and $|\cdot|_{\mathcal{A}}$ is the distance to the point $\mathcal{A}$. The general closed set $\mathcal{A}$ in the definition allows to study the stability of sets that have more than one point, in which case $|x|_{\mathcal{A}} = \inf_{y \in \mathcal{A}} |x - y|$. For example, the set of interest in Example 3.3 would be

$$\mathcal{A} = [z_{\min}, z_{\max}] \times Q$$

Stability of a set $\mathcal{A}$ can be certified using Lyapunov functions. A Lyapunov function is a function Lyapunov functions for a hybrid inclusion without inputs are given by functions $V : \operatorname{dom} V \to \mathbb{R}$ that are defined on $\operatorname{dom} V$ containing $\overline{C} \cup D \cup G(D)$ and that are continuously differentiable on an open set containing the closure of C. The following sufficient condition for stability of a closed set $\mathcal{A}$ can be established [1, Theorem 3.18]: suppose that V is positive definite with respect to $\mathcal{A}$ and such that

$$\langle \nabla V(x), f \rangle \leq 0 \qquad \forall x \in C,\ f \in F(x) \tag{28a}$$

$$V(g) - V(x) \leq 0 \qquad \forall x \in D,\ g \in G(x) \tag{28b}$$

then $\mathcal{A}$ is stable.

A property that is also of interest when studying dynamical systems is attractivity (or convergence). In particular, attractivity of a set $\mathcal{A}$ captures the property of complete solutions converging to $\mathcal{A}$. More precisely:

A set $\mathcal{A} \subset \mathbb{R}^n$ is said to be *attractive* if there exists $\mu > 0$ such that every maximal solution ϕ to the hybrid inclusion in (24)-(25) without inputs and with

$$|\phi(0,0)|_{\mathcal{A}} \leq \mu$$

is bounded and, if it is complete, satisfies

$$\lim_{(t,j)\in\operatorname{dom}\phi, t+j\to\infty} |\phi(t,j)|_{\mathcal{A}} = 0$$

Attractivity of a set can also be established using Lyapunov functions. The conditions on V plus a weak invariance property of a set can be used to characterize the set of points to which complete and bounded solutions converge to. Below, we say that a set $K \subset \mathbb{R}^n$ is *weakly forward invariant* if for each $\phi(0,0) \in K$, there exists at least one complete solution ϕ with $\phi(t,j) \in K$ for all $(t,j) \in \operatorname{dom}\phi$, *weakly backward invariant* if for each $x' \in K$, $N > 0$, there exist $\phi(0,0) \in K$ and at least one solution ϕ such that for some $(t^*, j^*) \in \operatorname{dom}\phi$, $t^* + j^* \geq N$, we have $\phi(t^*, j^*) = x'$ and $\phi(t,j) \in K$ for all $(t,j) \preceq (t^*, j^*)$, $(t,j) \in \operatorname{dom}\phi$, and *weakly invariant* if it is both weakly forward invariant and weakly backward invariant.

The invariance principle in [5] states that under the conditions on V above and further conditions on its data (see Section III-E), every complete and bounded solution to the hybrid inclusion converges to the largest weakly invariant set contained in the set of points

$$\left(\left\{x \in \mathbb{R}^n \,:\, \max_{f\in F(x)} \langle \nabla V(x), f\rangle = 0\right\} \bigcup \left\{x \in \mathbb{R}^n \,:\, \max_{g\in G(x)} V(g) - V(x) = 0\right\}\right) \bigcap \{x \in \mathbb{R}^n \,:\, V(x) = r\}$$

for some $r \geq 0$.

Alternatively, attractivity of a set $\mathcal{A}$ can be certified by strengthening the Lyapunov conditions above. Suppose that V is positive definite with respect to $\mathcal{A}$ and such that (28a)-(28b) hold, and

$$\langle \nabla V(x), f\rangle < 0 \qquad \forall x \in C \setminus \mathcal{A},\ f \in F(x) \tag{29a}$$

$$V(g) - V(x) < 0 \qquad \forall x \in D \setminus \mathcal{A},\ g \in G(x) \tag{29b}$$

then $\mathcal{A}$ is attractive. Note that these conditions also imply that $\mathcal{A}$ is stable, which in turn implies that $\mathcal{A}$ is asymptotically stable (both stable and attractive).

Example 3.4: The state estimation problem over a network in Example 2.9 can be solved by asymptotically stabilizing the set

$$\mathcal{A} = \left\{(z, \hat{z}) \in \mathbb{R}^{2n} \colon z = \hat{z}\right\} \tag{30}$$

As usual in estimation problems, we consider the estimation error defined as

$$\varepsilon := z - \hat{z}$$

In these coordinates, the system composed by the state ε and the timer variable τ_N can be represented by the following hybrid inclusion:

$$\left.\begin{array}{rcl} \dot{\varepsilon} &=& A\varepsilon \\ \dot{\tau}_N &=& -1 \end{array}\right\} \quad (\varepsilon, \tau_N) \in C$$

$$\left.\begin{array}{rcl} \varepsilon^+ &=& (I - LM)\varepsilon \\ \tau_N^+ &\in& [T_N^{*\min}, T_N^{*\max}] \end{array}\right\} \quad (\varepsilon, \tau_N) \in D \tag{31a}$$

with the flow set and the jump set defined as

$$\begin{aligned} C &= \left\{(\varepsilon,\tau)\in\mathbb{R}^{n+1}\colon \tau_N\in[0,T_N^{*\max}]\right\}\\ D &= \left\{(\varepsilon,\tau)\in\mathbb{R}^{n+1}\colon \tau_N=0\right\}. \end{aligned} \tag{31b}$$

It follows that if there exist a symmetric positive definite matrix P and a matrix L such that

$$(I-LM)^\top \exp(A^\top s)P\exp(As)(I-LM)-P<0 \qquad \forall s\in[T_N^{*\min},T_N^{*\max}], \tag{32}$$

then the set $\mathcal{A}$ defined in (30) is asymptotically stable for the hybrid inclusion (31); see [11] for more details. △

D. Temporal logic

Temporal logic allows for the definition of conditions that the executions to a system need to satisfy over time. The conditions are given in terms of a language that employs logical and temporal connectives (or operators) applied to propositions, which are functions of the state of the system and are used to define formulae. In particular, *linear temporal logic* (LTL) can be efficiently employed to determine *safety*, i.e., "something bad never happens," and *liveness*, i.e., "something good eventually happens." Verification of these properties can be performed using model checking tools. For example, safety can be ruled out by finding an execution that, in finite time, violates the assertion defining the safety property.

Following [12], the standard syntax of LTL language can be defined recursively as follows. An atomic proposition is a statement given in terms of the state x that, for each possible value of x, is either `True` (1 or $\top$) or `False` (0 or $\bot$). The following logical operators are defined:

- $\neg$ is the *negation* operator;
- $\vee$ is the *disjunction* operator;
- $\wedge$ is the *conjunction* operator;
- $\Rightarrow$ is the *implication* operator;
- $\Leftrightarrow$ is the *equivalence* operator.

In addition, temporal operators are defined as follows:

- $\bigcirc$ is the *next* operator;
- $\Box$ is the *always* operator;
- $\Diamond$ is the *eventually* operator;
- $\mathcal{U}$ is the *until* operator.

Informally speaking, the temporal operator *until* when used as $\mathfrak{f}_1\mathcal{U}\mathfrak{f}_2$ implies that $\mathfrak{f}_2$ will eventually become `True` and $\mathfrak{f}_1$ will keep being `True` until $\mathfrak{f}_2$ becomes `True`. In the case of the operator *next*, $\bigcirc\mathfrak{f}_1$ implies that $\mathfrak{f}_1$ will be `True` at the next time instance. Every atomic proposition is an LTL formula and if $\mathfrak{f}_1$ and $\mathfrak{f}_2$ are formulae then so are

$$\neg\mathfrak{f}_1, \qquad \mathfrak{f}_1\vee\mathfrak{f}_2, \qquad \bigcirc\mathfrak{f}_1, \qquad \mathfrak{f}_1\mathcal{U}\mathfrak{f}_2$$

Note that given the operators *negation* and *disjunction*, the operators *conjunction* ($\wedge$), *implication* ($\Rightarrow$), and *equivalency* ($\Leftrightarrow$) can be defined as

$$\mathfrak{f}_1\wedge\mathfrak{f}_2=\neg(\neg\mathfrak{f}_1\vee\neg\mathfrak{f}_2)$$

$$\mathfrak{f}_1\Rightarrow\mathfrak{f}_2=\neg\mathfrak{f}_1\vee\mathfrak{f}_2$$

$$\mathfrak{f}_1\Leftrightarrow\mathfrak{f}_2=(\mathfrak{f}_1\Rightarrow\mathfrak{f}_2)\wedge(\mathfrak{f}_2\Rightarrow\mathfrak{f}_1)$$

respectively. Note that the operators *eventually* ($\diamond$) and *always* ($\square$) can be defined as

$$\diamond \mathfrak{f}_1 = \texttt{True} \mathcal{U} \mathfrak{f}_1$$

and

$$\square \mathfrak{f}_1 = \neg \diamond \neg \mathfrak{f}_1$$

respectively.

With the above grammar we define a semantics for cyber-physical systems given in terms of hybrid inclusions, to which below we refer as $\mathcal{H}$. For simplicity, we consider the case of no inputs and state-dependent atomic propositions $\mathfrak{a}$. The value of a solution $(t, j) \mapsto \phi(t, j)$ assigning an atomic proposition $\mathfrak{a}$ the value `True` at time (t, j) is denoted

$$\phi(t, j) \Vdash \mathfrak{a}$$

while, if the value assigned is `False`, we denote it

$$\phi(t, j) \nVdash \mathfrak{a}$$

(equivalently, $\phi(t, j) \Vdash \mathfrak{a}$ can be written as $\mathfrak{a}(\phi(t, j)) = \texttt{True}$). Similarly, a formula $\mathfrak{f}_1$ being satisfied by a solution $(t, j) \mapsto \phi(t, j)$ at some time (t, j) is denoted by

$$(\phi, (t, j)) \vDash \mathfrak{f}_1$$

Now, we define a semantics for solutions to $\mathcal{H}$. Let $\mathfrak{a}$ be an atomic proposition, and $\mathfrak{f}_1$ and $\mathfrak{f}_2$ be two formulae given in terms of the LTL syntax described above. Given a solution $(t, j) \mapsto \phi(t, j)$ to $\mathcal{H}$, we define the following operators:

$$(\phi, (t, j)) \vDash \mathfrak{a} \quad \text{iff} \quad \phi(t, j) \Vdash \mathfrak{a} \tag{33}$$

$$(\phi, (t, j)) \vDash \neg \mathfrak{f}_1 \quad \text{iff} \quad (\phi, (t, j)) \nvDash \mathfrak{f}_1 \tag{34}$$

$$(\phi, (t, j)) \vDash \mathfrak{f}_1 \vee \mathfrak{f}_2 \quad \text{iff} \quad (\phi, (t, j)) \vDash \mathfrak{f}_1 \text{ or } (\phi, (t, j)) \vDash \mathfrak{f}_2 \tag{35}$$

$$(\phi, (t, j)) \vDash \bigcirc \mathfrak{f}_1 \quad \text{iff} \quad (\phi, (t, j + 1)) \vDash \mathfrak{f}_1 \tag{36}$$

$$\begin{aligned}(\phi, (t, j)) \vDash \mathfrak{f}_1 \mathcal{U} \mathfrak{f}_2 \quad \text{iff} \quad & \exists (t', j') \in \operatorname{dom} \phi, t' + j' \geq t + j \text{ such that } (\phi, (t', j')) \vDash \mathfrak{f}_2, \\ & \text{and for all } (t'', j'') \in \operatorname{dom} \phi \text{ s.t. } t + j \leq t'' + j'' < t' + j', (\phi, (t'', j'')) \vDash \mathfrak{f}_1\end{aligned} \tag{37}$$

Even though the above completely define the semantics, the following operators are used for convenience. Let $\mathfrak{a}_1$ and $\mathfrak{a}_2$ be two atomic propositions. Given a solution $(t, j) \mapsto \phi(t, j)$ to $\mathcal{H}$, we have the following additional operators:

$$(\phi, (t, j)) \vDash \mathfrak{a}_1 \wedge \mathfrak{a}_2 \quad \text{iff} \quad \phi(t, j) \Vdash \mathfrak{a}_1 \text{ and } \phi(t, j) \Vdash \mathfrak{a}_2; \tag{38}$$

$$(\phi, (t, j)) \vDash \square \mathfrak{f}_1 \quad \text{iff} \quad (\phi, (t', j')) \vDash \mathfrak{f}_1 \text{ for all } t' + j' \geq t + j, (t', j') \in \operatorname{dom} \phi; \tag{39}$$

$$(\phi, (t, j)) \vDash \diamond \mathfrak{a}_1 \quad \text{iff} \quad \exists (t', j') \in \operatorname{dom} \phi, t' + j' \geq t + j \text{ such that } (\phi, (t', j')) \vDash \mathfrak{a}_1. \tag{40}$$

We say that a solution $(t, j) \mapsto \phi(t, j)$ satisfies the formula $\mathfrak{f}_1$ if and only if $(\phi, 0) \vDash \mathfrak{f}_1$. With this semantics, we propose methods to check whether a given formula is true or not at a particular time (t, j), and more generally, over the whole domain domain of definition, namely, whether a solution satisfies a formula.

A solution $(t, j) \mapsto \phi(t, j)$ to $\mathcal{H}$ satisfies the formula

$$\mathfrak{f}_1 = \diamond \mathfrak{a}_1$$

if there exists N^* such that for each $t+j \geq N^*$, $(t,j) \in \operatorname{dom}\phi$ we have that $\phi(t,j)$ satisfies $\mathfrak{a}_1$. Given $\mathfrak{a}_1$ and the data (C,F,D,G) of $\mathcal{H}$, define the system $\mathcal{H}_{\cap\neg\mathfrak{a}_1}$ as the system with data $(C_{\cap\neg\mathfrak{a}_1}, F, D_{\cap\neg\mathfrak{a}_1}, G)$, where

$$C_{\cap\neg\mathfrak{a}_1} = C \cap \{x \in \mathbb{R}^n \,:\, \mathfrak{a}_1(x) = \texttt{False}\}, \qquad D_{\cap\neg\mathfrak{a}_1} = D \cap \{x \in \mathbb{R}^n \,:\, \mathfrak{a}_1(x) = \texttt{False}\}$$

Let $\phi_0 = \phi(0,0)$. The formula $\mathfrak{f}_1 = \Diamond\mathfrak{a}_1$ is true for every maximal solution from ϕ_0 when the following two properties hold:

- Every maximal solution to $\mathcal{H}$ from ϕ_0 is complete;
- Every maximal solution to $\mathcal{H}_{\cap\neg\mathfrak{a}_1}$ from ϕ_0 is not complete.

In fact, since every maximal solution from ϕ_0 is complete, maximal solutions from ϕ_0 to $\mathcal{H}_{\cap\neg\mathfrak{a}_1}$ would not be complete if there exists (t',j') from which flow within $C_{\cap\neg\mathfrak{a}_1}$ or jumps back to $C_{\cap\neg\mathfrak{a}_1} \cup D_{\cap\neg\mathfrak{a}_1}$ are not possible, which corresponds to the formula eventually being satisfied. (Note that a complete solution ϕ to the system $\mathcal{H}_{\cap\neg\mathfrak{a}_1}$ would satisfy $\phi(t,j) \in C_{\cap\neg\mathfrak{a}_1} \cup D_{\cap\neg\mathfrak{a}_1}$ for all $(t,j) \in \operatorname{dom}\phi$, which, by construction of $C_{\cap\neg\mathfrak{a}_1}$ and $D_{\cap\neg\mathfrak{a}_1}$, implies that $(\phi,(t,j)) \not\models \mathfrak{a}_1$ for all $(t,j) \in \operatorname{dom}\phi$.) Hence, under the conditions above, $\mathfrak{f}_1 = \Diamond\mathfrak{a}_1$ is not satisfied for any solution to $\mathcal{H}$ from ϕ_0.

According to the definition of the $\Box$ operator, a solution $(t,j) \mapsto \phi(t,j)$ to $\mathcal{H}$ satisfies the formula

$$\mathfrak{f}_1 = \Box\mathfrak{a}_1$$

if for each $(t,j) \in \operatorname{dom}\phi$ we have that $\phi(t,j)$ satisfies $\mathfrak{a}_1$. Along similar lines for the $\Diamond$ operator, given $\mathfrak{a}_1$ and the data (C,F,D,G) of $\mathcal{H}$, consider the system $\mathcal{H}_{\cap\neg\mathfrak{a}_1}$ as the system with data $(C_{\cap\neg\mathfrak{a}_1}, F, D_{\cap\neg\mathfrak{a}_1}, G)$ given above. The formulae $\mathfrak{f}_1 = \Box\mathfrak{a}_1$ is true for every solution to $\mathcal{H}$ when the following properties hold:

- $D \cap D_{\cap\neg\mathfrak{a}_1} = \emptyset$;
- $G(D) \cap D_{\cap\neg\mathfrak{a}_1} = \emptyset$;
- $\overline{C} \cap \overline{C_{\cap\neg\mathfrak{a}_1}} = \emptyset$.

If $D \cap D_{\cap\neg\mathfrak{a}_1}$ is nonempty, then there exists a point in D for which $\mathfrak{a}_1$ is not true, which would violate the satisfaction $\Box\mathfrak{a}_1$. Similarly, if there is a point in D from where, after a jump, the state is mapped to $D_{\cap\neg\mathfrak{a}_1}$ (which is possible when $G(D) \cap D_{\cap\neg\mathfrak{a}_1} = \emptyset$), then $\Box\mathfrak{a}_1$ will not be satisfied. Finally, the last condition rules out the possibility of solutions to $\mathcal{H}$ flowing to a boundary point of C that is not in C and at which $\mathfrak{a}_1$ is not true.

Regarding satisfaction of general formulae, the satisfaction of the formula $\mathfrak{f}_1$ by a solution ϕ to $\mathcal{H}$ from ϕ_0 is implied by the following property of maximal solutions:

$$\mathcal{S}_{\mathcal{H}_{\cap\mathfrak{f}_1}}(\phi_0) = \mathcal{S}_{\mathcal{H}}(\phi_0)$$

where the system $\mathcal{H}_{\cap\mathfrak{f}_1}$ has data $(F, C_{\cap\mathfrak{f}_1}, G, D_{\cap\mathfrak{f}_1})$ with

$$C_{\cap\mathfrak{f}_1} = C \cap \{x \in \mathbb{R}^n \,:\, \mathfrak{f}_1(x) = \texttt{True}\}, \qquad D_{\cap\mathfrak{f}_1} = D \cap \{x \in \mathbb{R}^n \,:\, \mathfrak{f}_1(x) = \texttt{True}\}$$

and $\mathcal{S}_{\mathcal{H}_{\cap\mathfrak{f}_1}}(\phi_0)$ and $\mathcal{S}_{\mathcal{H}}(\phi_0)$ denote the collection of maximal solutions to $\mathcal{H}$ and to $\mathcal{H}_{\cap\mathfrak{f}_1}$ from ϕ_0, respectively.

Similar conditions can be derived for other operators and formulae.

The conditions above can be checked using those in Section III-A characterizing existence and type of solutions to hybrid inclusions. Note that time-varying formulae $\mathfrak{f}_1(x,t,j)$ can be considered by incorporating τ and k as states keeping track of flow time t and jump time j, respectively.

E. Robustness

In this section, we present conditions on the data of a hybrid inclusion that guarantee that small perturbations do not significantly change the behavior of solutions. In [1], these conditions are shown to be key in assuring that perturbed solutions are close to some unperturbed solution and that asymptotic stability is robust to small perturbations. Robustness to small perturbations is of particular interest for cyber-physical systems due to the fact that several of the perturbations introduced during the implementation of the cyber components can be assumed to be small. Such is the case of quantization and discretization effects, which while cannot always be neglect, can be assume to be small.

We consider hybrid inclusions without inputs, perhaps due to the result of assigning their inputs via a function of the state. If the set $\mathcal{A}$ is compact and the data of the hybrid inclusion is such that the resulting flow and jump maps are "continuous" and the flow and jump sets are closed, then the asymptotic stability property is robust to small perturbations. To formally state this result, we introduce the following notions for a set-valued map. A set-valued map $S : \mathbb{R}^n \rightrightarrows \mathbb{R}^m$ is *outer semicontinuous* at $x \in \mathbb{R}^n$ if for each sequence $\{x_i\}_{i=1}^{\infty}$ converging to a point $x \in \mathbb{R}^n$ and each sequence $y_i \in S(x_i)$ converging to a point y, it holds that $y \in S(x)$; see [13, Definition 5.4]. Given a set $X \subset \mathbb{R}^n$, it is *outer semicontinuous relative to* X if the set-valued mapping from $\mathbb{R}^n$ to $\mathbb{R}^m$ defined by $S(x)$ for $x \in X$ and $\emptyset$ for $x \notin X$ is outer semicontinuous at each $x \in X$. It is *locally bounded* if, for each compact set $K \subset \mathbb{R}^n$ there exists a compact set $K' \subset \mathbb{R}^n$ such that $S(K) := \cup_{x \in K} S(x) \subset K'$.

Then, following [14, Theorem 6.6], if the data of the hybrid inclusion satisfies

(A1) C and D are closed sets;

(A2) $F : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ is outer semicontinuous and locally bounded, and $F(x)$ is nonempty and convex for all $x \in C$;

(A3) $G : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ is outer semicontinuous and locally bounded, and $G(x)$ is a nonempty subset of $\mathbb{R}^n$ for all $x \in D$;

and the compact set $\mathcal{A} \subset \mathbb{R}^n$ is asymptotically stable for the hybrid inclusion, then there exists a $\mathcal{KL}$ function[2] β such that for each $\varepsilon > 0$ and each compact set $K \subset \mathbb{R}^n$, there exists $\delta > 0$ such that every maximal solution ϕ to a perturbation of the hybrid inclusion starting from K satisfies

$$|\phi(t,j)|_{\mathcal{A}} \leq \beta(|\phi(0,0)|_{\mathcal{A}}, t+j) + \varepsilon \quad \forall (t,j) \in \operatorname{dom} \phi \tag{41}$$

where the perturbation of the hybrid inclusion is given by

$$\begin{array}{lll} \dot{x} & \in F_\delta(x) & x \in C_\delta \\ x^+ & \in G_\delta(x) & x \in D_\delta \end{array} \tag{42}$$

with

$$\begin{array}{rcl} F_\delta(x) & := & \overline{\text{co}} F(x + \delta\mathbb{B}) + \delta\mathbb{B} \\ G_\delta(x) & := & \{\eta \,:\, \eta \in x' + \delta\mathbb{B}, x' \in G(x + \delta\mathbb{B})\} \\ C_\delta & := & \{x \,:\, (x + \delta\mathbb{B}) \cap C \neq \emptyset\} \\ D_\delta & := & \{x \,:\, (x + \delta\mathbb{B}) \cap D \neq \emptyset\} \end{array}$$

The $\mathcal{KL}$ estimate in (41) guarantees that, when the data of the former hybrid inclusion is perturbed by δ, every solution $(t,j) \mapsto \phi(t,j)$ to it is such that it approaches $\mathcal{A} + \varepsilon\mathbb{B}$ when $t+j$, $(t,j) \in \operatorname{dom} \phi$, grows unbounded.

[2]A function $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is said to belong to class-$\mathcal{KL}$ ($\beta \in \mathcal{KL}$) if it is continuous, nondecreasing in its first argument, nonincreasing in its second argument, and $\lim_{s \searrow 0} \beta(s,r) = \lim_{r \to \infty} \beta(s,r) = 0$.

F. Simulation

The coupling between physics and computations in cyber-physical systems makes their simulation difficult. A simulator for cyber-physical systems has to be capable of computing the solution while evolving according to the physics of the system while monitoring the variables for a potential event triggering a discrete transition in the cyber components. A few software packages are available for numerical simulation of systems with such behavior, including Modelica [15], Ptolemy [16], Charon [17], HYSDEL [18], HyVisual [19], among others. In this article, we summarize a recent software package developed for the simulation of hybrid equations within Matlab/Simulink, namely, the Hybrid Equations (HyEQ) Toolbox [20]. Due to its capability of modeling interconnections of hybrid systems given in terms of hybrid equations, which are the single-valued version of hybrid inclusions (their flow map and jump map are single valued), within the HyEQ Toolbox one can model the physical components, the cyber components, and the subsystems used to interconnect them separately within Simulink and then interconnect them to define the entire cyber-physical system.

Over a finite amount of flow and a finite number of jumps, the HyEQ Toolbox computes an approximation of a solution ϕ by evaluating the flow condition imposed by C and the jump condition imposed by D, and according to the result of this evaluation, by appropriately discretizing the differential equation defining the flows in (24) or computing the new value of the state after jumps using (25). In this way, the HyEQ Toolbox returns a discrete version of ϕ and its hybrid time domain $\operatorname{dom}\phi$. More precisely, given an input γ, the computed version of the solution (ϕ, γ) is given by

$$\phi_s : \operatorname{dom}\phi_s \to \mathbb{R}^n, \qquad \gamma_s : \operatorname{dom}\gamma_s \to \mathbb{R}^m$$

which we call *a simulated solution* of the hybrid inclusion, and satisfies

$$x_s^+ = F_{P,s}(x_s, \gamma_s) \quad (x_s, \gamma_s) \in C_{P,s} \tag{43}$$

over the intervals of flow and, at jumps, satisfies the discrete dynamics

$$x_s^+ = G_{P,s}(x_s, \gamma_s) \quad (x_s, \gamma_s) \in D_{P,s} \tag{44}$$

The input γ_s is the discretization of γ. The function $F_{P,s}$ is the resulting discretized flow map obtained when employing an integration scheme for the differential equation $\dot{x} = F_P(x, \gamma)$. For instance, when the integration scheme is given by the forward Euler integration scheme, $F_{P,s}(x_s, \gamma_s) = x_s + sF_P(x_s, \gamma_s)$ with $s > 0$ denoting the step size for integration. Similarly, $G_{P,s}$, $C_{P,s}$, and $D_{P.s}$ are the discretized version of G_P, C_P, and D_P, respectively. Formal definitions of simulated solutions and dynamical properties of the discretization (43)-(44) of a hybrid inclusion can be found in [21].

Within this framework for simulation, the HyEQ Toolbox includes two scripts to compute solutions to a hybrid equation:

1) a Matlab script for simulation of hybrid equations within Matlab's workspace, called *Lite HyEQ Simulator*, and
2) a Simulink library and associated Matlab scripts for simulation of hybrid equations within Simulink, called *HyEQ Simulator*.

Figure 4 shows the Simulink implementation of a hybrid inclusion, in which the user needs to enter the data (C_P, F_P, D_P, G_P). Several examples of systems that can be simulated within this toolbox, including some specific cyber-physical systems can be found in [22].

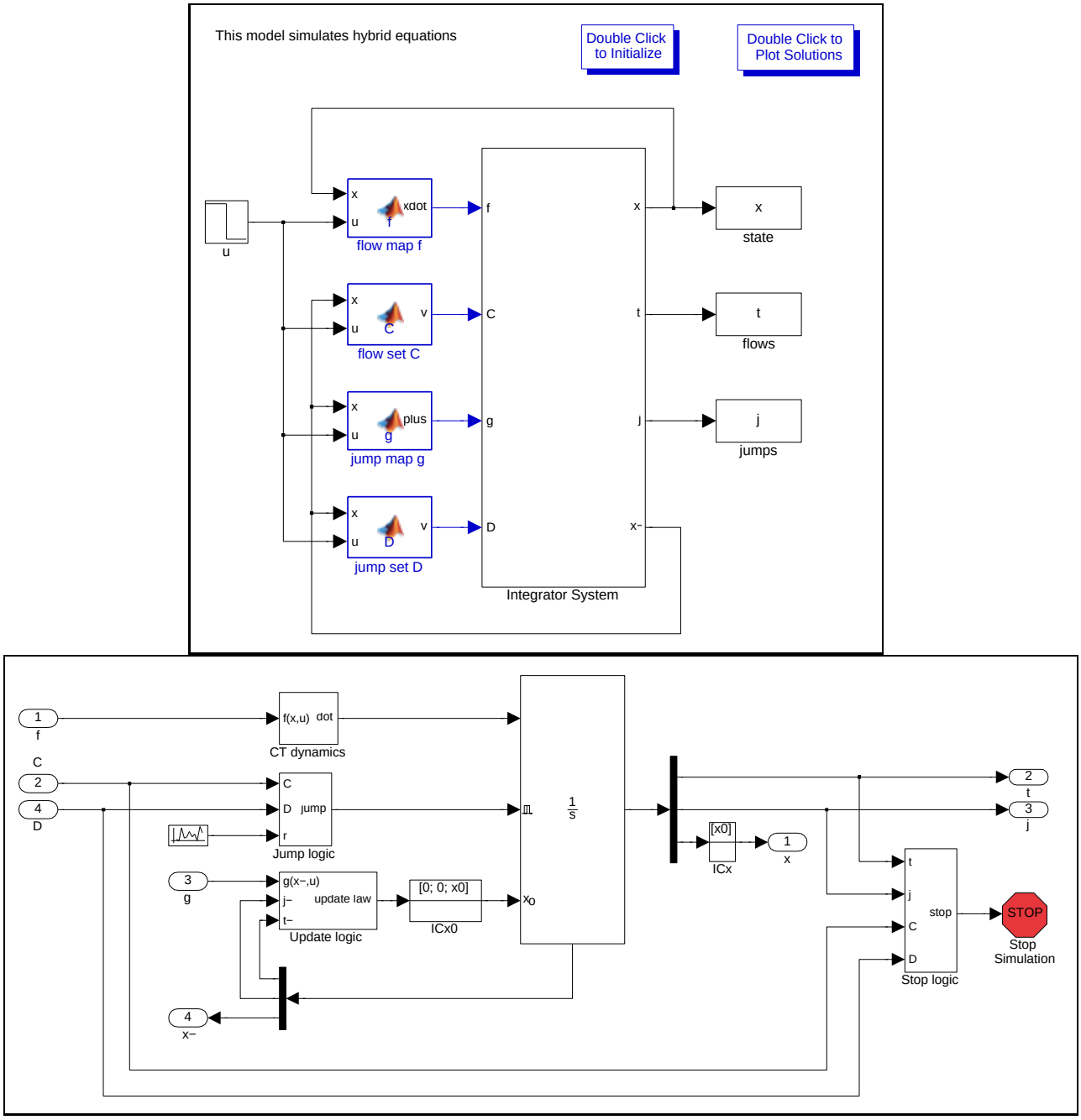


Fig. 4. Matlab/Simulink implementation of a hybrid inclusion with data (C, f, D, g) with inputs (left). Internals of integrator system (right).

IV. Conclusion

In this article, we showed that a class of cyber-physical systems can be treated as hybrid dynamical systems within the framework of hybrid inclusions. Modeling and analysis tools for such systems were summarized and illustrated in several examples. These tools are amenable for the mathematical analysis and design of cyber-physical systems. Additional tools that did not get covered due to space limitations include the design of feedback controllers using control

Lyapunov functions [2], passivity and passivity-based control [23], and the input/output stability tools in [24], [25].

V. Acknowledgments

This research has been partially supported by the National Science Foundation under CAREER Grant no. ECS-1150306 and by the Air Force Office of Scientific Research under Grant no. FA9550-12-1-0366. The author would like to thank Ryan Kastner for discussions on models of computations and to the students in his Fall 2014 class on CPSs who provided useful feedback in the initial versions of this article.

References

[1] R. Goebel, R. G. Sanfelice, and A. R. Teel. *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. Princeton University Press, New Jersey, 2012.

[2] R. G. Sanfelice. On the existence of control Lyapunov functions and state-feedback laws for hybrid systems. *IEEE Transactions on Automatic Control*, 58(12):3242–3248, December 2013.

[3] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[4] A. Girault, B. Lee, and E. A. Lee. Hierarchical finite state machines with multiple concurrency models. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18:742–760, 1999.

[5] R. G. Sanfelice, R. Goebel, and A. R. Teel. Invariance principles for hybrid systems with connections to detectability and asymptotic stability. *IEEE Transactions on Automatic Control*, 52(12):2282–2297, 2007.

[6] E. A. Lee. Computing for embedded systems. In *Proc. IEEE Instrumentation and Measurement*, volume 3, pages 1830–1837, 2001.

[7] P. Hartman. *Ordinary Differential Equations*. Birkhauser, 1982.

[8] H.K. Khalil. *Nonlinear Systems*. Prentice Hall, 3rd edition, 2002.

[9] R.G. Sanfelice, R. Goebel, and A.R. Teel. Generalized solutions to hybrid dynamical systems. *ESAIM: Control, Optimisation and Calculus of Variations*, 14(4):699–724, 2008.

[10] R. G. Sanfelice. Results on input-to-output and input-output-to-state stability for hybrid systems and their interconnections. In *Proc. 49th IEEE Conference on Decision and Control*, pages 2396–2401, 2010.

[11] R. Ferrante, F. Gouaisbaut, R. G. Sanfelice, and S. Tarbouriech. An observer with measurement-triggered jumps for linear systems with known input. In *To appear in the Proceedings of the 19th IFAC World Congress*, 2014.

[12] S. Karaman, R.G. Sanfelice, and E. Frazzoli. Optimal control of mixed logical dynamical systems with linear temporal logic specifications. In *Proc. 47th IEEE Conference on Decision and Control*, pages 2117–2122, 2008.

[13] R.T. Rockafellar and R. J-B Wets. *Variational Analysis*. Springer, Berlin Heidelberg, 1998.

[14] R. Goebel and A.R. Teel. Solutions to hybrid inclusions via set and graphical convergence with stability theory applications. *Automatica*, 42(4):573–587, 2006.

[15] H. Elmqvist, S.E. Mattsson, and M. Otter. Modelica: The new object-oriented modeling language. In *Proc. of 12th European Simulation Multiconference*, 1998.

[16] J. Liu and E. A. Lee. A component-based approach to modeling and simulating mixed-signal and hybrid systems. *ACM Transactions on Modeling and Computer Simulation*, 12(4):343–368, 2002.

[17] R. Alur, T. Dang, J. Esposito, Y. Hur, F. Ivancic, V. Kumar, I. Lee, P. Mishra, G. J. Pappas, and O. Sokolsky. Hierarchical modeling and analysis of embedded systems. *Proc. of the IEEE*, 91:11–28, 2003.

[18] F.D. Torrisi and A. Bemporad. HYSDEL - A tool for generating computational hybrid models for analysis and synthesis problems. *IEEE Trans. Control Systems Technology*, 12:235–249, 2004.

[19] E. A. Lee and H. Zheng. Operational semantics for hybrid systems. In *Hybrid Systems: Computation and Control: 8th International Workshop*, pages 25–53, 2005.

[20] R. G. Sanfelice. Hybrid Equations (HyEQ) Toolbox. http://www.u.arizona.edu/∼sricardo/index.php?n=Main.Software, 2013.

[21] R. G. Sanfelice and A. R. Teel. Dynamical properties of hybrid systems simulators. *Automatica*, 46(2):239–248, 2010.

[22] R. G. Sanfelice. Online Blog of the Hybrid Equations (HyEQ) Toolbox. http://hybridsimulator.wordpress.com, 2013.

[23] R. Naldi and R. G. Sanfelice. Passivity-based control for hybrid systems with applications to mechanical systems exhibiting impacts. *Automatica*, 49(5):1104–1116, May 2013.

[24] R. G. Sanfelice. Input-output-to-state stability tools for hybrid systems and their interconnections. *To appear in IEEE Transactions on Automatic Control*, 2014.

[25] R. G. Sanfelice. Interconnections of hybrid systems: Some challenges and recent results. *Journal of Nonlinear Systems and Applications*, pages 111–121, 2011.