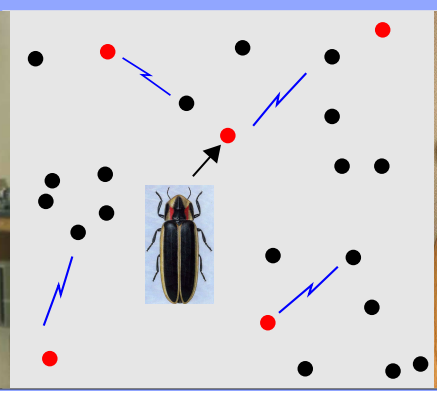
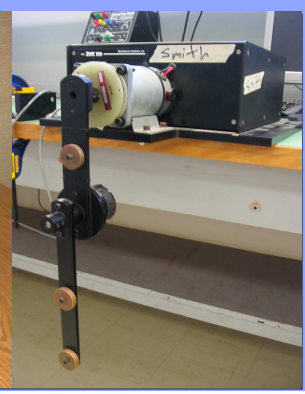
Dynamical Models of Cyber-physical Systems (CPSs)

Ricardo Sanfelice

Department of Computer Engineering
Hybrid Systems Lab
University of California, Santa Cruz







Plan of Work

Plan of Work

- Continuous-time systems

Plan of Work

- Continuous-time systems
- Modeling of physical processes

Plan of Work

- Continuous-time systems
- Modeling of physical processes
- Linear time-invariant systems

Plan of Work

- Continuous-time systems
- Modeling of physical processes
- Linear time-invariant systems
- Numerical simulation of differential equations

Plan of Work

- Continuous-time systems
- Modeling of physical processes
- Linear time-invariant systems
- Numerical simulation of differential equations
- Discrete-time systems and return maps

Plan of Work

- Continuous-time systems
- Modeling of physical processes
- Linear time-invariant systems
- Numerical simulation of differential equations
- Discrete-time systems and return maps
- Finite state machines

Plan of Work

- Continuous-time systems
- Modeling of physical processes
- Linear time-invariant systems
- Numerical simulation of differential equations
- Discrete-time systems and return maps
- Finite state machines
- Event triggered systems

Plan of Work

- Continuous-time systems
- Modeling of physical processes
- Linear time-invariant systems
- Numerical simulation of differential equations
- Discrete-time systems and return maps
- Finite state machines
- Event triggered systems
- Stateflow

Plan of Work

- Continuous-time systems
- Modeling of physical processes
- Linear time-invariant systems
- Numerical simulation of differential equations
- Discrete-time systems and return maps
- Finite state machines
- Event triggered systems
- Stateflow
- Timed automata

Plan of Work

- Continuous-time systems
- Modeling of physical processes
- Linear time-invariant systems
- Numerical simulation of differential equations
- Discrete-time systems and return maps
- Finite state machines
- Event triggered systems
- Stateflow
- Timed automata
- Hybrid automata

Plan of Work

- Continuous-time systems
- Modeling of physical processes
- Linear time-invariant systems
- Numerical simulation of differential equations
- Discrete-time systems and return maps
- Finite state machines
- Event triggered systems
- Stateflow
- Timed automata
- Hybrid automata
- Invariants

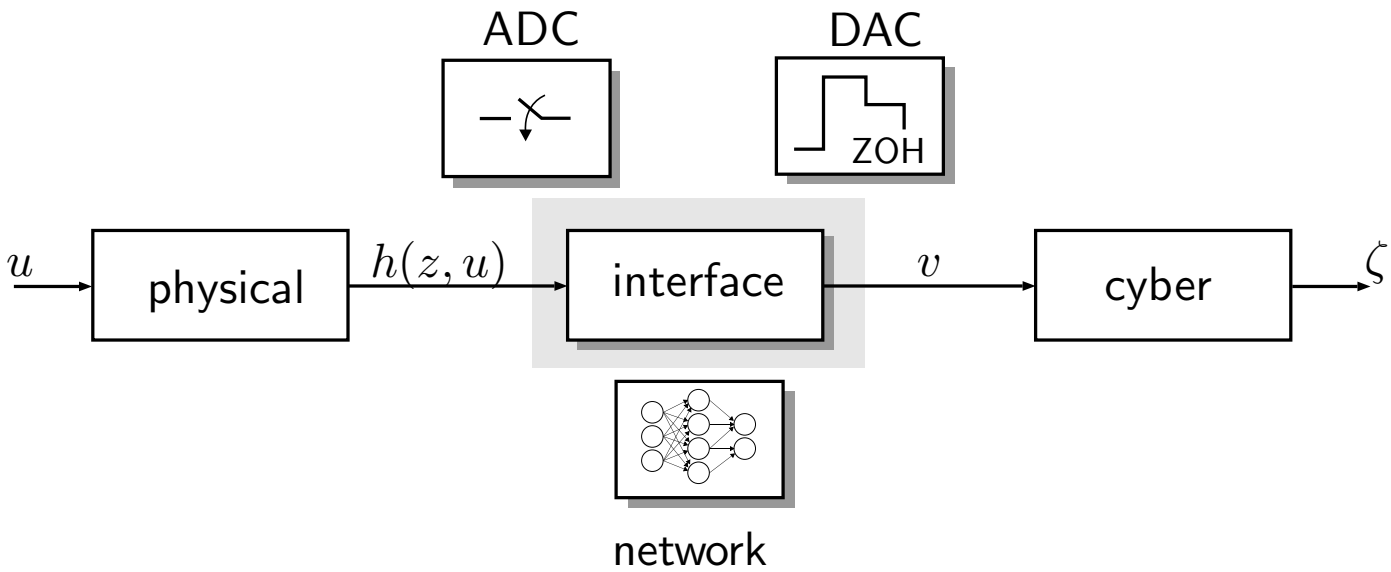
Plan of Work

- Continuous-time systems
- Modeling of physical processes
- Linear time-invariant systems
- Numerical simulation of differential equations
- Discrete-time systems and return maps
- Finite state machines
- Event triggered systems
- Stateflow
- Timed automata
- Hybrid automata
- Invariants
- Linear temporal logic

Plan of Work

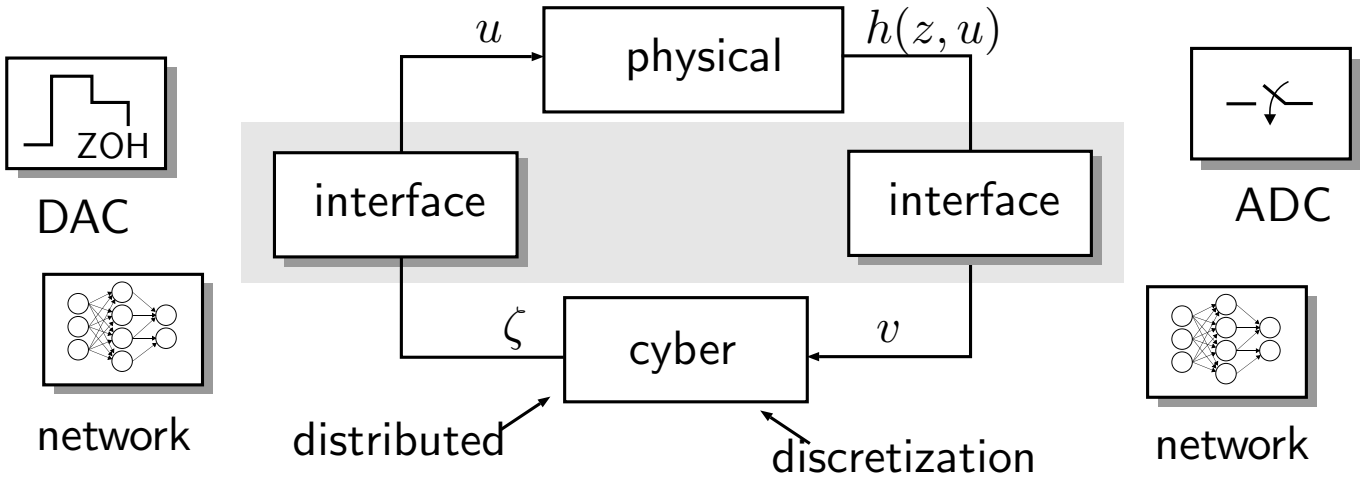
- Continuous-time systems
- Modeling of physical processes
- Linear time-invariant systems
- Numerical simulation of differential equations
- Discrete-time systems and return maps
- Finite state machines
- Event triggered systems
- Stateflow
- Timed automata
- Hybrid automata
- Invariants
- Linear temporal logic
- Verification

Interconnection Between Physical and Cyber Components



Cyber-physical systems: series 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).

Interconnection Between Physical and Cyber Components



Cyber-physical systems: feedback interconnection 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).

Outline

1. Models of physical components
2. Models of cyber components
   - Pure finite state machines
   - Finite state machines with conditional structures as guards
   - Models of computer computations and discrete-time algorithms
3. Models of the interface between physical and cyber components
   - Analog-to-Digital converters
   - Digital-to-Analog converters
   - Digital networks
4. Combining models of physical and cyber components
5. Simulating CPSs

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.

- state variable is denoted by $z \in \mathbb{R}^{n_P}$

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.

- state variable is denoted by $z \in \mathbb{R}^{n_P}$
- the input signals is denoted by $u \in \mathbb{R}^{m_P}$

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.

- state variable is denoted by $z \in \mathbb{R}^{n_P}$
- the input signals is denoted by $u \in \mathbb{R}^{m_P}$
- the output $y \in \mathbb{R}^{r_P}$ is defined by a function h

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.

- state variable is denoted by $z \in \mathbb{R}^{n_P}$
- the input signals is denoted by $u \in \mathbb{R}^{m_P}$
- the output $y \in \mathbb{R}^{r_P}$ is defined by a function h

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.

- state variable is denoted by $z \in \mathbb{R}^{n_P}$
- the input signals is denoted by $u \in \mathbb{R}^{m_P}$
- the output $y \in \mathbb{R}^{r_P}$ is defined by a function h

The general mathematical description of the physical component is

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

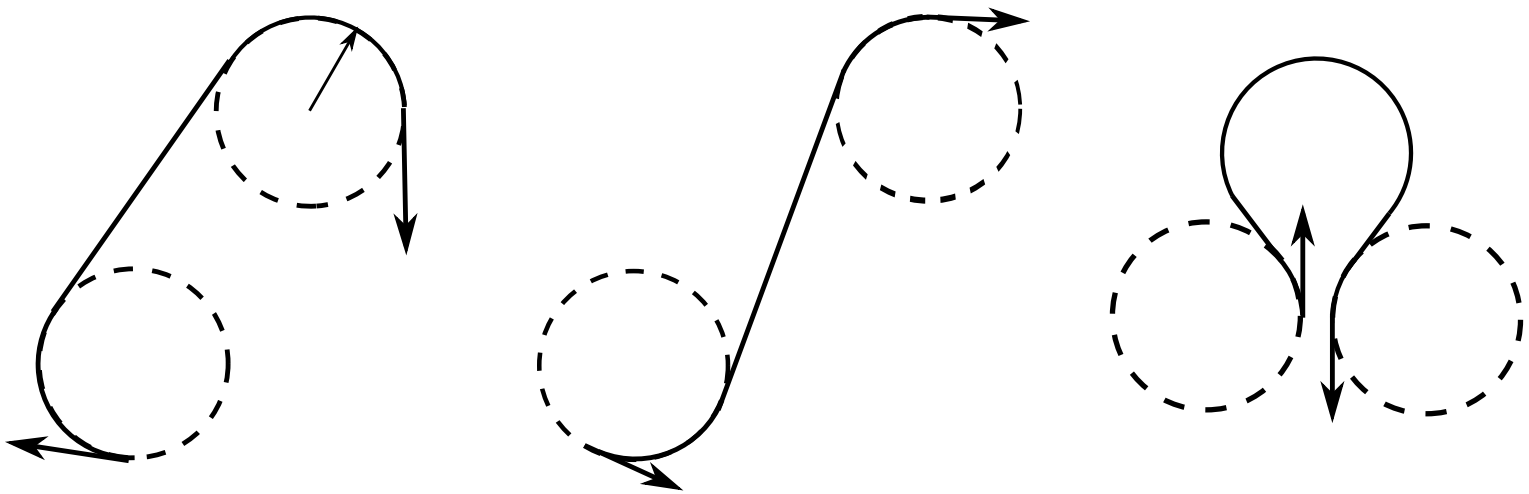
subject to the constraint

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

An Example of Physical Component

Dubins vehicle

Assumed to be a particle on the plane that describes smooth paths and capable of making turns satisfying a minimum turning radius constraint (similar to a car).



An Example of Physical Component

Dubins vehicle

Assumed to be a particle on the plane that describes smooth paths and capable 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), it can be captured as in

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

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.

Other Continuous-time Models

Models of Cyber Components

The cyber components of a CPS include those in charge of performing computations, implementing algorithms, and transmitting digital data over networks.

- variables change at discrete events, not necessarily periodically

We capture it by difference equations

The general mathematical description of the cyber component is

$$\eta^+ \in G_C(\eta, v), \quad \zeta = \kappa(\eta, v)$$

with the constraint

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

Models of Cyber Components

The cyber components of a CPS include those in charge of performing computations, implementing algorithms, and transmitting digital data over networks.

- variables change at discrete events, not necessarily periodically

We capture it by difference equations

- state variable is denoted by $\eta \in \Upsilon \subset \mathbb{R}^{n_C}$

The general mathematical description of the cyber component is

$$\eta^+ \in G_C(\eta, v), \quad \zeta = \kappa(\eta, v)$$

with the constraint

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

Models of Cyber Components

The cyber components of a CPS include those in charge of performing computations, implementing algorithms, and transmitting digital data over networks.

- variables change at discrete events, not necessarily periodically

We capture it by difference equations

- state variable is denoted by $\eta \in \Upsilon \subset \mathbb{R}^{n_C}$
- the input signals is denoted by $v \in \mathbb{V} \subset \mathbb{R}^{m_C}$

The general mathematical description of the cyber component is

$$\eta^+ \in G_C(\eta, v), \quad \zeta = \kappa(\eta, v)$$

with the constraint

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

Models of Cyber Components

The cyber components of a CPS include those in charge of performing computations, implementing algorithms, and transmitting digital data over networks.

- variables change at discrete events, not necessarily periodically

We capture it by difference equations

- state variable is denoted by $\eta \in \Upsilon \subset \mathbb{R}^{n_C}$
- the input signals is denoted by $v \in \mathbb{V} \subset \mathbb{R}^{m_C}$
- the output ζ is defined by the function $\zeta = \kappa(\eta, v)$

The general mathematical description of the cyber component is

$$\eta^+ \in G_C(\eta, v), \quad \zeta = \kappa(\eta, v)$$

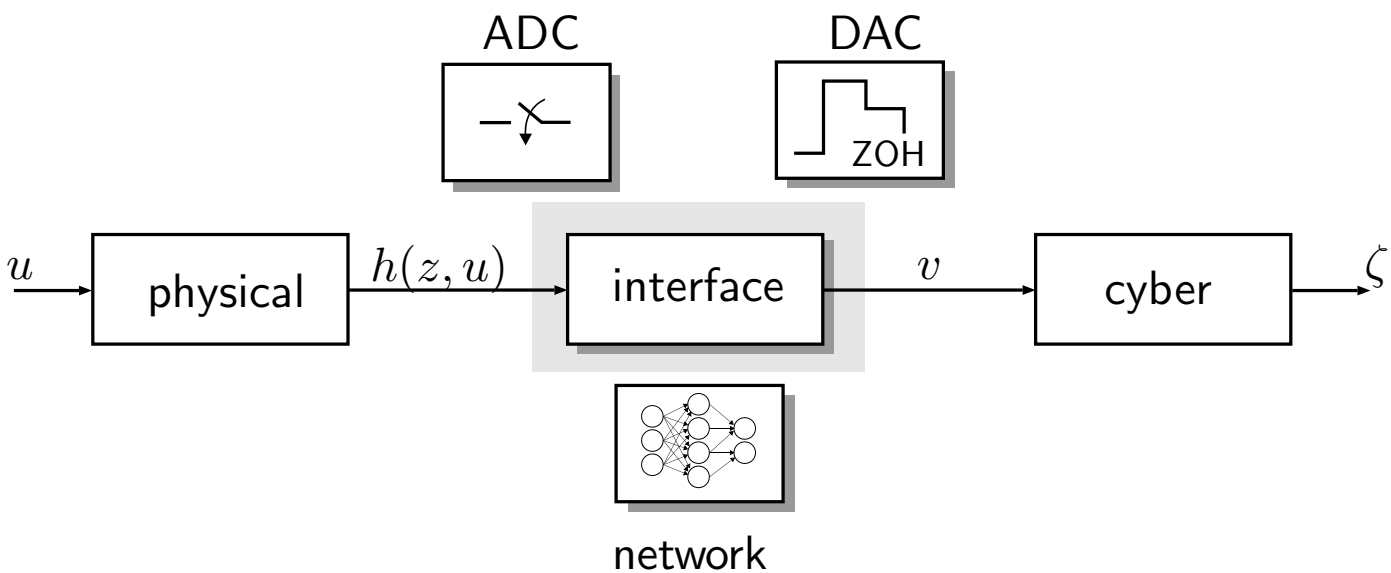
with the constraint

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

Models of Cyber Components

Widely used models of cyber components

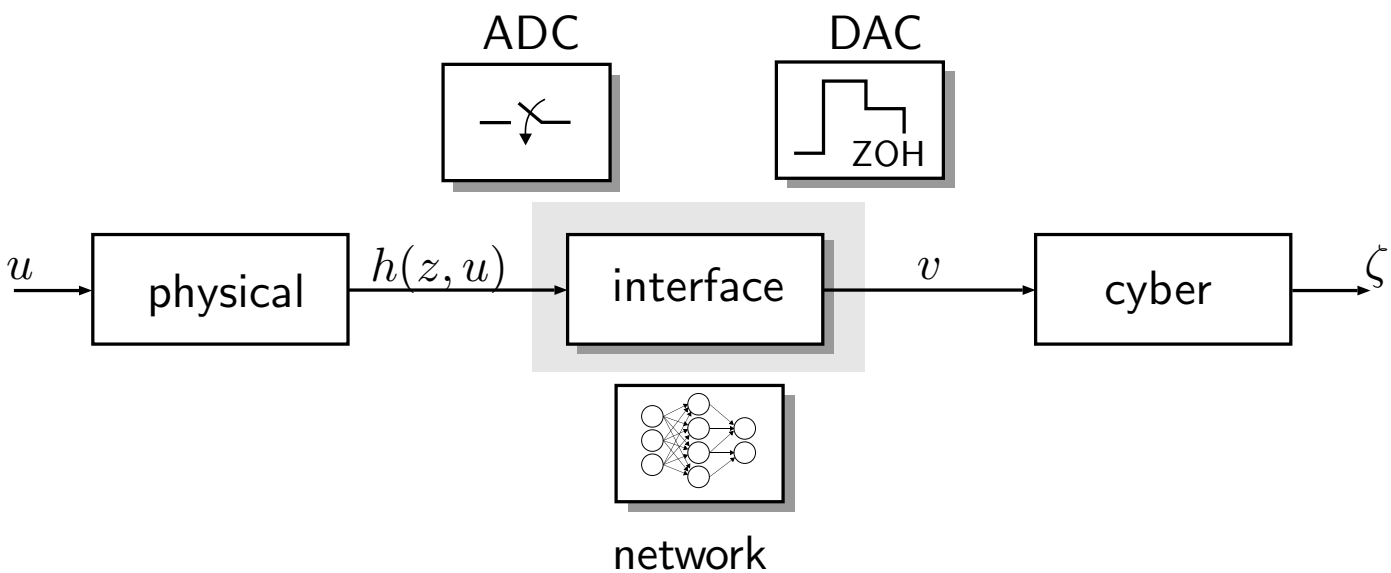
- Discrete-time models



Models of Cyber Components

Widely used models of cyber components

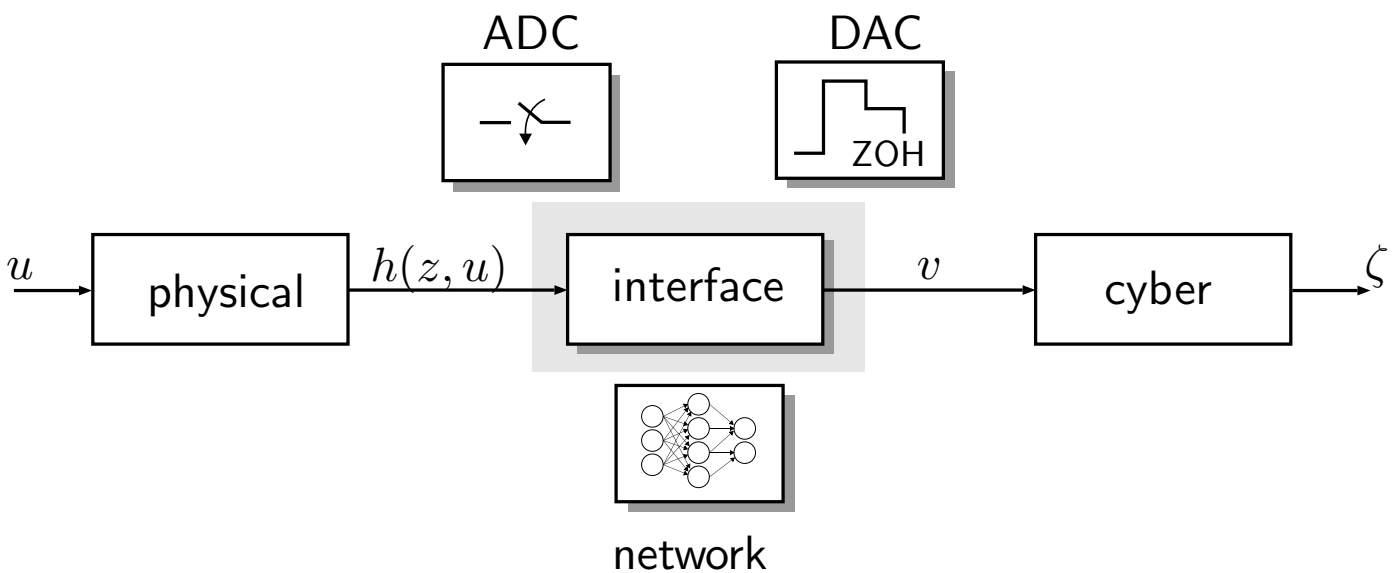
- Discrete-time models
- Pure finite state machines



Models of Cyber Components

Widely used models of cyber components

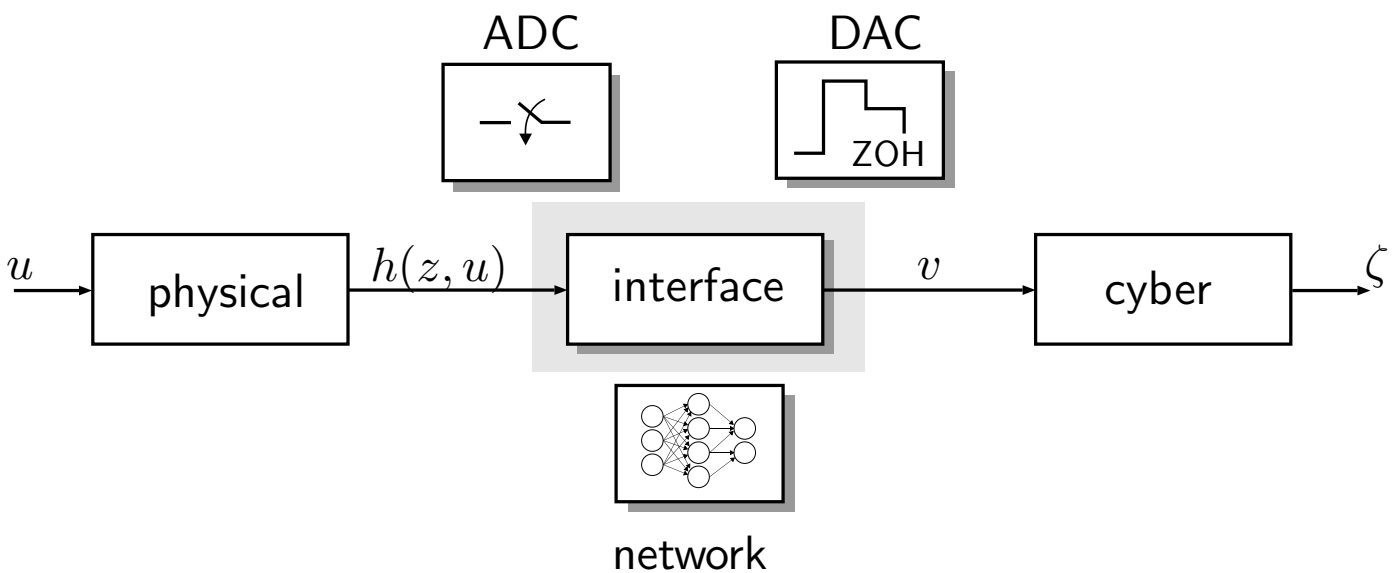
- Discrete-time models
- Pure finite state machines
- Conditional/Triggered finite state machines



Models of Cyber Components

Widely used models of cyber components

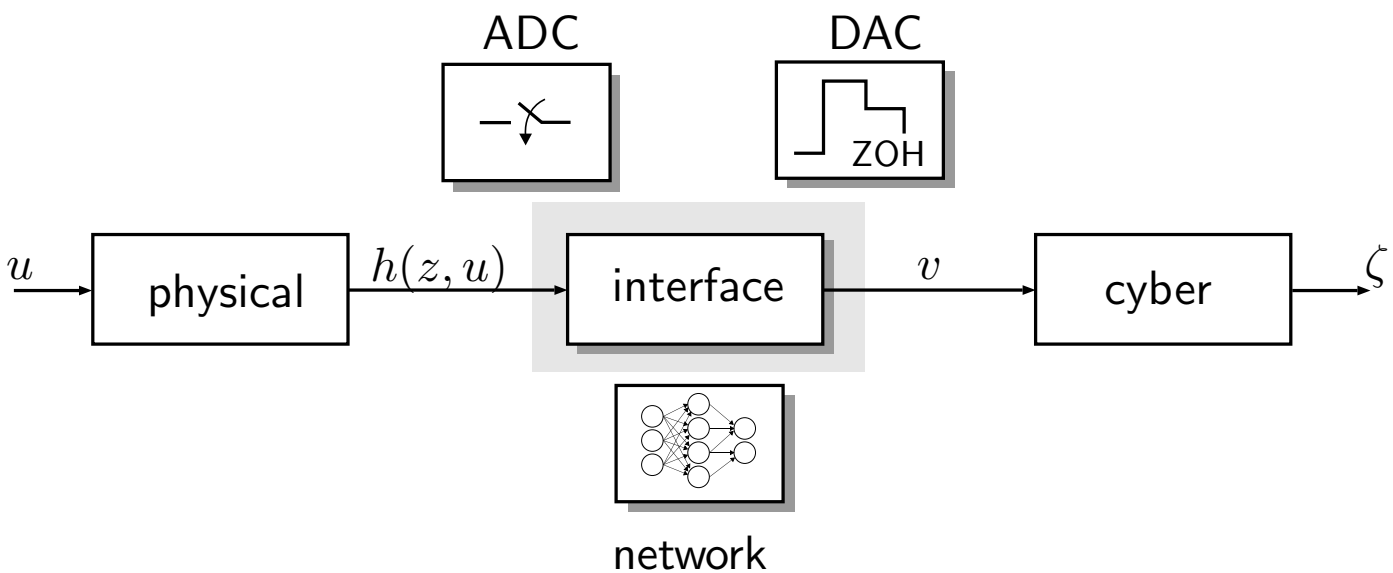
- Discrete-time models
- Pure finite state machines
- Conditional/Triggered finite state machines
- Automata



Models of Cyber Components

Widely used models of cyber components

- Discrete-time models
- Pure finite state machines
- Conditional/Triggered finite state machines
- Automata
- ⋮



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. Let v denote the inputs, q denote the states (or mode), and r denote the outputs of the FSM.

- An input alphabet Σ where v takes values from;

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. Let v denote the inputs, q denote the states (or mode), and r denote the outputs of the FSM.

- An input alphabet Σ where v takes values from;
- A finite set of states Q where q takes values from;

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. Let v denote the inputs, q denote the states (or mode), and r denote the outputs of the FSM.

- 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;

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. Let v denote the inputs, q denote the states (or mode), and r denote the outputs of the FSM.

- 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 $\kappa : Q \to \Delta$;

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. Let v denote the inputs, q denote the states (or mode), and r denote the outputs of the FSM.

- 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 $\kappa : Q \to \Delta$;
- A transition function $\delta : Q \times \Sigma \to Q$.

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. Let v denote the inputs, q denote the states (or mode), and r denote the outputs of the FSM.

- 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 $\kappa : Q \to \Delta$;
- A transition function $\delta : Q \times \Sigma \to Q$.

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. Let v denote the inputs, q denote the states (or mode), and r denote the outputs of the FSM.

- 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 $\kappa : Q \to \Delta$;
- 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 $Q_\infty \subset Q$.

$$q^+ = \delta(q, v) \qquad \zeta = \kappa(q) \qquad (q, v) \in Q \times \Sigma$$

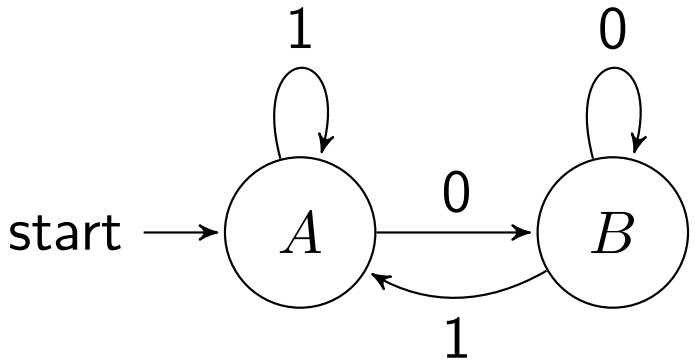
An Example of a FSM

$$Q = \{A, B\}$$

$$\Sigma = \{0, 1\}$$

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

$$\kappa(q) = q$$



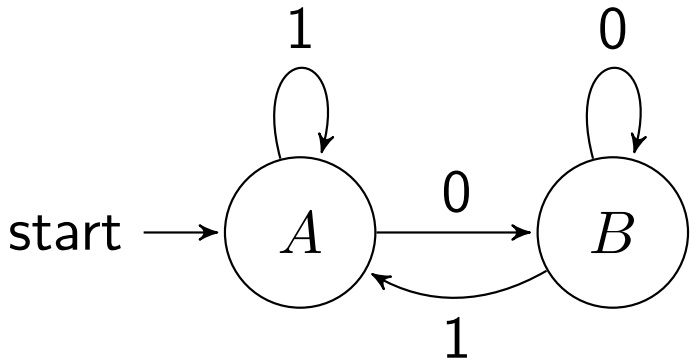
An Example of a FSM

$$Q = \{A, B\}$$

$$\Sigma = \{0, 1\}$$

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

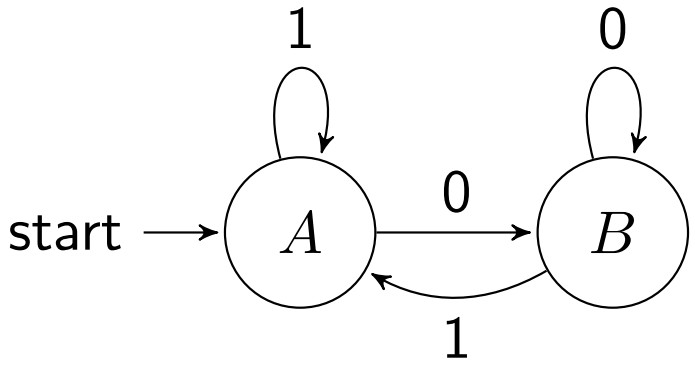
$$\kappa(q) = q$$



Note that this FSM does not terminate.

An Example of a FSM

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



Note that this FSM does not terminate.

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

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

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

FSM Triggered Based On Conditional Structures

To model conditional structures within a pure FSM,

- the input alphabet Σ has to be of infinite size,

FSM Triggered Based On Conditional Structures

To model conditional structures within a pure FSM,

- the input alphabet Σ has to be of infinite size,
- 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,

FSM Triggered Based On Conditional Structures

To model conditional structures within a pure FSM,

- the input alphabet Σ has to be of infinite size,
- 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,
- 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 $q \in Q$,

FSM Triggered Based On Conditional Structures

To model conditional structures within a pure FSM,

- the input alphabet Σ has to be of infinite size,
- 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,
- 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.

FSM Triggered Based On Conditional Structures

To model conditional structures within a pure FSM,

- the input alphabet Σ has to be of infinite size,
- 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,
- 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.

FSM Triggered Based On Conditional Structures

To model conditional structures within a pure FSM,

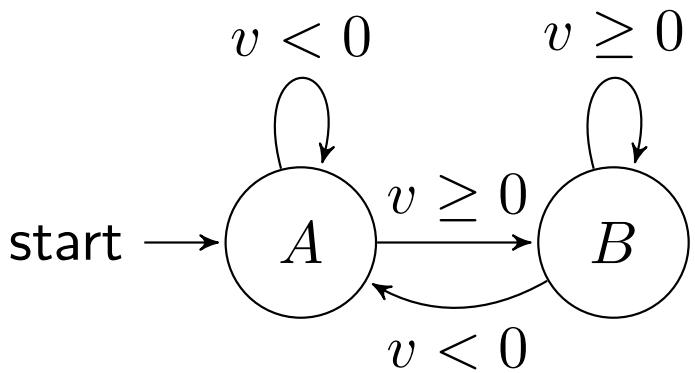
- the input alphabet Σ has to be of infinite size,
- allowing for an infinite alphabet and including the conditional structure as a guard, i.e., a boolean-valued expression that evaluates to true when the transition is enabled, and to false otherwise,
- let the function $\ell : Q \times \Sigma \times \Delta \to \mathbb{R}$ 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.

$$q^+ = \delta(q, v) \quad \zeta = \kappa(q) \quad \ell(q, v, \zeta) \leq 0, \quad (q, v) \in Q \times \Sigma$$

An Example of Triggered FSM

Suppose we want to transition when $v < 0$. Since $v \in \Sigma$ which is a finite set (finite alphabet), we need to check $v < 0$ over an infinite alphabet. The technical solution is to use a function Dataflow (DC) to implement:

```
function out = checkCond (v)
if v < 0 then
    out = 1;
else
    out = 0;
end if
return out
```



The output, which is 0 or 1, is feed to the pure FSM.

Models of Computer Computations

Computations and discrete-time algorithms

Models of Computer Computations

Computations and discrete-time algorithms

- As a purely discrete system, provides the outcome after one or a series of steps.

Models of Computer Computations

Computations and discrete-time algorithms

- As a purely discrete system, provides the outcome after one or a series of steps.
- It may require inputs v and the result could be used to determine the output ζ of the model.

Models of Computer Computations

Computations and discrete-time algorithms

- As a purely discrete system, provides the outcome after one or a series of steps.
- It may require inputs v and the result could be used to determine the output ζ of the model.
- One step can be modeled by the static system $\zeta = \widetilde{\kappa}(v)$, where the function $\widetilde{\kappa}$ models the computations.

Models of Computer Computations

Computations and discrete-time algorithms

- As a purely discrete system, provides the outcome after one or a series of steps.
- It may require inputs v and the result could be used to determine the output ζ of the model.
- One step can be modeled by the static system $\zeta = \widetilde{\kappa}(v)$, where the function $\widetilde{\kappa}$ models the computations.
- Discrete-time algorithms are naturally modeled using difference equations, which can be captured by

$$\eta^+ = G_C(\eta, v) \qquad \zeta = \kappa(\eta)$$

where G_C is obtained from discretizing the control algorithm.

Iterative Computations

Iterative implementations of computations require a number of steps to reach an answer and, at times, additional variables.

- denoting the additional variables as $m \in \mathbb{R}^{nC-1}$ and the counter as $k \in \{0, 1, 2, \ldots, k^*\}$, $k^* \in \{0, 1, 2, \ldots\} =: \mathbb{N}$,

Iterative Computations

Iterative implementations of computations require a number of steps to reach an answer and, at times, additional variables.

- denoting the additional variables as $m \in \mathbb{R}^{nC-1}$ and the counter as $k \in \{0, 1, 2, \ldots, k^*\}$, $k^* \in \{0, 1, 2, \ldots\} =: \mathbb{N}$,
- a discrete system that performs k^* iterations to provide the final outcome of the computations.

Iterative Computations

Iterative implementations of computations require a number of steps to reach an answer and, at times, additional variables.

- denoting the additional variables as $m \in \mathbb{R}^{nC-1}$ and the counter as $k \in \{0, 1, 2, \dots, k^*\}$, $k^* \in \{0, 1, 2, \dots\} =: \mathbb{N}$,
- 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

Iterative Computations

Iterative implementations of computations require a number of steps to reach an answer and, at times, additional variables.

- denoting the additional variables as $m \in \mathbb{R}^{nC-1}$ and the counter as $k \in \{0, 1, 2, \dots, k^*\}$, $k^* \in \{0, 1, 2, \dots\} =: \mathbb{N}$,
- 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

Iterative Computations

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 $k \in \{0, 1, 2, \dots, k^*\}$, $k^* \in \{0, 1, 2, \dots\} =: \mathbb{N}$,
- 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

then, a model is given by

$$\eta^+ = \begin{bmatrix} \widetilde{\kappa}(m, k, v) \\ k + 1 \end{bmatrix} \qquad \zeta = m,$$

$$k \in \{0, 1, 2, \dots, k^* - 1\}, m \in \mathbb{R}^{n_C - 1}, v \in \mathcal{V}$$

An Example of Iterative Computations

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

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

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

An Example of Iterative Computations

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

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

The one-step computation of the factorial of the integer a can be performed iteratively by initializing m to a and k to one, setting $k^* = a$, and defining

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

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

[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 $\widetilde{\kappa}(v) = \text{fact}(v)$.

Models of Systems at The Interface

Interfaces need to convert conditions and signals appropriately. Some widely used interfaces include

- **Analog-to-Digital Converters**: are commonly used to provide measurements of the physical systems to the cyber components

Models of Systems at The Interface

Interfaces need to convert conditions and signals appropriately. Some widely used interfaces include

- **Analog-to-Digital Converters**: are commonly used to provide measurements of the physical systems to the cyber components
- **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

Models of Systems at The Interface

Interfaces need to convert conditions and signals appropriately. Some widely used interfaces include

- **Analog-to-Digital Converters**: are commonly used to provide measurements of the physical systems to the cyber components
- **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 Networks**: the information transfer between the physical and cyber components, or between subsystems within the cyber components, might occur over a digital communication network

Analog-to-Digital Converters

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.

Analog-to-Digital Converters

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.

Denote a timer state $\tau_s \in \mathbb{R}_{\geq 0}$, a sample state $m_s \in \mathbb{R}^{r_P}$, the input to the sampling device $v_s \in \mathbb{R}^{r_p}$, then, the model of the sampling devices is

$$\begin{bmatrix} \dot{\tau}_s \\ \dot{m}_s \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad (\tau_s, m_s, v_s) \in C_s$$

$$\begin{bmatrix} \tau_s^+ \\ m_s^+ \end{bmatrix} = \begin{bmatrix} 0 \\ v_s \end{bmatrix} \qquad (\tau_s, m_s, v_s) \in D_s$$

$$C_s = \{(\tau_s, m_s, v_s) : \tau_s \in [0, T_s^*]\}$$

$$D_s = \{(\tau_s, m_s, v_s) : \tau_s \in [T_s^*, \infty)\}$$

Analog-to-Digital Converters

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.

Denote a timer state $\tau_s \in \mathbb{R}_{\geq 0}$, a sample state $m_s \in \mathbb{R}^{r_P}$, the input to the sampling device $v_s \in \mathbb{R}^{r_p}$, then, the model of the sampling devices is

$$\begin{aligned} \dot{\tau}_s = 1, \quad \dot{m}_s = 0 \qquad & \text{when} \quad \tau_s \in [0, T_s^*] \\ \tau_s^+ = 0, \quad m_s^+ = v_s \qquad & \text{when} \quad \tau_s \geq T_s^* \end{aligned}$$

Omitted effects (can be incorporated if needed):

- ADC acquisition time, limits the maximum sampling frequency

Analog-to-Digital Converters

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.

Denote a timer state $\tau_s \in \mathbb{R}_{\geq 0}$, a sample state $m_s \in \mathbb{R}^{r_P}$, the input to the sampling device $v_s \in \mathbb{R}^{r_p}$, then, the model of the sampling devices is

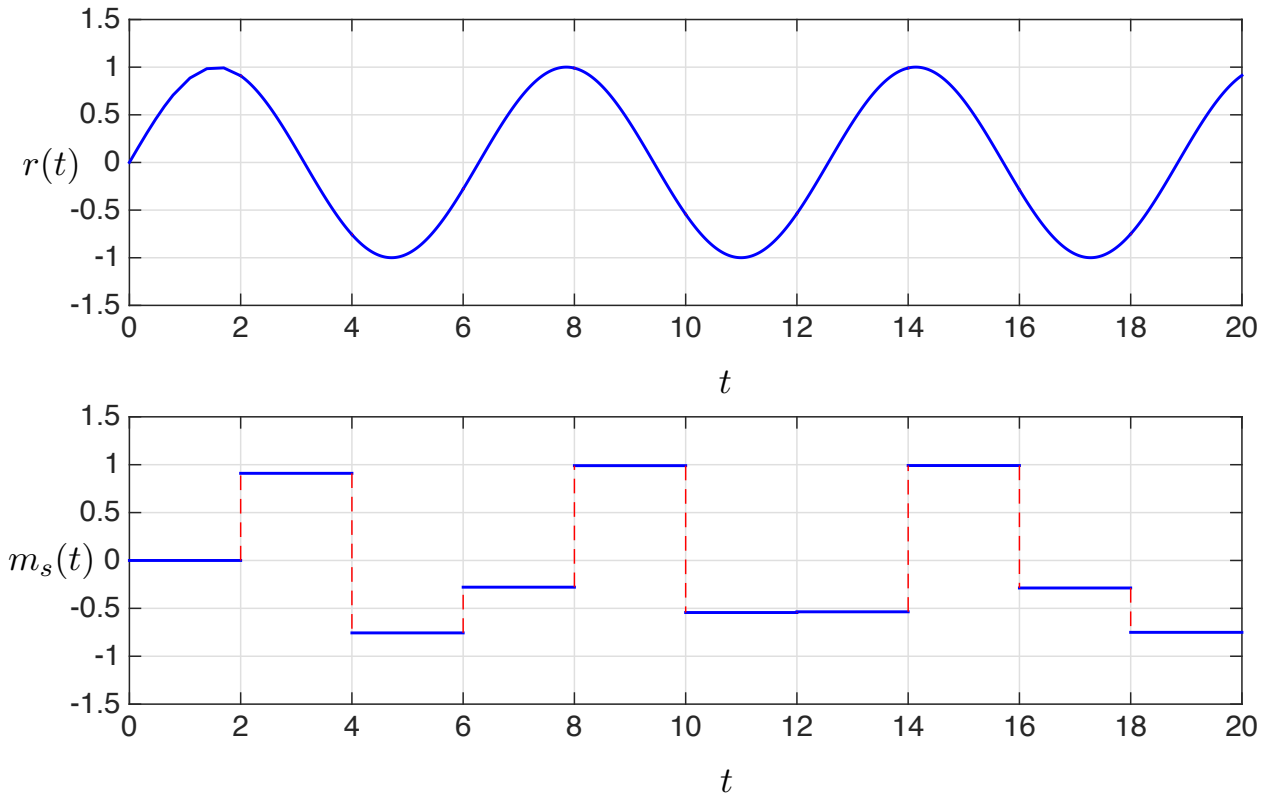
$$\begin{aligned} \dot{\tau}_s = 1, \quad \dot{m}_s = 0 \qquad & \text{when} \quad \tau_s \in [0, T_s^*] \\ \tau_s^+ = 0, \quad m_s^+ = v_s \qquad & \text{when} \quad \tau_s \geq T_s^* \end{aligned}$$

Omitted effects (can be incorporated if needed):

- ADC acquisition time, limits the maximum sampling frequency
- Finite-length store space, which causes quantization

A Simulation of ADC

Parameters: $T^* = 2$, $\tau_s(0,0) = 0$, $m_s(0,0) = 0$.



Writing the ADC Model as a Hybrid Inclusion

Digital-to-Analog Converters

Digital-to-analog converters (DACs) perform conversion from digital signals into analog equivalents. One of the most common models for a DAC is the zero-order hold model (ZOH).

Digital-to-Analog Converters

Digital-to-analog converters (DACs) perform conversion from digital signals into analog equivalents. One of the most common models for a DAC is the zero-order hold model (ZOH).

Let $\tau_h \in \mathbb{R}_{\geq 0}$ be the timer state, $m_h \in \mathbb{R}^{r_C}$ be the sample state, and $v_h \in \mathbb{R}^{r_C}$ be the inputs of the DAC. When $\tau_h \geq T_h^*$, the timer state is reset to zero and the sample state is updated with v_h.

A model is given by

$$\begin{aligned} \dot{\tau}_h = 1, \quad \dot{m}_h = 0 \qquad & \text{when} \quad \tau_h \in [0, T_h^*] \\ \tau_h^+ = 0, \quad m_h^+ = v_h \qquad & \text{when} \quad \tau_h \geq T_h^* \end{aligned}$$

Digital Networks

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.

Digital Networks

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.Assume that the information transmission 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

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

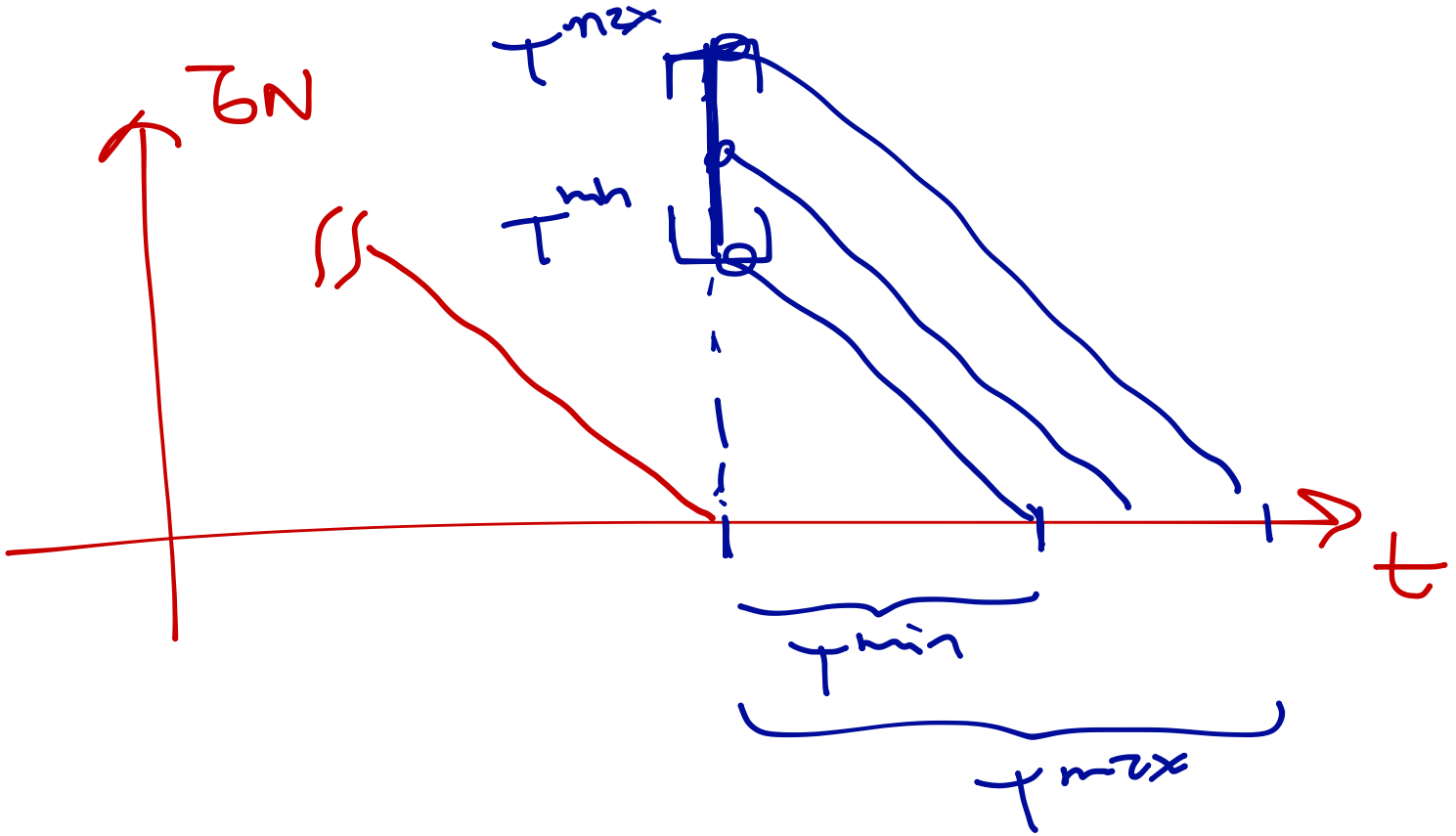
and i^* is the number of transmission events, which might be finite or infinite.

Digital Networks

A mathematical model capturing the said mechanism is given by

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

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



Models of Systems at The Interface

In general, the interface will be modeled as a hybrid inclusion with state λ, input w, output ψ, and dynamics

$$\begin{aligned} \dot{\lambda} &\in F_I(\lambda, w) && \text{when } (\lambda, w) \in C_I \\ \lambda^+ &\in G_I(\lambda, w) && \text{when } (\lambda, w) \in D_I \\ \psi &= \varphi(\lambda) \end{aligned}$$

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

Interconnection Between 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.

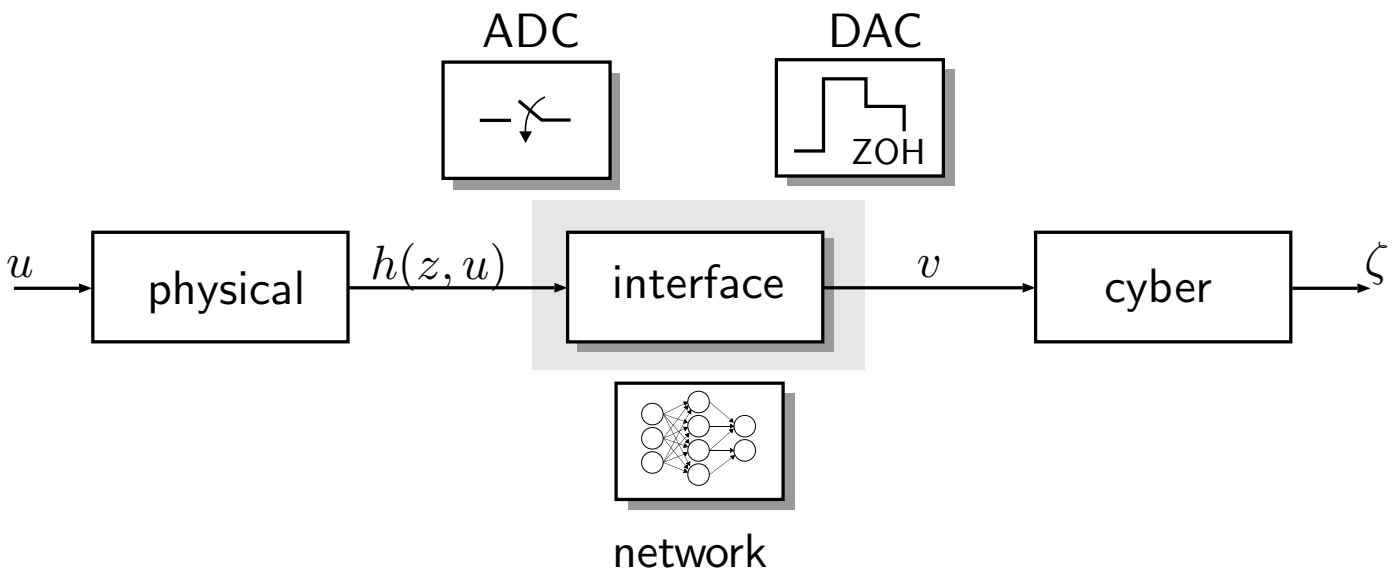


Figure : **Cyber-physical systems:** series interconnections between a plant (part of the physical component), controller (part of the cyber component), and interfaces/converters/signal conditioners (part of both the physical and cyber components).

Interconnection Between Physical and Cyber Components

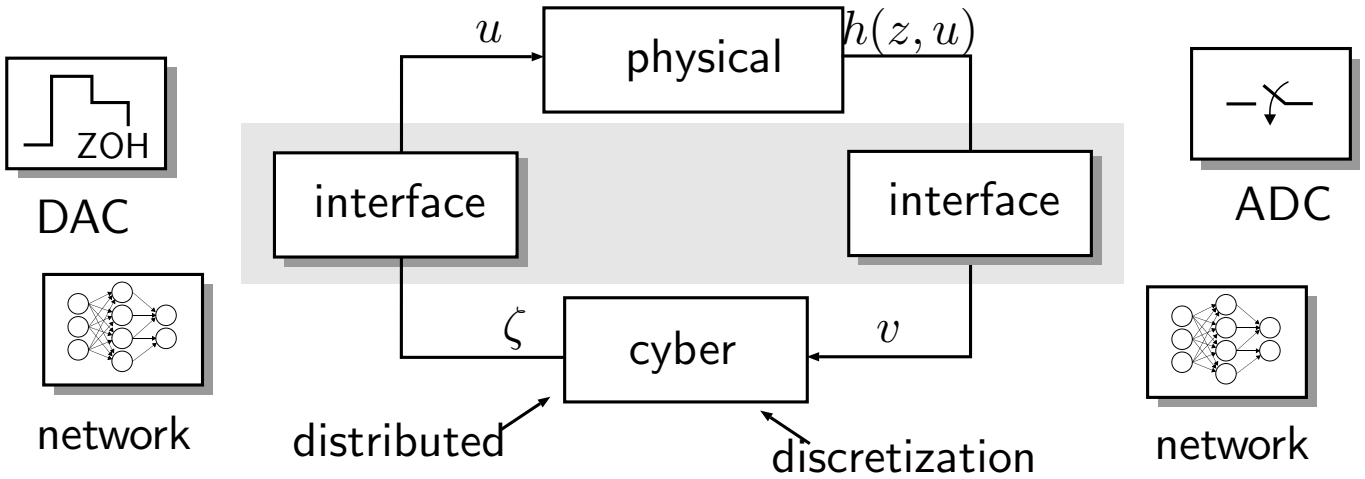


Figure : **Cyber-physical systems:** feedback interconnections between a plant (part of the physical component), controller (part of the cyber component), and interfaces/converters/signal conditioners (part of both the physical and cyber components).

An Full Model of CPSs

A full model of a CPSs in terms of a hybrid inclusion has state $x = (z, \eta, \lambda)$, input $\gamma = (u, v, w)$, and dynamics

$$\dot{x} \in F(x, \gamma) \qquad (x, \gamma) \in C \tag{1}$$

$$x^+ \in G(x, \gamma) \qquad (x, \gamma) \in D \tag{2}$$

- The map F is the *flow map*

An Full Model of CPSs

A full model of a CPSs in terms of a hybrid inclusion has state $x = (z, \eta, \lambda)$, input $\gamma = (u, v, w)$, and dynamics

$$\dot{x} \in F(x, \gamma) \qquad (x, \gamma) \in C \tag{1}$$

$$x^+ \in G(x, \gamma) \qquad (x, \gamma) \in D \tag{2}$$

- The map F is the *flow map*
- The set C is the *flow set*

An Full Model of CPSs

A full model of a CPSs in terms of a hybrid inclusion has state $x = (z, \eta, \lambda)$, input $\gamma = (u, v, w)$, and dynamics

$$\dot{x} \in F(x, \gamma) \qquad (x, \gamma) \in C \tag{1}$$

$$x^+ \in G(x, \gamma) \qquad (x, \gamma) \in D \tag{2}$$

- The map F is the *flow map*
- The set C is the *flow set*
- The map G is the *jump map*

An Full Model of CPSs

A full model of a CPSs in terms of a hybrid inclusion has state $x = (z, \eta, \lambda)$, input $\gamma = (u, v, w)$, and dynamics

$$\dot{x} \in F(x, \gamma) \qquad (x, \gamma) \in C \tag{1}$$

$$x^+ \in G(x, \gamma) \qquad (x, \gamma) \in D \tag{2}$$

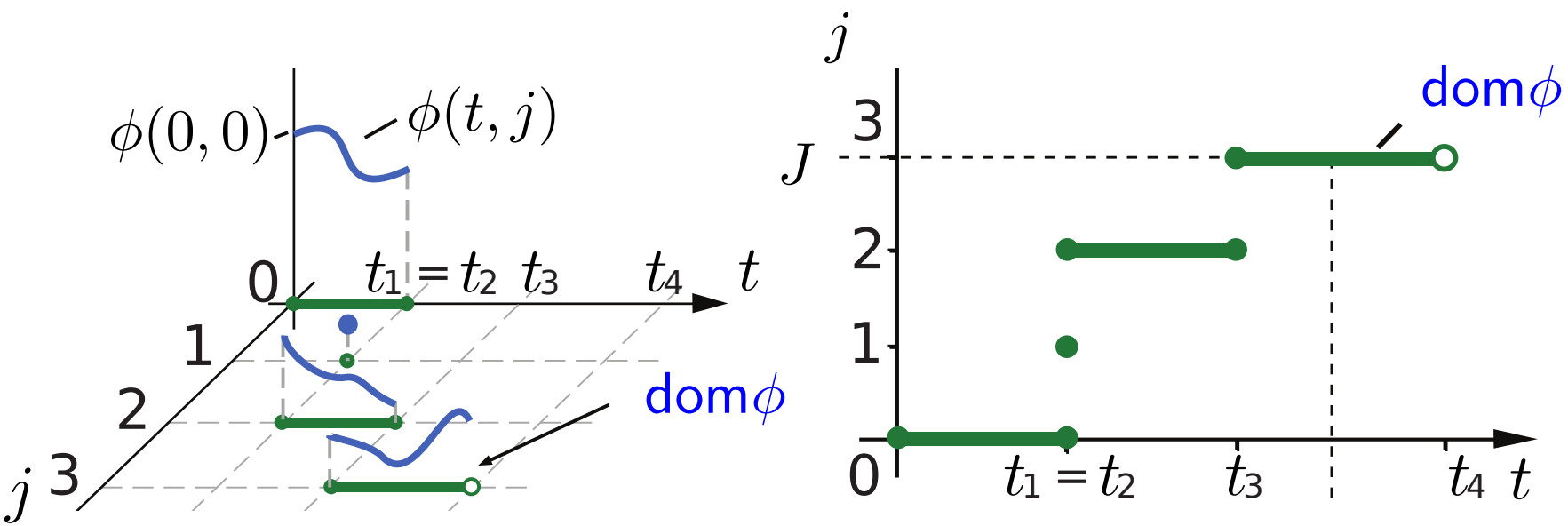
- The map F is the *flow map*
- The set C is the *flow set*
- The map G is the *jump map*
- The set D is the *jump set*

Solutions to CPSs

Hybrid time:

- Flows parameterized by $t \in \mathbb{R}_{\geq 0} := [0, +\infty)$.

A solution ϕ to $\mathcal{H}$ is a function given on hybrid time domains, called hybrid arc:

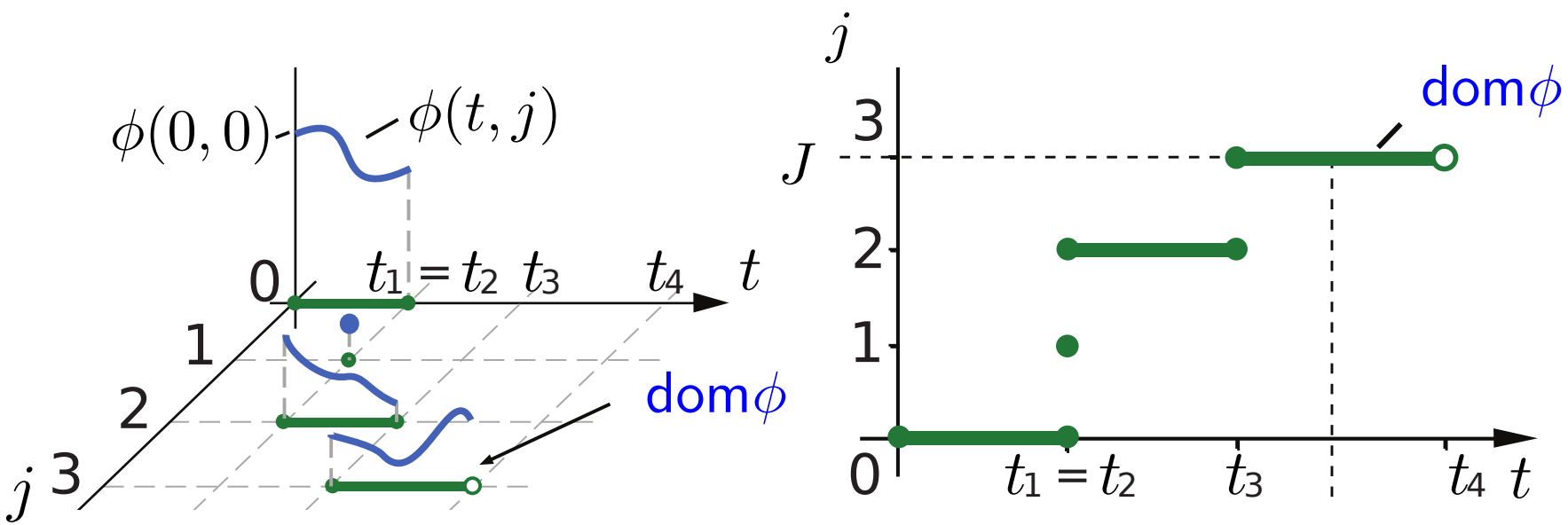


Solutions to CPSs

Hybrid time:

- Flows parameterized by $t \in \mathbb{R}_{\geq 0} := [0, +\infty)$.
- Jumps parameterized by $j \in \mathbb{N}_{\geq 0} := \{0, 1, 2, \ldots\}$.

A solution ϕ to $\mathcal{H}$ is a function given on hybrid time domains, called hybrid arc:



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.

- Conversion occurs at isolated time instants t_i

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

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

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.

- Conversion occurs at isolated time instants t_i
- Results in a cascade of two systems, an analog-to-digital converter drives the FSM

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

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

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.

- Conversion occurs at isolated time instants t_i
- Results in a cascade of two systems, an analog-to-digital converter drives the FSM
- Assume there is no memory state m_s in the converter

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

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

Simulating CPSs

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.

Simulating CPSs

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.

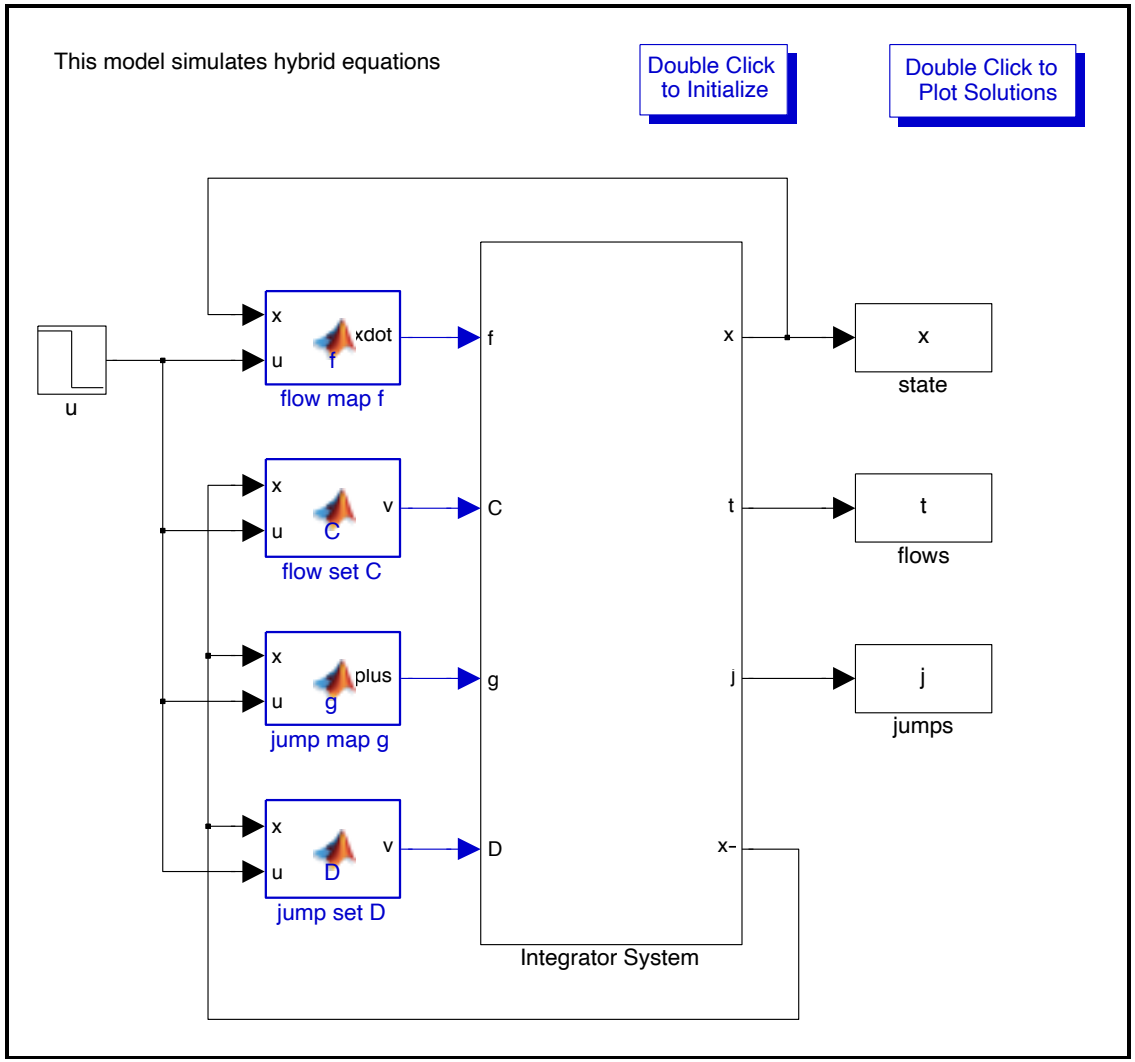
- Available tools in literature, Modelica, Ptolemy, Charon, HYSDEL, HyVisual

Simulating CPSs

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.

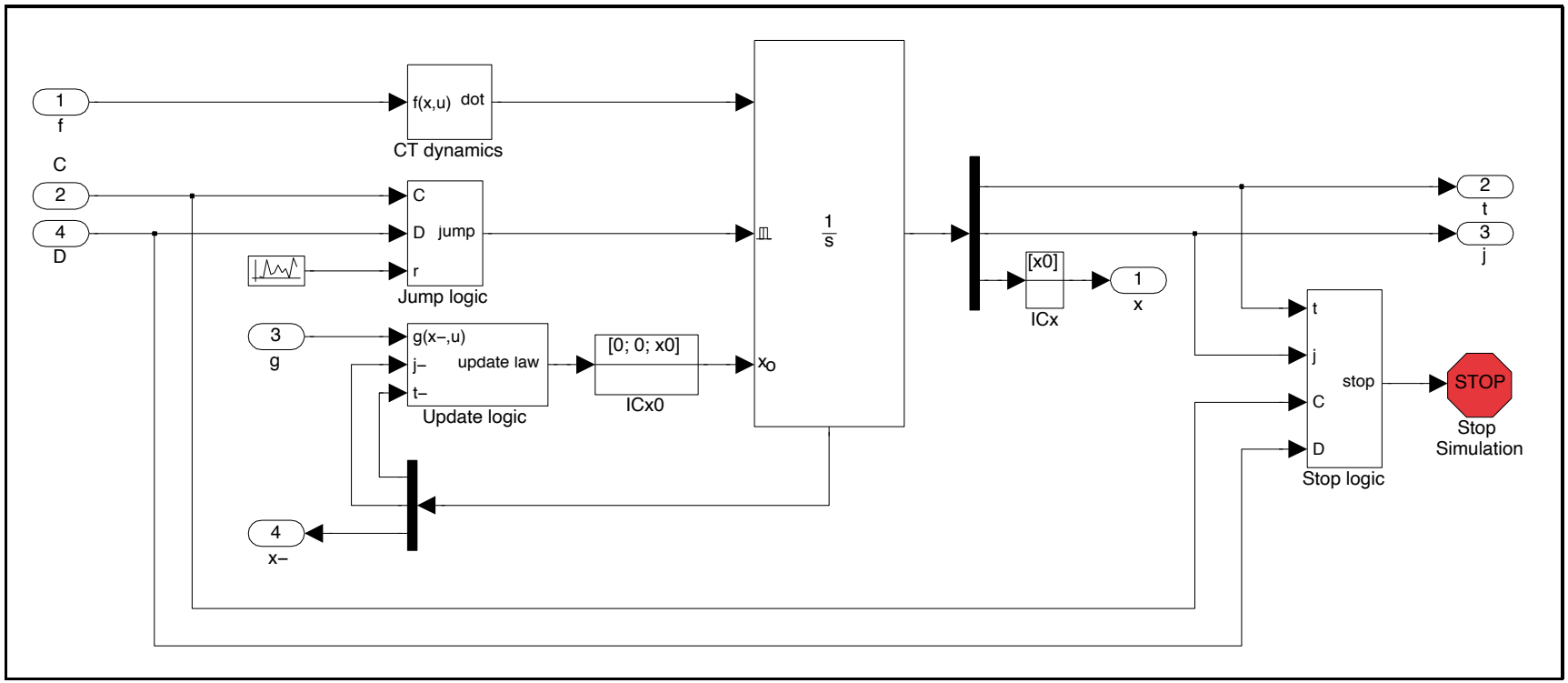
- Available tools in literature, Modelica, Ptolemy, Charon, HYSDEL, HyVisual
- A recent software package developed for the simulation of hybrid equations within Matlab/Simulink, namely, the Hybrid Equations (HyEQ) Toolbox

Simulating CPSs



Matlab/Simulink implementation of a hybrid inclusion with data (C, f, D, g) with inputs.

Simulating CPSs



Internals of integrator system.

Suggested Activities and Next Steps

- Review modeling slides

Suggested Activities and Next Steps

- Review modeling slides
- Complete Worksheets # 1 and # 2 (Quiz #2 tomorrow)

Suggested Activities and Next Steps

- Review modeling slides
- Complete Worksheets # 1 and # 2 (Quiz #2 tomorrow)
- Read reference [34] at https://hybrid.soe.ucsc.edu/pubs

Suggested Activities and Next Steps

- Review modeling slides
- Complete Worksheets # 1 and # 2 (Quiz #2 tomorrow)
- Read reference [34] at https://hybrid.soe.ucsc.edu/pubs
- Download Hybrid Equations Toolbox and simulate problems in Worksheets

Suggested Activities and Next Steps

- Review modeling slides
- Complete Worksheets # 1 and # 2 (Quiz #2 tomorrow)
- Read reference [34] at https://hybrid.soe.ucsc.edu/pubs
- Download Hybrid Equations Toolbox and simulate problems in Worksheets
- Tomorrow 8:45-10:30: **Hybrid Systems Tools for CPSs**