## Embedded Systems 5




## Synchronous Composition

**Lee/Seshia**
**Section 6.2**

- Important semantic model for concurrent composition
- Here: composition of actors
- Foundation of Statecharts, Simulink, synchronous programming languages
  - Esterel
  - Lustre
  - Scade
- Idealistic view of concurrency, not adequate for distributed systems (Implicit assumption: presence of global clock and instant communication; requires broadcast mechanism)

## Goal: deterministic behavior

An important advantage of synchronous over asynchronous composition is that determinacy can be preserved.

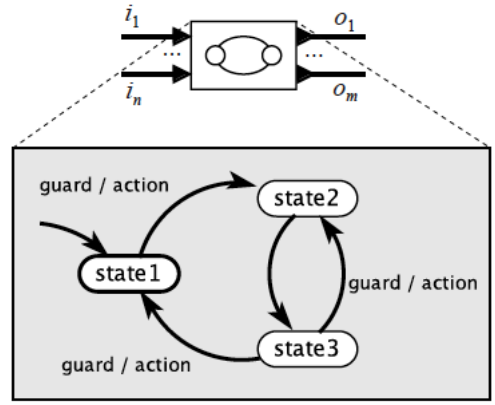
In the following, we'll assume that the individual actors are deterministic, and ensure that the composition remains deterministic.

For example, StateCharts are deterministic, if priority rules are introduced for transitions enabled at the same time (see, for example, the Stateflow semantics.)



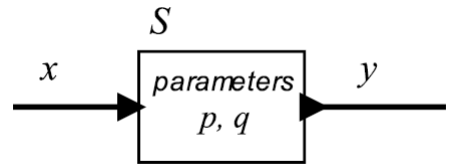
## REVIEW: Actor Model for State Machines

Expose inputs and outputs, enabling composition:



## REVIEW: Actor Model of Continuous-Time Systems

- A *system* is a function that accepts an input *signal* and yields an output signal.
- The domain and range of the system function are sets of signals, which themselves are functions.
- Parameters may affect the definition of the function *S*.



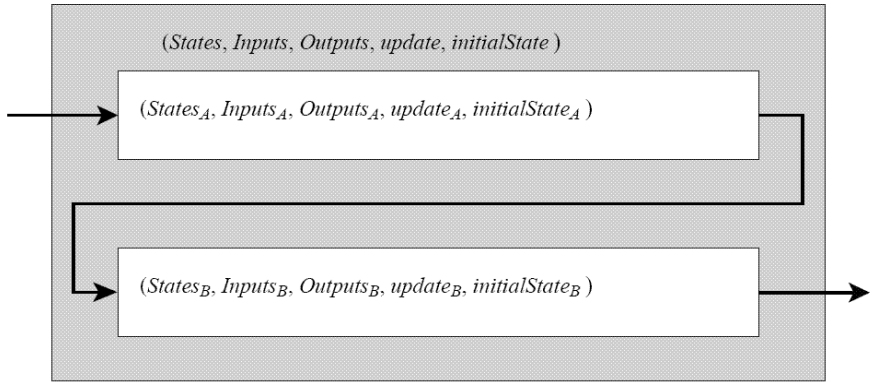

$$x: \mathbb{R} \to \mathbb{R}, \quad y: \mathbb{R} \to \mathbb{R}$$

$$S: X \to Y$$

$$X = Y = (\mathbb{R} \to \mathbb{R})$$

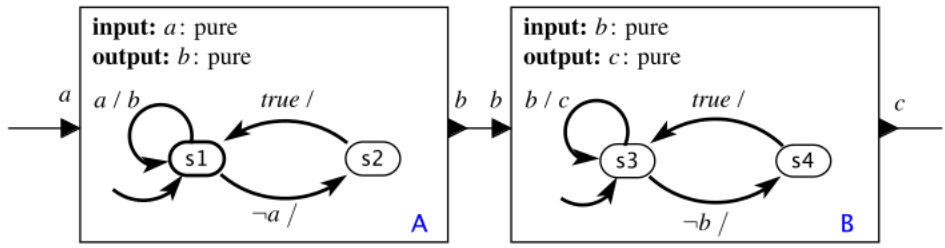


## Synchronous composition




Synchronous composition: the machines react simultaneously and instantaneously, despite the apparent causal relationship!

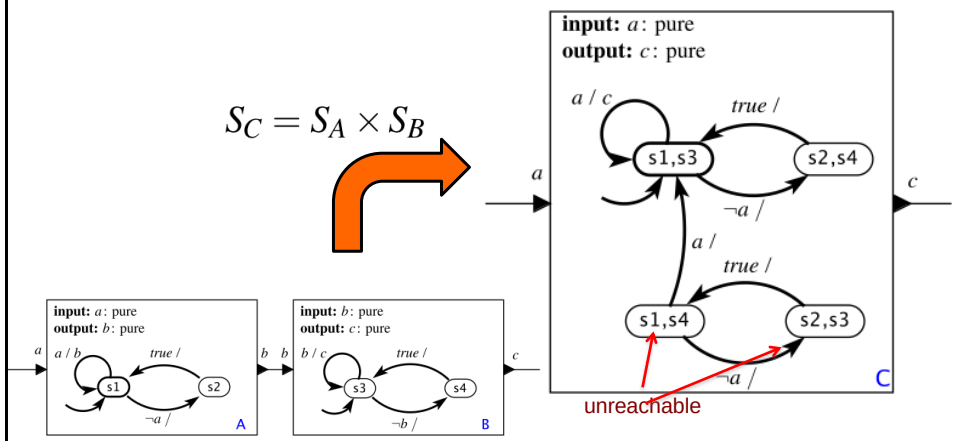
## Synchronous composition:
## Reactions are *simultaneous* and *instantaneous*



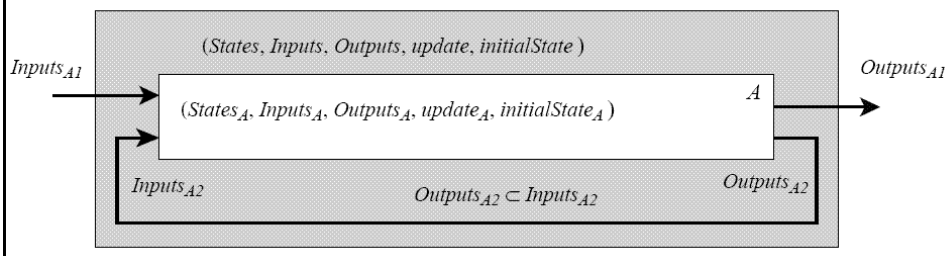



## Synchronous composition:
## Reactions are *simultaneous* and *instantaneous*

$S_C = S_A \times S_B$

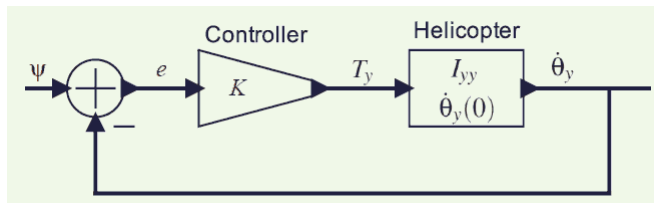


## Feedback composition






## Continuous feedback composition

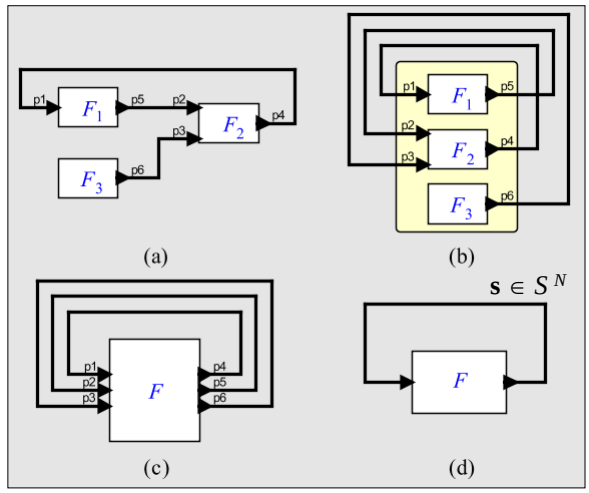



$$\dot{\theta}_y(t) = \dot{\theta}_y(0) + \frac{1}{I_{yy}} \int_0^t T_y(\tau) d\tau$$

$$= \dot{\theta}_y(0) + \frac{K}{I_{yy}} \int_0^t (\psi(\tau) - \dot{\theta}_y(\tau)) d\tau$$

Angular velocity appears on both sides. The semantics (meaning) of the model is the solution to this equation.

## Observation: Any Composition is a feedback composition



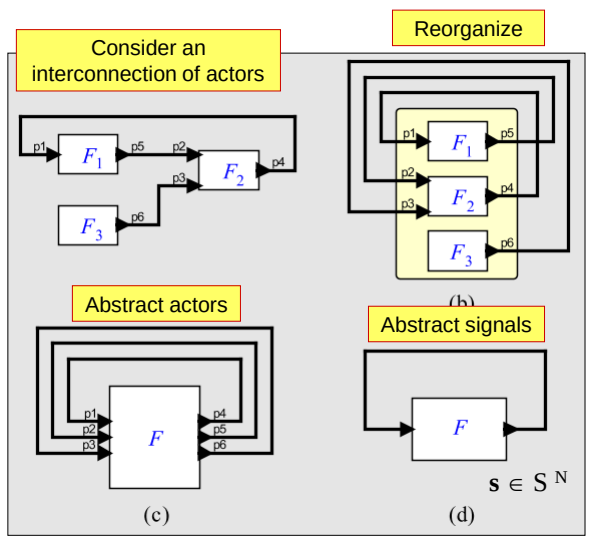

If every actor is a function then the semantics of the overall system is a

$\mathbf{s} \in S^N$ such that $F(\mathbf{s}) = \mathbf{s}$.

The behavior of the system is a "fixed point."



## Fixed point semantics




We seek an $\mathbf{s} \in S^N$ that satisfies $F(\mathbf{s}) = \mathbf{s}$.

Such an **s** is called a *fixed point*.

We would like the fixed point to exist and be unique. And we would like a constructive procedure to find it.

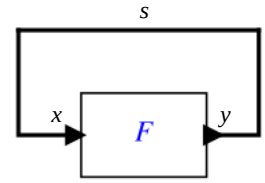
## Data types

As with any connection, we require compatible data types:

$$V_y \subseteq V_x$$

Then the signal on the feedback loop is a function

$$s: \mathbb{N} \to V_y \cup \{absent\}$$




Then we seek $s$ such that

$$F(s) = s$$

where $F$ is the actor function, which for determinate systems has form

$$F: (\mathbb{N} \to V_x \cup \{absent\}) \to (\mathbb{N} \to V_y \cup \{absent\})$$



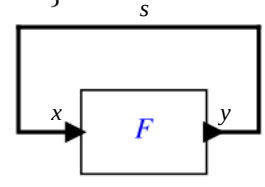
## Firing functions

With synchronous composition of determinate state machines, we can break this down by reaction. At the $n$-th reaction, there is a (state-dependent) function

$$f(n): V_x \cup \{absent\} \to V_y \cup \{absent\}$$

such that

$$s(n) = (f(n))(s(n))$$



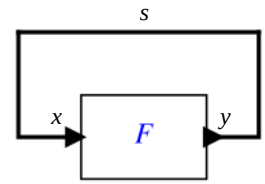

This too is a fixed point.

## Well-formed feedback

At the $n$-th reaction, we seek $s(n) \in V_y \cup \{absent\}$ such that

$$s(n) = (f(n))(s(n))$$

There are two potential problems:



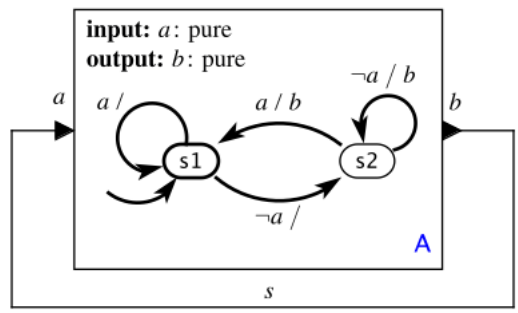

1. It does not exist.
2. It is not unique.

In either case, we call the system **ill formed**. Otherwise, it is **well formed**.

Note that if a state is not reachable, then it is irrelevant to determining whether the machine is well formed.

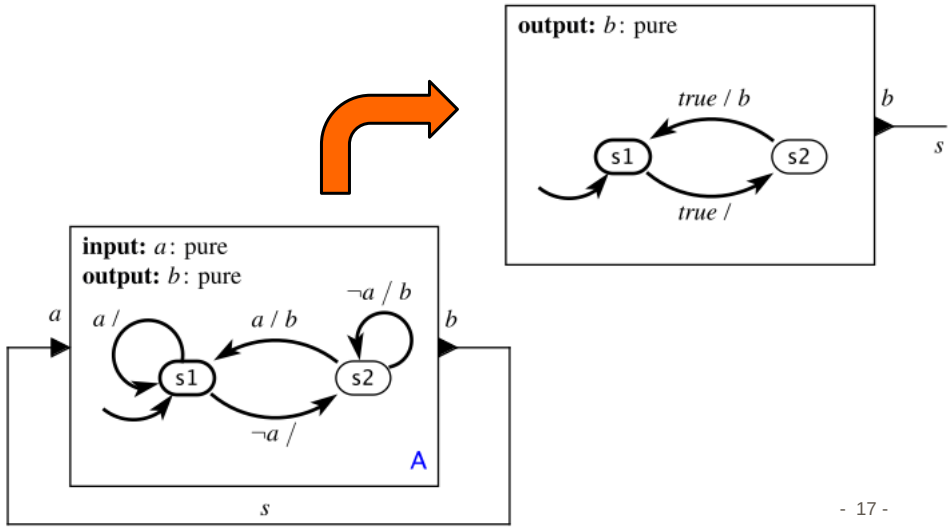


## Well-formed example



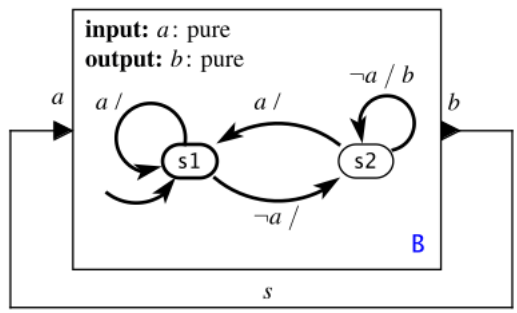

In state s1, we get the unique $s(n) = absent$.
In state s2, we get the unique $s(n) = present$.
Therefore, $s$ alternates between *absent* and *present*.

## Composite machine



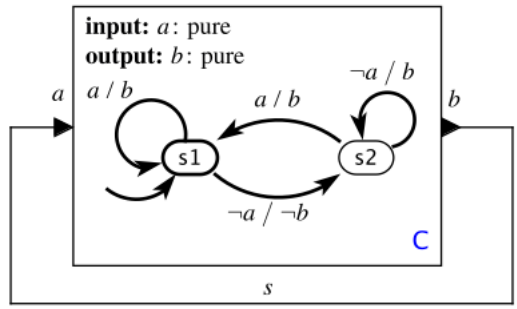



## Ill-Formed Example 1 (Existence)




In state s1, we get the unique $s(n) = absent$.
In state s2, there is no fixed point.
Since state s2 is reachable, this composition is ill formed.

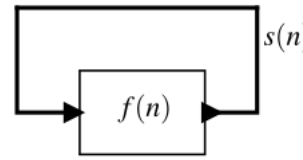
## Ill-Formed Example 2 (Uniqueness)




In s1, both $s(n) = absent$ and $s(n) = present$ are fixed points.
In state s2, we get the unique $s(n) = present$.
Since state s1 is reachable, this composition is ill formed.



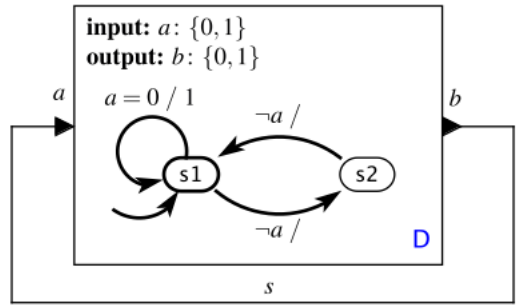
## Constructive Semantics: Single Signal




1. Start with $s(n)$ *unknown*.
2. Determine as much as you can about $(f(n))(s(n))$.
3. If $s(n)$ becomes known (whether it is present, and if it is not pure, what its value is), then we have a unique fixed point.

A state machine for which this procedure works is said to be **constructive**.

## Non-Constructive Well-Formed State Machine

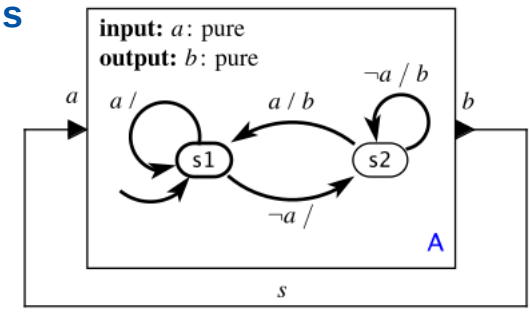



In state s1, if the input is unknown, we cannot immediately tell what the output will be. We have to try all the possible values for the input to determine that in fact $s(n) = absent$ for all $n$.

For non-constructive machines, we are forced to do **exhaustive search**. This is only possible if the data types are finite, and is only practical if the data types are small.



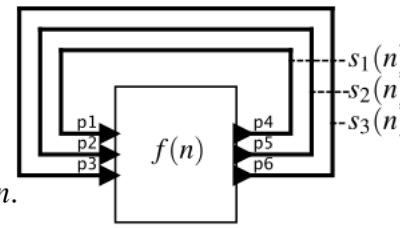
## Must / May Analysis




For the above constructive machine, in state s1, we can immediately determine that the machine *may not* produce an output. Therefore, we can immediately conclude that the output is *absent*, even though the input is unknown.

In state s2, we can immediately determine that the machine *must* produce an output, so we can immediately conclude that the output is *present*.

## Constructive Semantics: Multiple Signals

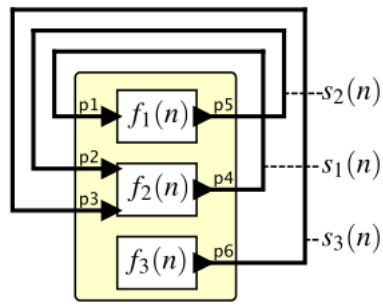



1. Start with $s_1(n), \cdots, s_N(n)$ *unknown*.
2. Determine as much as you can about $(f(n))(s_1(n), \cdots, s_N(n))$.
3. Using new information about $s_1(n), \cdots, s_N(n)$, repeat step (2) until no information is obtained.
4. If $s_1(n), \cdots, s_N(n)$ all become known, then we have a unique fixed point and a constructive machine.

A state machine for which this procedure works is said to be **constructive**.

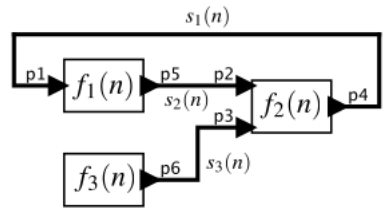


## Constructive Semantics: Multiple Actors




- Procedure is the same.

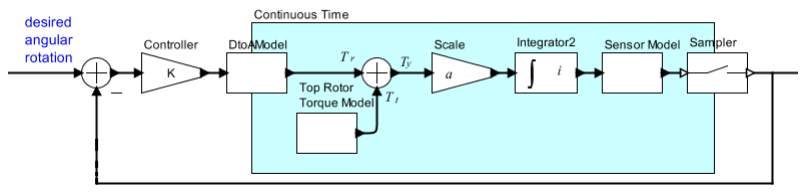
## Constructive Semantics: Arbitrary Structure




- Procedure is the same.
- A state machine language with constructive semantics will reject all compositions that in any iteration fail to make all signals known.
- Such a language rejects some well-formed compositions.

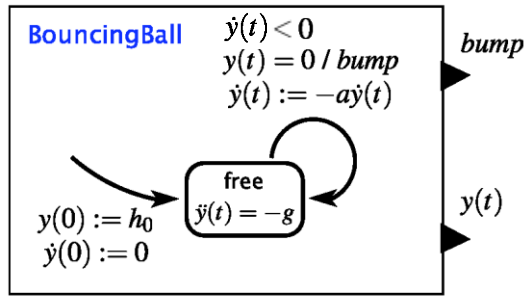


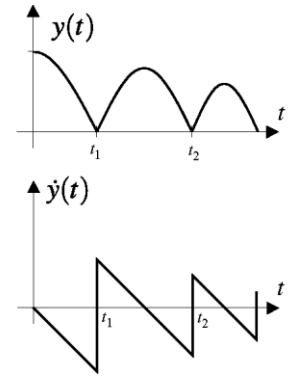
## Mixed Signal Models




The signals inside the blue area are continuous-time signals, and the ones outside are discrete-time signals.

## REVIEW: Hybrid Automaton for Bouncing Ball




$y$ – vertical distance from ground (position)
$a$ – coefficient of restitution, $0 < a < 1$



## Hybrid Automata

- Q: set of modes
- S: set of state variables, partitioned into
  - C={$c_1$, $c_2$, ..., $c_n$}: continuous signals (with range $\Re$)
  - D={$d_1$, $d_2$, ..., $d_m$}: discrete signals (with range {absent} $\cup$ X)
- U={$u_1$, $u_2$, ..., $u_k$}: set of input signals,
- Init $\subseteq$ Q $\times$ $\Re^n$ $\times$ ({absent} $\cup$ X)$^m$ : initial condition
- F: flows, defining differential equations for each continuous state variable in each mode
- J: Q $\times$ Guards $\rightarrow$ Q $\times$ Resets: jumps, where Guards is a constraint over C and U and Resets is a set of assignments of the form $x_i$ := expr(X,U) for the state variables

Note: our definitions follow Lee/Seshia, there are several other definitions of hybrid automata

## Hybrid Time Set

A hybrid time set is a finite or infinite sequence of intervals $\tau = \{I_i\}_{i=0..N}$ such that

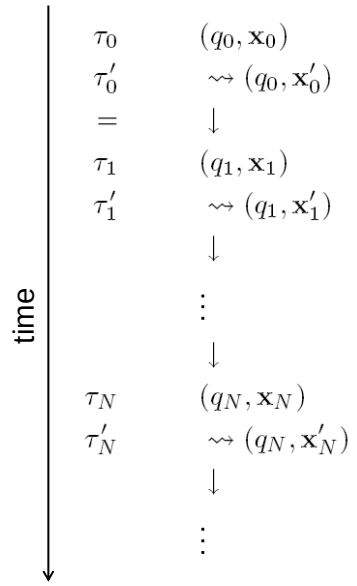
- $I_i = [\tau_i, \tau_i']$ for all $i < N$;
- If $N < \infty$ then either $I_N = [\tau_N, \tau_N']$ or $I_N = [\tau_N, \tau_N')$; and
- $\tau_i \leq \tau_i' = \tau_{i+1}$ for all $i$



## Hybrid Trajectory

- A hybrid trajectory $(\tau, q, x)$ consists of a hybrid time set $\tau$ and two sequences of functions
  - $q = \{q_i(\cdot): I_i \rightarrow Q\}_{i=0..N}$
  - $x = \{c_i(\cdot): I_i \rightarrow \Re\}_{i=0..N} \cup \{d_i(\cdot): I_i \rightarrow X \cup \{\text{absent}\}\}_{i=0..N}$

## Execution of a Hybrid Automaton

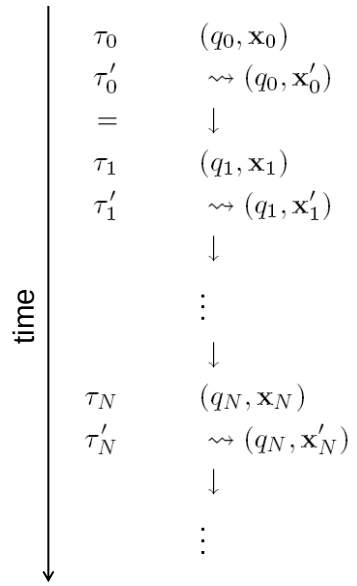



An **execution** of a hybrid automaton is a hybrid trajectory (τ, q, x) that statisfies the following conditions

- Initial condition: $(q_0, x_0) \in$ Init
- Discrete evolution: the pair $((q_i(\tau'_i), x_i(\tau'_i)), (q_{i+1}(\tau_{i+1}), x_i(\tau_{i+1}))$ satisfies J
- Continuous evolution: for all i,
  1. $q_i(\cdot)$ is constant over $I_i$
  2. $c_i(\cdot)$ is the solution to the differential equations in $F(q(\tau_i))$
  3. $d_i(\cdot)$ are absent during $(\tau_i, \tau'_i)$
  4. All jumps in J are disabled during $(\tau_i, \tau'_i)$



## Execution of a Hybrid Automaton




$$\tau = \tau_0, \tau_1, \tau_2, \ldots, \tau_N[, \ldots]$$

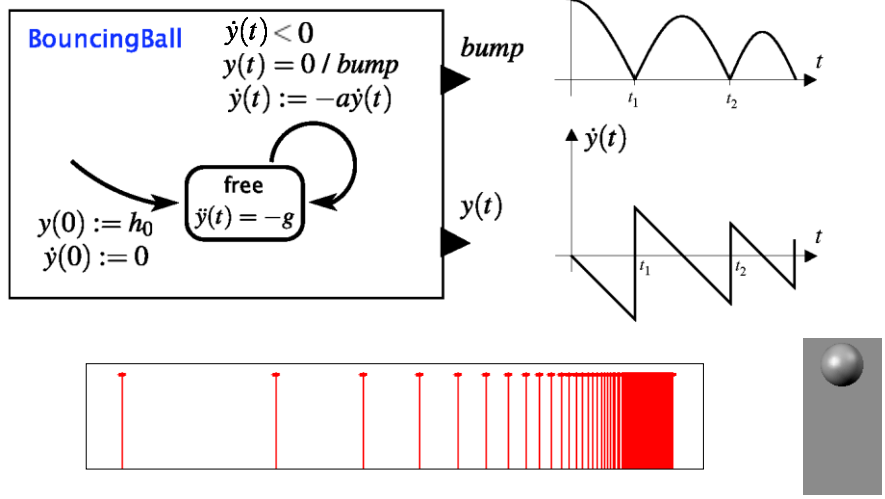
Continuous extent of $\tau$:

$$|\tau| = \sum_{i=0}^{\infty} \tau_{i+1} - \tau_i$$

Discrete extent of $\tau$:

$$\langle \tau \rangle = \begin{cases} N & \text{if } \tau \text{ is a finite sequence of length } N \\ \infty & \text{if } \tau \text{ is an infinite sequence} \end{cases}$$

## REVIEW: Hybrid Automaton for Bouncing Ball

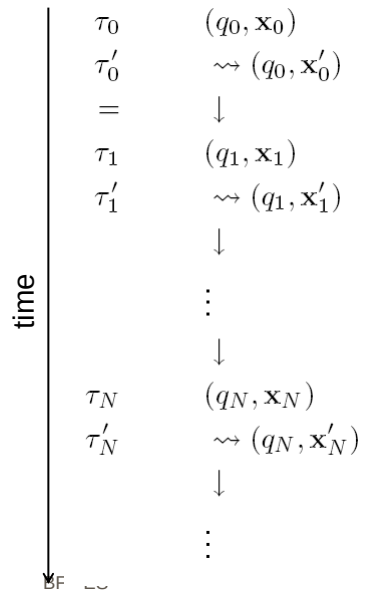








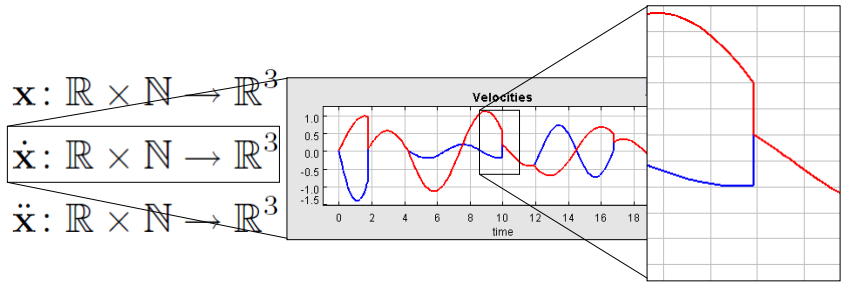

## Zeno Behavior




An execution of a hybrid automaton with time set $\tau$ is **zeno** iff $\langle \tau \rangle = \infty$ but $|\tau| < \infty$.

## Superdense Time

A signal can have a sequence of values at each (real) time.




At (real) time $t$, $x$ has a sequence of values

$$x(t,0), x(t,1), \cdots$$



## Initial and final value signals

Let $x: \mathbb{R} \times \mathbb{Z} \to \mathbb{R}^3$ be a CT signal. Define the *initial value signal* to be a function $x_i: \mathbb{R} \to \mathbb{R}$ where

$$x_i(t) = x(t,0)$$

Define the *final value signal* to be a function $x_f: \mathbb{R} \to \mathbb{R}$ where

$$x_f(t) = x(t,m)$$

where $m \in \mathbb{N}$ is the least value such that

$$\forall\, n > m, \quad x(t,n) = x(t,m).$$

If there is no such $m$ at any $t$, then the signal is said to be a *stuttering Zeno signal*.

## Simulation of continuous-time systems

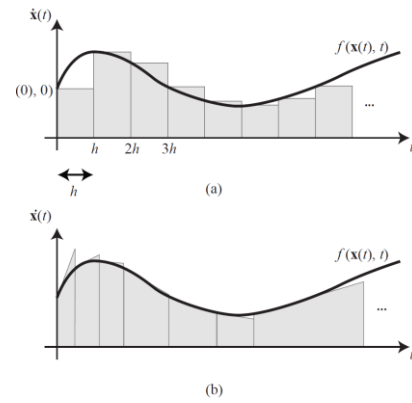
**Lee/Seshia**
**Section 6.4**

- The (numeric) simulator cannot directly deal with the time continuum, but can approximate it
- We consider equations of the form

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), t)$$

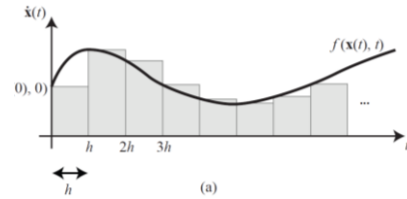
- An equivalent model is an integral equation

$$\begin{aligned} \mathbf{x}(t) &= \mathbf{x}(0) + \int_0^t \dot{\mathbf{x}}(\tau) d\tau \\ &= \mathbf{x}(0) + \int_0^t f(\mathbf{x}(\tau), \tau) d\tau \end{aligned}$$






## Forward Euler Solver




A forward Euler solver estimates the value of $\mathbf{x}$ at time points $0, h, 2h, 3h, \cdots$, where $h$ is called the **step size**. The integration is approximated as follows,

$$\begin{aligned} \mathbf{x}(h) &= \mathbf{x}(0) + hf(\mathbf{x}(0), 0) \\ \mathbf{x}(2h) &= \mathbf{x}(h) + hf(\mathbf{x}(h), h) \\ \mathbf{x}(3h) &= \mathbf{x}(2h) + hf(\mathbf{x}(2h), 2h) \\ &\cdots \\ \mathbf{x}((k+1)h) &= \mathbf{x}(kh) + hf(\mathbf{x}(kh), kh). \end{aligned}$$