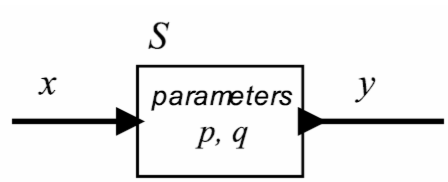
Introduction to Embedded Systems

Edward A. Lee & Sanjit Seshia
UC Berkeley
EECS 124
Spring 2008

Lecture 5: Actors and Dataflow

Recall: Actor Model of 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.

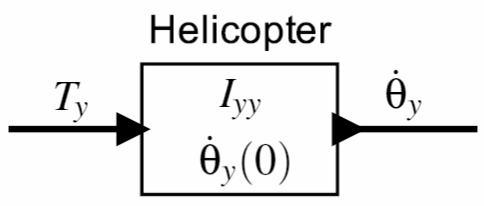
$$x: \mathbb{R} \to \mathbb{R}, \quad y: \mathbb{R} \to \mathbb{R}$$

$$S: X \to Y$$

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

Parameters may affect the definition of the function *S*.

Example: Actor model of the helicopter

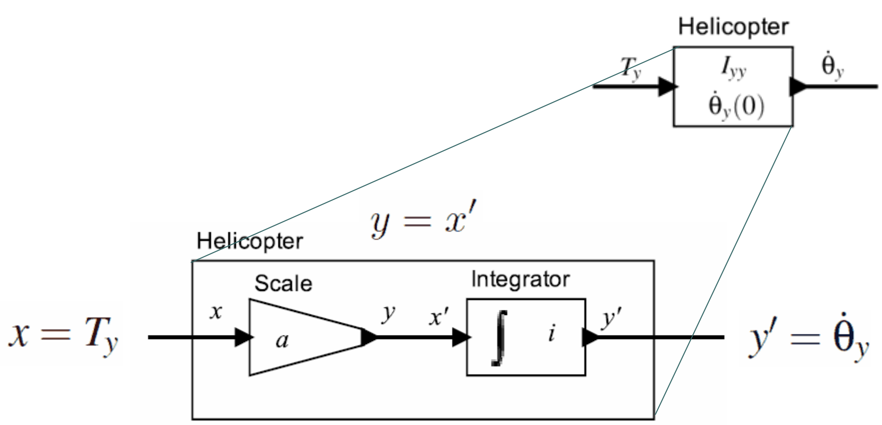


Input is the net torque of the tail rotor and the top rotor. Output is the angular velocity around the *y* axis.

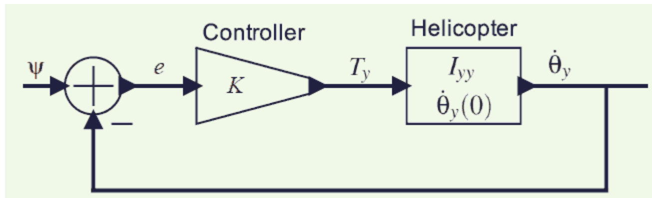
Parameters of the model are shown in the box. The input and output relation is given by the equation to the right.

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

Recall: Composition of actor models



Recall: Feedback Composition

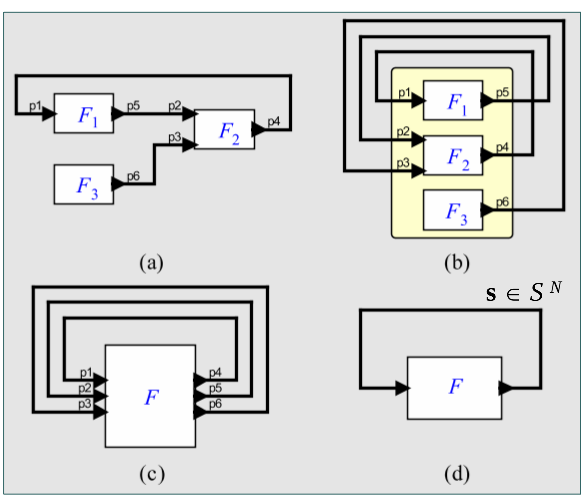


$$\dot{\theta}_y(t) = \dot{\theta}_y(0) + \frac{1}{I_{yy}} \int_0^t T_y(\tau) d\tau$$

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

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

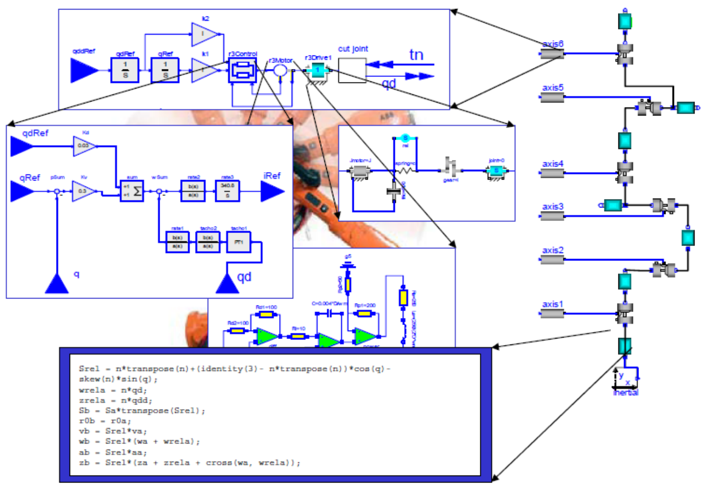


If every actor is a function, then the semantics of the overall system is the least $\mathbf{s} \in S^N$ such that $F(\mathbf{s}) = \mathbf{s}$.

The behavior of the system is a “fixed point.”

Actors are not the only way to do things

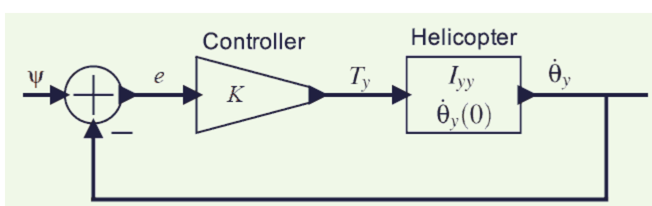
Imperative, Threads, Declarative physical models, Constraints, …



Modelica model of an industrial robot. Modelica uses Spice-like models where components have no inputs or outputs.

Courtesy of ABB Corp. Research and of Martin Otter, DLR

Model of Computation (MoC): Continuous Time (CT)



Structure of a signal:

$$x: T \to R$$

where in our helicopter model $T = R = \mathbb{R}$.
But signals can have a rather different form.

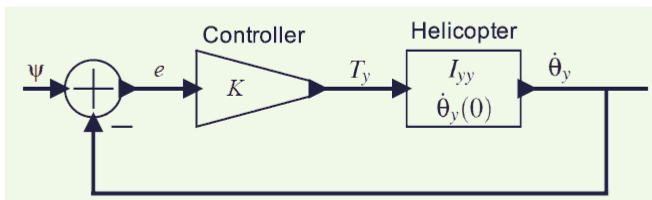
Discrete-Time (DT) Actor Models

Discrete-time signals have the form

$$x : \mathbb{Z} \to R$$

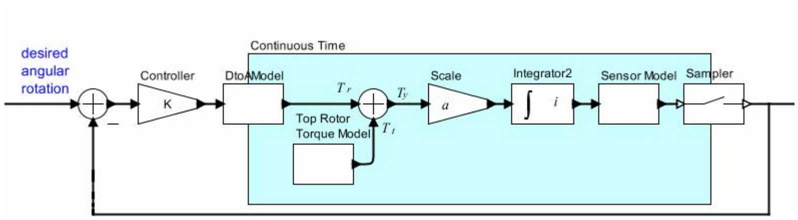
for some *data type* R, where $\mathbb{Z}$ is the set of integers. An index $n \in \mathbb{Z}$ is typically associated with a time value nT, where $T \in \mathbb{R}$ is the *sampling interval.*

Discrete-time helicopter model looks the same:



Mixed Signal Models

A more reasonable model of a discrete helicopter controller would be a mixed-signal model:



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

To jointly model discrete and continuous-time signals in the same MoC, augment the data type with "absent."

Let a continuous-time signal be a function of form

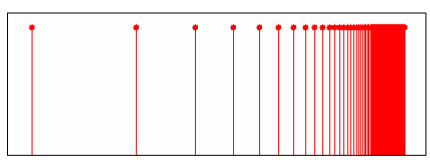
$$x\colon \mathbb{R} \to \mathbb{R} \cup \{\varepsilon\},$$

where ε denotes "absent." A discrete-time signal is now modeled as a CT signal whose value is ε except at times $t \in \mathbb{R}$ where $t = nT$, for some $n \in \mathbb{Z}$.

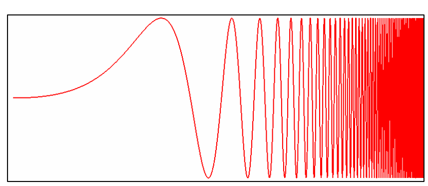
Now we can also model **discrete-event** (DE) systems, where the discrete events need not be regularly spaced.

Zeno systems, Stiff systems

DE systems can have Zeno conditions, where the number events in a finite time is bounded.



CT systems can be "stiff," where extremely fine time resolution (infinitely fine in the extreme) is required for part of the system.



But the standard model for continuous-time signals has major limitations

Consider the position, velocity, and acceleration of an object in three-dimensional space:

$$\mathbf{x} \colon \mathbb{R} \to \mathbb{R}^3$$

$$\dot{\mathbf{x}} \colon \mathbb{R} \to \mathbb{R}^3$$

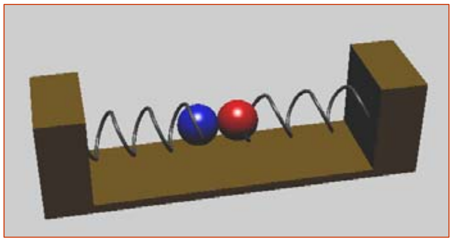
$$\ddot{\mathbf{x}} \colon \mathbb{R} \to \mathbb{R}^3$$

Such signals are *continuous* at $t \in \mathbb{R}$ if (e.g.):

$$\forall \epsilon > 0,\ \exists \delta > 0,\ \text{s.t.}\ \forall \tau \in (t-\delta, t+\delta), \quad ||\mathbf{x}(t) - \mathbf{x}(\tau)|| < \epsilon$$

Spring-mass system with collisions

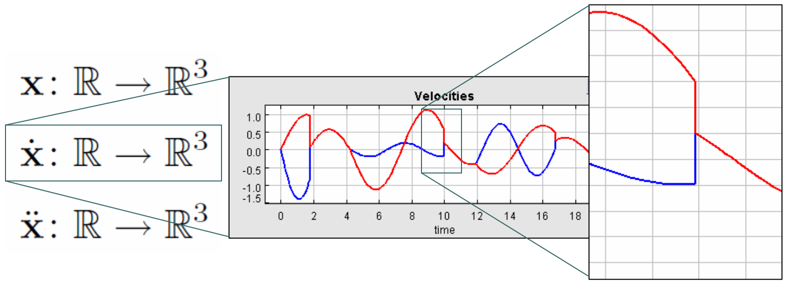
A model of a spring-mass system with collisions:



Consider the velocity of each mass. Is it continuous? What about the acceleration?

Piecewise Continuous Signals

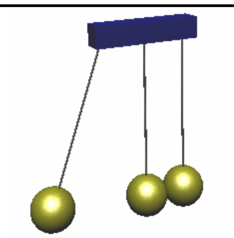
Many practical systems have signals with discontinuities.



Piecewise continuous signals are continuous at all $t \in \mathbb{R} \setminus D$ where $D \subset \mathbb{R}$ is a *discrete set*.[1]

[1]A set D with an order relation is a *discrete set* if there exists an order embedding to the integers.

A harder problem



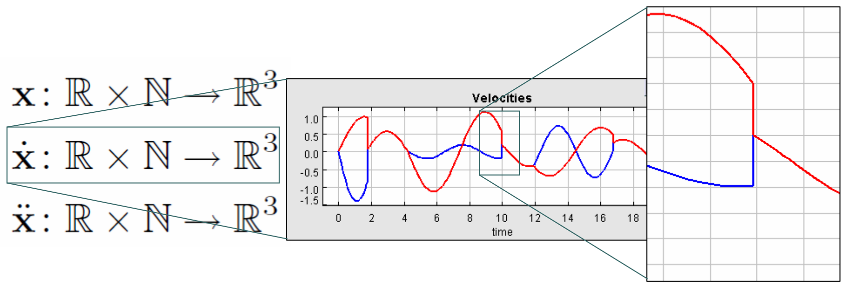
What if a signal takes on more than two values at a particular time?

Consider the momentum (velocity times mass) of the center ball of the Newton's Cradle system shown above.

Let $m\colon \mathbb{R} \to \mathbb{R}$ be the momentum of the middle ball. Let $\tau \in \mathbb{R}$ be the time at which the first ball collides with the second. What is the value of $x(\tau)$? No single answer adequately models the transfer of momentum from ball 1 to ball 3.

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

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

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

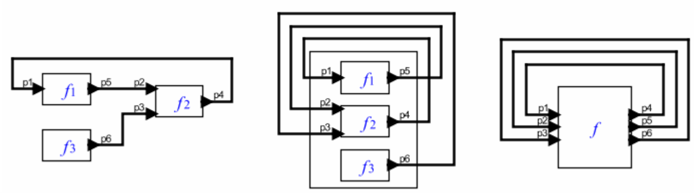
Piecewise continuity

A signal $x\colon \mathbb{R} \times \mathbb{Z} \to \mathbb{R}^3$ is piecewise continuous if at all $t \in \mathbb{R} \setminus D$, where D is a discrete set,

$$\forall\, m, n \in \mathbb{N}, \quad x(t, m) = x(t, n)$$

and its initial value signal is continuous from the left at all $t \in \mathbb{R}$, and its final value signal is continuous from the right at $t \in \mathbb{R}$.

Synchronous/Reactive (SR) models



A signal is a function of form $x\colon \mathbb{N} \to R \cup \{\varepsilon\}$ where the domain represents "ticks" of a "clock" and ε is "absent." An actor with n inputs and m outputs is

$$S\colon (\mathbb{N} \to R \cup \{\varepsilon\})^n \to (\mathbb{N} \to R \cup \{\varepsilon\})^m$$

At tick n of the clock, the actor realizes a function

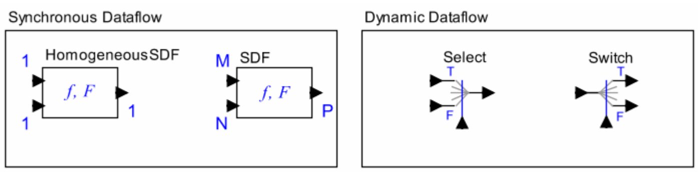
$$f_n\colon (R \cup \{\varepsilon\})^n \to (R \cup \{\varepsilon\})^m$$

At tick n, the system above has $m = n = 3$ and signal values $\mathbf{x}(n)$ satisfying $\mathbf{x}(n) = f_n(\mathbf{x}(n))$, a "fixed point."

Streams: The basis for Dataflow models

A stream is a signal $x\colon \mathbb{N} \to R$, for some set R. There is not necessarily any relationship between $x(n)$, an element in a stream, and $y(n)$, an element in another stream. Unlike discrete-time models or SR models, they are not "simultaneous."

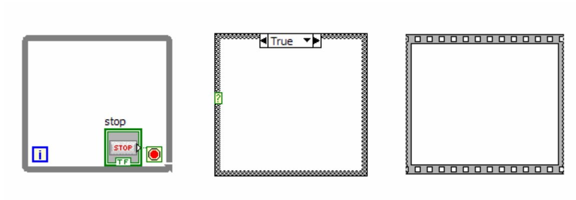
Dataflow



Each signal has form $x\colon \mathbb{N} \to R$. The function F maps such signals into such signals. The function f (the "firing function") maps prefixes of these signals into prefixes of the output. Operationally, the actor *consumes* some number of tokens and *produces* some number of tokens to construct the output signal(s) from the input signal(s). If the number of tokens consumed and produced is a constant over all firings, then the actor is called a *synchronous dataflow* (SDF) actor.

Firing rules: the number of tokens required to fire an actor.

Structured Dataflow



LabVIEW uses homogeneous SDF augmented with syntactically constrained forms of feedback and rate changes:

- While loops
- Conditionals
- Sequences

LabVIEW models are decidable.

EECS 124, UC Berkeley: 23

Many other concurrent MoCs have been explored

- (Kahn) process networks
- Communicating sequential processes (rendezvous)
- Time-driven models
- More dataflow variants:
  - cyclostatic
  - heterochronous
- Petri nets

Many of these have been combined with state machines to get *modal models*. That's next.

EECS 124, UC Berkeley: 24