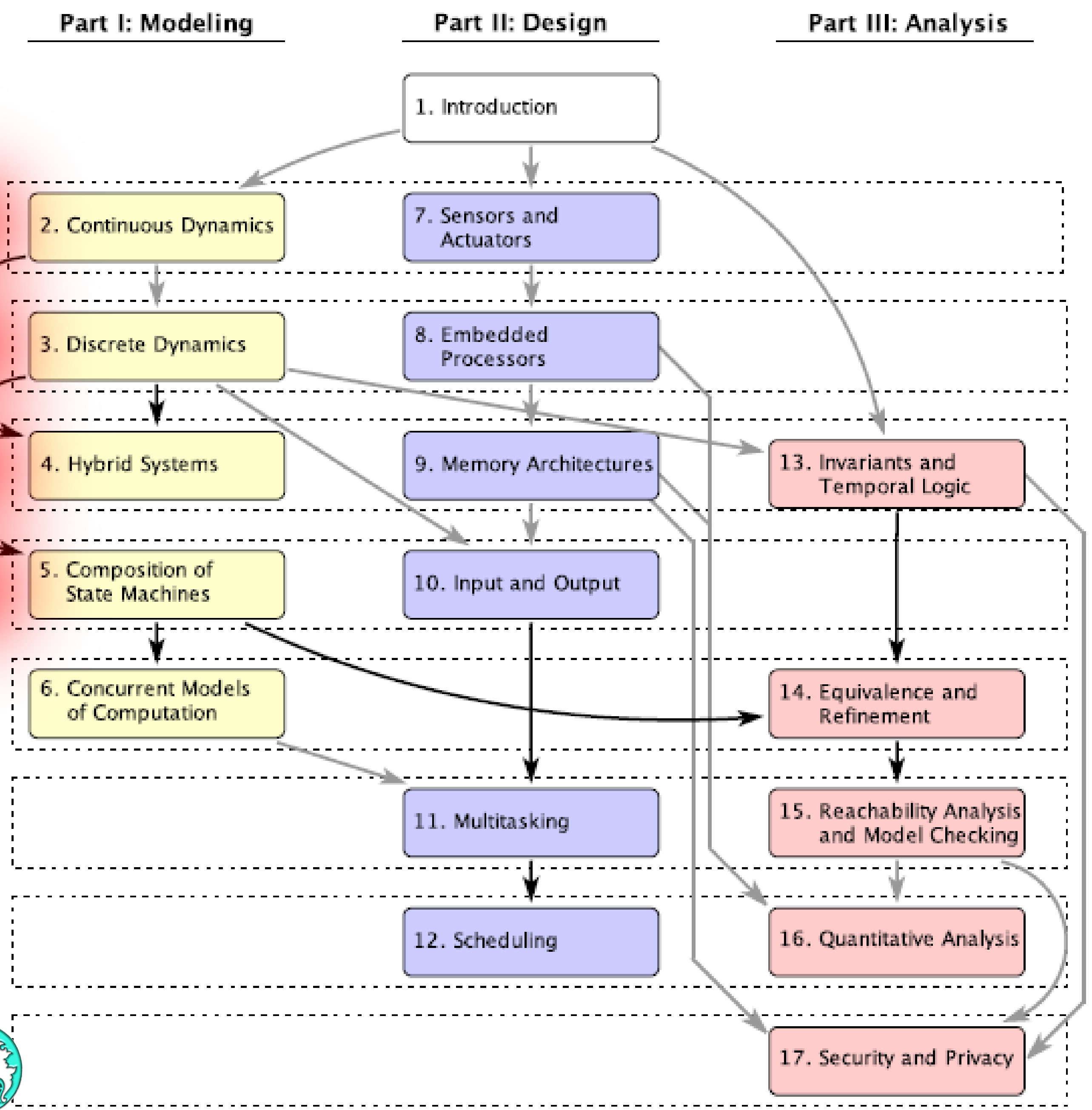
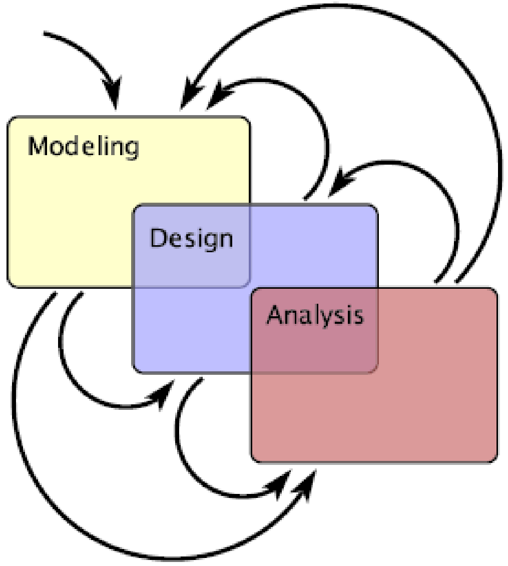
In the Name of GOD

modeling in embedded systems

By MohammadJavad Vaez

Supervisor: Professor Jamshidi

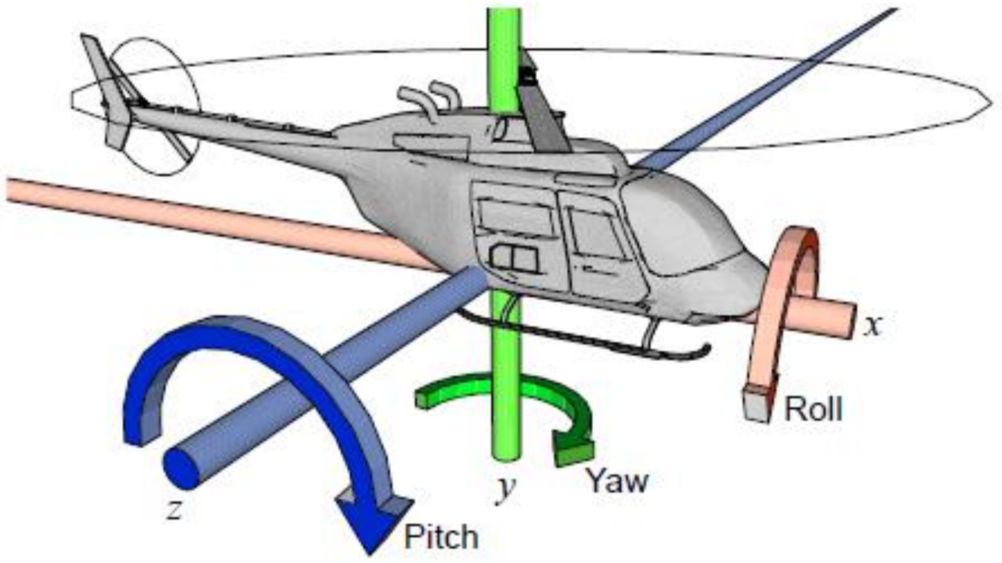




In this presentation, we will discuss chapters 2-5.

Chapter 2

- Continuous dynamics



Six degrees of freedom $(\underbrace{x, y, z}_{\text{vector x}}, \underbrace{\theta_x, \theta_y, \theta_z}_{\text{vector } \theta})$

Newton's second law:

$$F(t) = M\ddot{x}(t)$$

In fact

$$F(t) = \frac{dP}{dt} = \frac{d}{dt}\big(M(t)v(t)\big) = Ma(t)$$

We consider the mass is constant

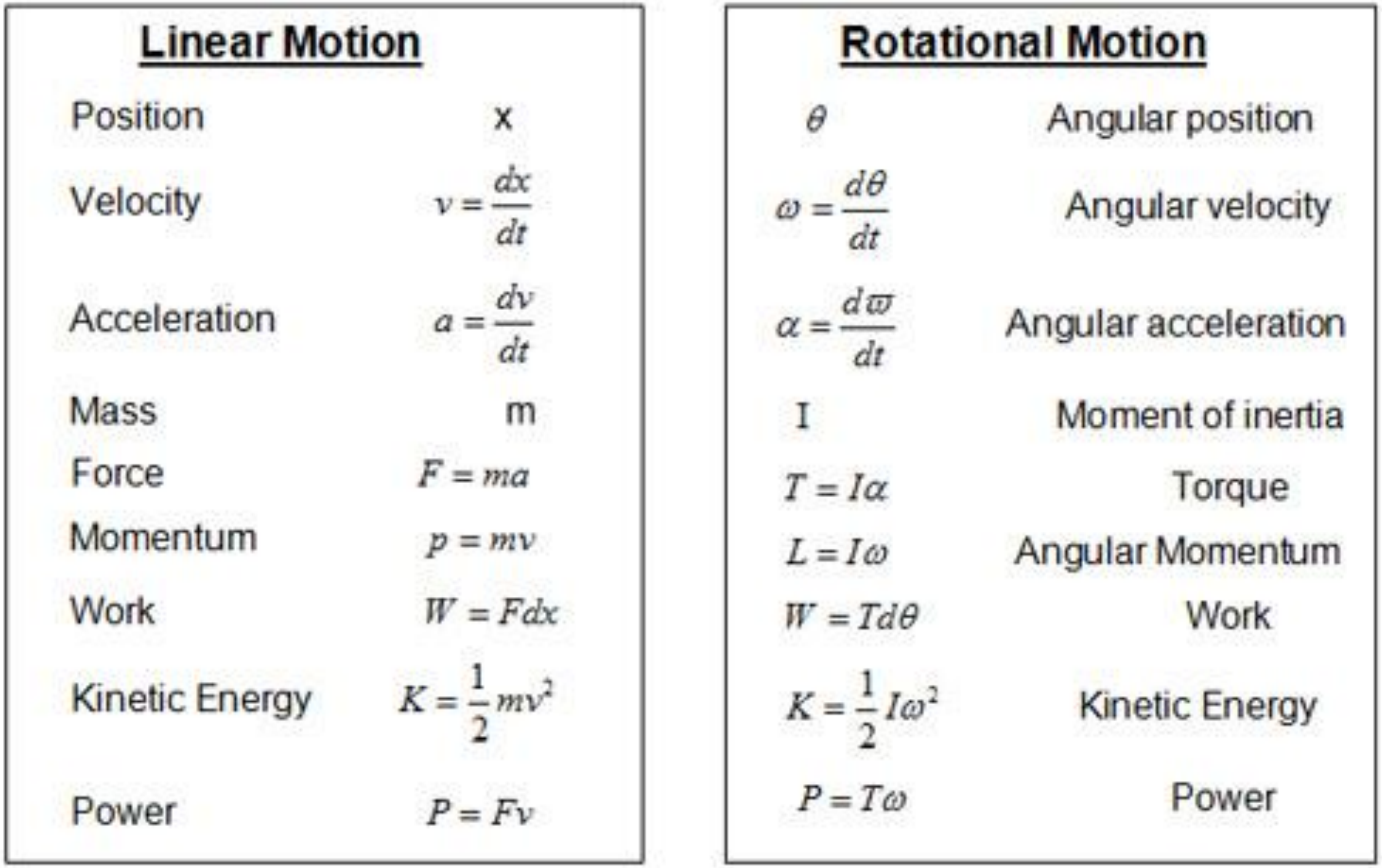
$$\forall\, t > 0, \quad \dot{\mathbf{x}}(t) \;=\; \dot{\mathbf{x}}(0) + \int_0^t \ddot{\mathbf{x}}(\tau)d\tau$$

where $\dot{\mathbf{x}}(0)$ is the initial velocity in three directions. Using (2.1), this becomes

$$\forall\, t > 0, \quad \dot{\mathbf{x}}(t) = \dot{\mathbf{x}}(0) + \frac{1}{M}\int_0^t \mathbf{F}(\tau)d\tau,$$

Position is the integral of velocity,

$$\begin{aligned} \mathbf{x}(t) &= \mathbf{x}(0) + \int_0^t \dot{\mathbf{x}}(\tau)d\tau \\ &= \mathbf{x}(0) + t\dot{\mathbf{x}}(0) + \frac{1}{M}\int_0^t \int_0^\tau \mathbf{F}(\alpha)d\alpha d\tau, \end{aligned}$$



Linear Motion	
Position	x
Velocity	$v = \frac{dx}{dt}$
Acceleration	$a = \frac{dv}{dt}$
Mass	m
Force	$F = ma$
Momentum	$p = mv$
Work	$W = Fdx$
Kinetic Energy	$K = \frac{1}{2}mv^2$
Power	$P = Fv$

Rotational Motion	
θ	Angular position
$\omega = \frac{d\theta}{dt}$	Angular velocity
$\alpha = \frac{d\varpi}{dt}$	Angular acceleration
I	Moment of inertia
$T = I\alpha$	Torque
$L = I\omega$	Angular Momentum
$W = Td\theta$	Work
$K = \frac{1}{2}I\omega^2$	Kinetic Energy
$P = T\omega$	Power

remembrance

Newton's second law:

$$T(t) = \frac{dL}{dt} = \frac{d}{dt}(I(t)\dot{\theta}(t))$$

$$\begin{pmatrix} L_x \\ L_y \\ L_z \end{pmatrix} = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

moment of inertia tensor

It depends on the geometry and orientation of the object.

It represents the reluctance that an object has to spin around any axis as a function of its orientation along the three axes.

$I_{yx}(t)$ is the inertia that determines how acceleration around the x axis is related to torque around the y axis.

If the object is spherical, for example, this reluctance is the same around all axes, so the moment of inertia reduces to a constant scalar I.

$$T(t) = \frac{d}{dt}\left(I(t)\dot{\theta}(t)\right) = I\ddot{\theta}(t)$$

Rotational velocity is the integral of acceleration,

$$\dot{\theta}(t) = \dot{\theta}(0) + \int_0^t \ddot{\theta}(\tau)d\tau,$$

where $\dot{\theta}(0)$ is the initial rotational velocity in three axes. For a spherical object, using (2.3), this becomes

$$\dot{\theta}(t) = \dot{\theta}(0) + \frac{1}{I}\int_0^t \mathbf{T}(\tau)d\tau.$$

Orientation is the integral of rotational velocity,

$$\begin{aligned}\theta(t) &= \theta(0) + \int_0^t \dot{\theta}(\tau)d\tau \\ &= \theta(0) + t\dot{\theta}(0) + \frac{1}{I}\int_0^t \int_0^\tau \mathbf{T}(\alpha)d\alpha d\tau\end{aligned}$$

$$x(t) = x(0) + t\dot{x}(t) + \frac{1}{M}\int_0^t\int_0^\tau F(\alpha)d\alpha d\tau$$

$$\theta(t) = \theta(0) + t\dot{\theta}(t) + \frac{1}{I}\int_0^t\int_0^\tau T(\alpha)d\alpha d\tau$$

A helicopter has two rotors, one above, which provides lift, and one on the tail. Without the rotor on the tail, the body of the helicopter would spin. The rotor on the tail counteracts that spin. Specifically, the force produced by the tail rotor must counter the torque produced by the main rotor. Here we consider that the helicopter position is fixed at the origin, so there is no need to consider equations describing position. Moreover, we assume that the helicopter remains vertical, so pitch and roll are fixed at zero. These assumptions are not as unrealistic as they may seem since we can define the coordinate system to be fixed to the helicopter.

We model the simplified helicopter by a system that takes as input a continuous-time signal T_y, the torque around the y axis (which causes changes in yaw). This torque is the sum of the torque caused by the main rotor and that caused by the tail rotor. When these are perfectly balanced, that sum is zero. The output of our system will be the angular velocity $\dot{\theta}_y$ around the y axis. The dimensionally-reduced version of (2.2) can be written as

$$\ddot{\theta}_y(t) = T_y(t)/I_{yy}.$$

Integrating both sides, we get the output $\dot{\theta}$ as a function of the input T_y,

$$\dot{\theta}_y(t) = \dot{\theta}_y(0) + \frac{1}{I_{yy}} \int_0^t T_y(\tau)d\tau. \tag{2.4}$$

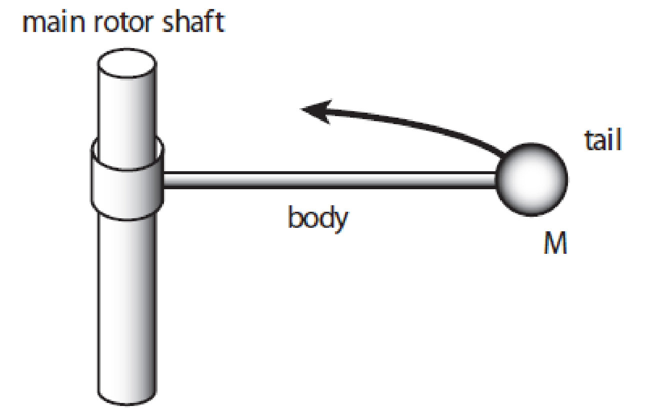
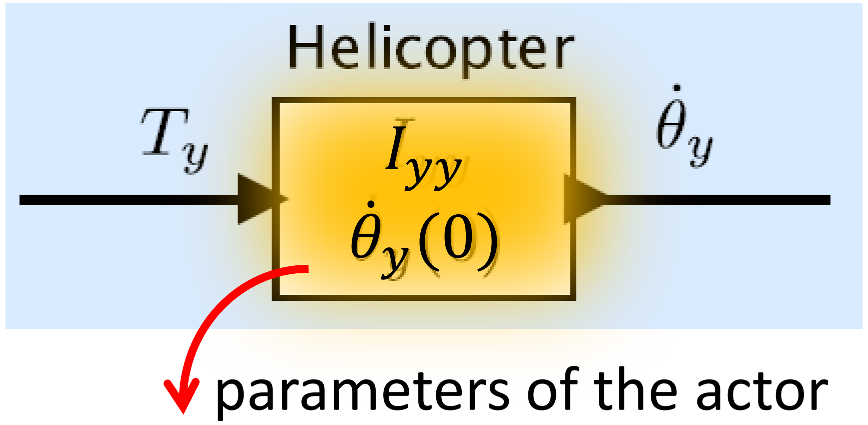
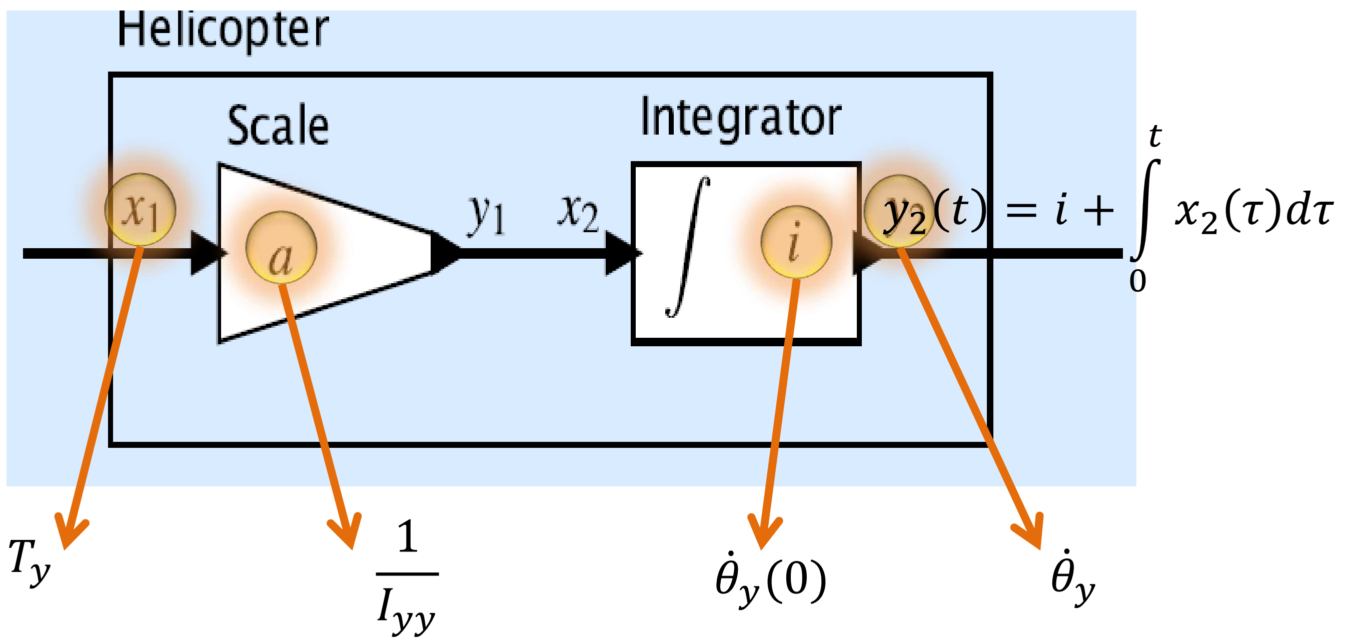


Figure 2.2: Simplified model of a helicopter.

We are familiar with the concept of a system from the lesson “Signals and Systems”. Now, we can depict the actor model for the helicopter as follows:



It can be represented as a cascade composition of two actors as follows:



We can also argue about different properties of a system. For example causality, to have or not to have memory, linearity, time invariance and stability.

The helicopter system defined by this equation

$\dot{\theta}_y = \dot{\theta}_y(0) + \frac{1}{I_{yy}} \int_0^t T_y(\tau)d\tau$ is not time invariant.

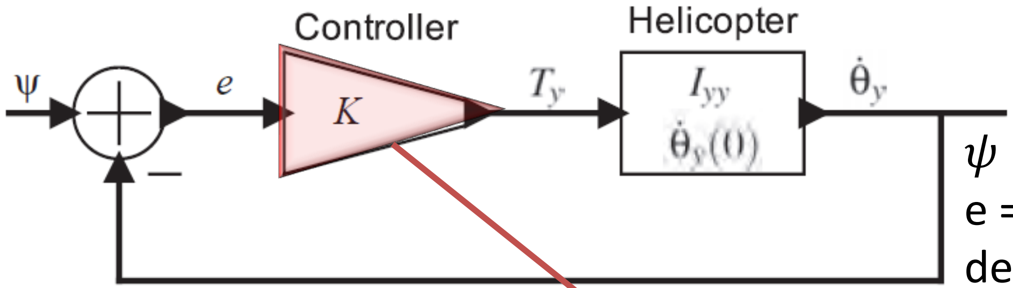
However, this equation describes a time invariant system:

$$\dot{\theta}_y = \frac{1}{I_{yy}} \int_{-\infty}^t T_y(\tau)d\tau$$

Since $\frac{1}{I_{yy}} \int_{-\infty}^t T_y(\tau-\alpha)d\tau = \frac{1}{I_{yy}} \int_{-\infty}^{t-\alpha} T_y(\tau)d\tau = \dot{\theta}_y(t-\alpha)$

The helicopter it's also unstable. Let the input be T_y = u (unit step). Then $\dot{\theta}_y$ grows without bound. **In practice, a helicopter uses a feedback system to determine how much torque to apply at the tail rotor to keep the body of the helicopter straight.**

We can stabilize the helicopter with a simple feedback control system, as shown below.



$\psi = disired\ angular\ velocity$
e = difference between the actual and the desired angular velocity

Proportional controller
When K is bigger, $\dot{\theta}_y$ converges faster to the desired output

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

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

An integral (or differential) equation
Assume $\psi(t) = 0$

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

derivation $\Rightarrow\ \ddot{\theta}_y(t) = \frac{-K}{I_{yy}}\dot{\theta}_y(t)$

Laplace $\Longrightarrow$ Transform

$$s\dot{\Theta}_y(s) - \dot{\theta}_y(0) = \frac{-K}{I_{yy}}\dot{\Theta}_y(s)$$

$$\Rightarrow \dot{\Theta}_y(s) = \frac{\dot{\theta}_y(0)}{s+\frac{K}{I_{yy}}}$$

$$\Rightarrow\ \dot{\theta}_y(t) = \dot{\theta}_y(0)e^{-\frac{Kt}{I_{yy}}}u(t)$$

Assume that the helicopter is initially at rest, i.e $\dot{\theta}(0) = 0$
and the desired signal is $\psi(t) = au(t)$
So we wish the helicopter to rotate at a fixed rate.

$$\dot{\theta}_y(\mathrm{t}) = \frac{1}{I_{yy}}\int_0^t T_y(\tau)d\tau = \frac{K}{I_{yy}}\int_0^t \left(\psi(\tau) - \dot{\theta}_y(\tau)\right)d\tau = \frac{K}{I_{yy}}\int_0^t a d\tau - \frac{K}{I_{yy}}\int_0^t \dot{\theta}_y(\tau)d\tau$$

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

Using same $\Rightarrow$ techniques

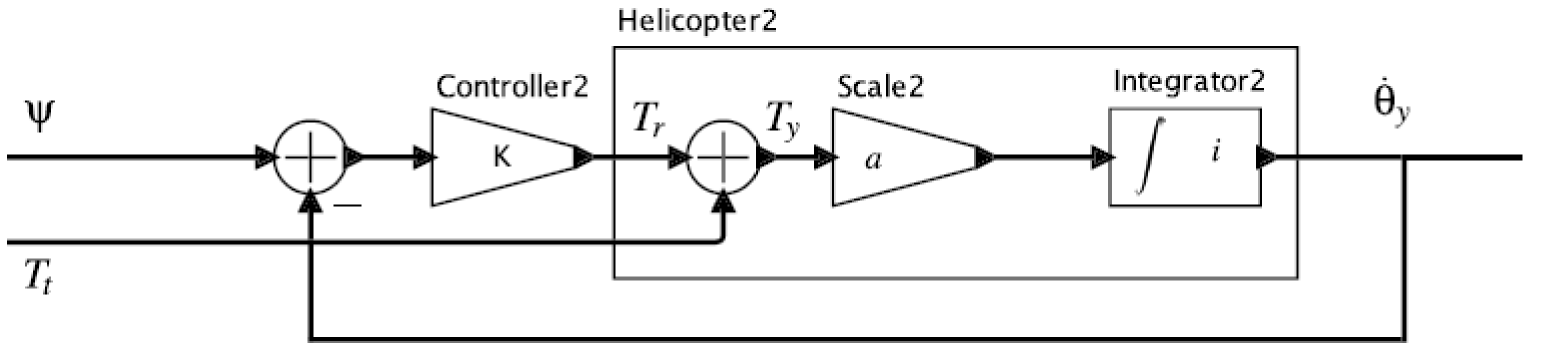
$$\boxed{\dot{\theta}_y(t) = au(t)(1 - e^{\frac{-Kt}{I_{yy}}})}$$

The above example is somewhat unrealistic because we cannot independently control the **net** torque of the helicopter. Actually

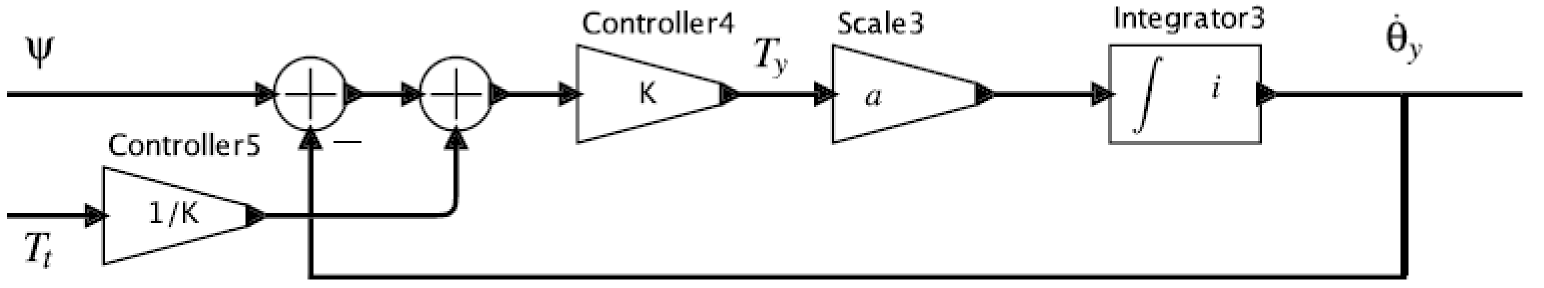
$$T_y(t) = \underbrace{T_t(t)}_{\text{Torque due to the top rotor}} + \underbrace{T_r(t)}_{\text{Torque due to the tail rotor}}$$

T_t will be determined by the rotation required to maintain or achieve a desired altitude. Thus, we will actually need to design a control system that controls T_r and stabilizes the helicopter for any T_t.

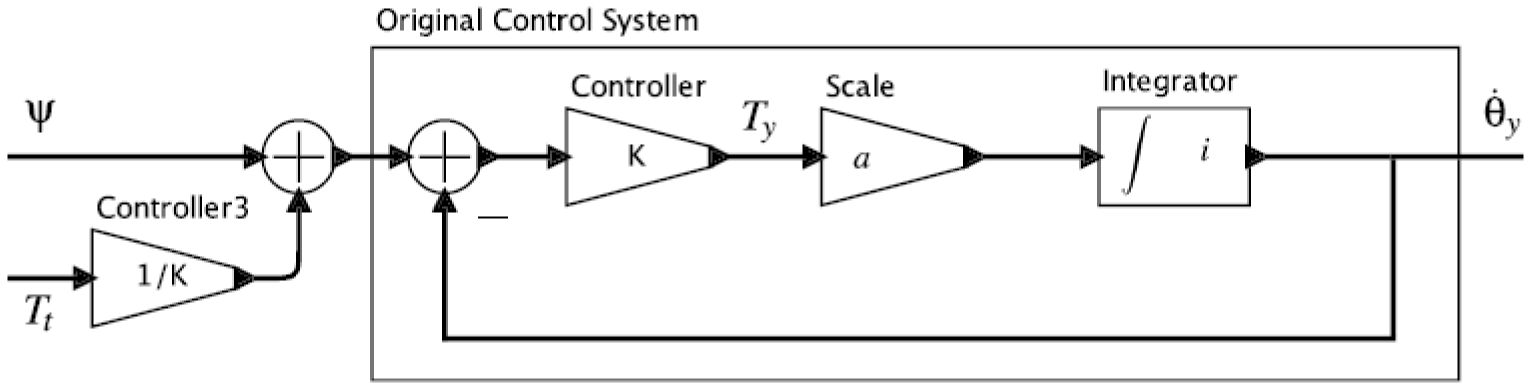
Suppose $T_t = bu(t)$ and the helicopter is initially at rest.



(a)

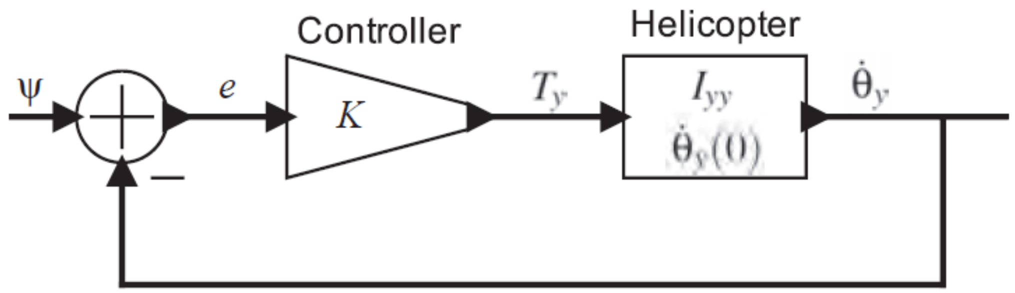


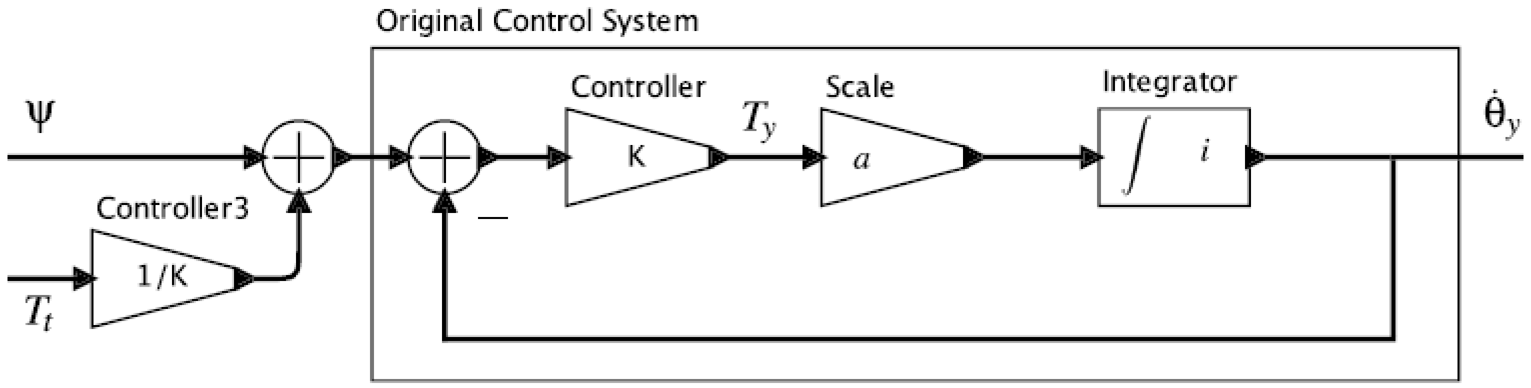
(b)



(c)

we see that the portion of the model enclosed in the box is exactly the same as the control system analyzed in Slide 12. Thus, the same analysis still applies.





(c)

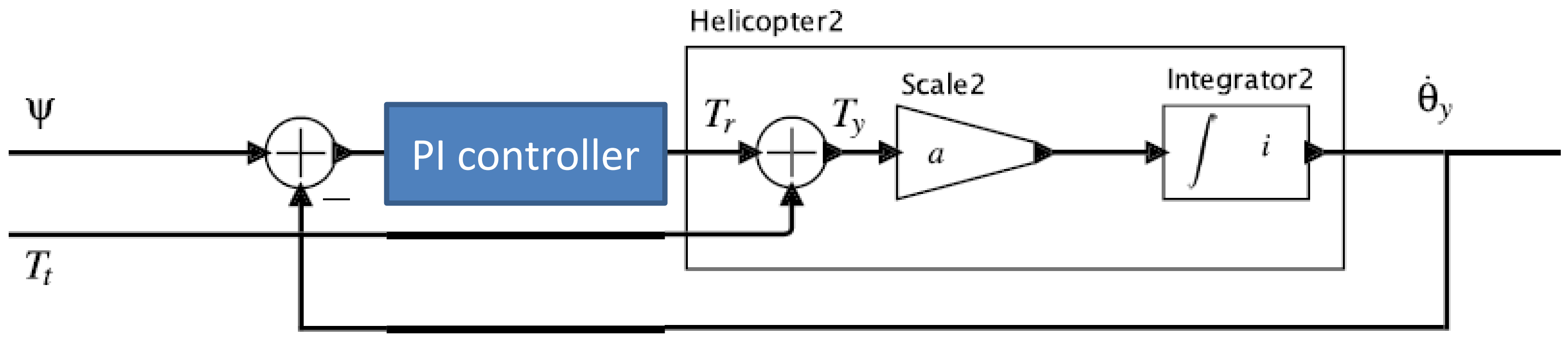
Suppose that desired angular rotation is $\psi(t) = 0$

Then the input to the original control system = $\psi(t) + \frac{\mathrm{T(t)}}{\mathrm{K}} = \frac{b}{K}u(t)$

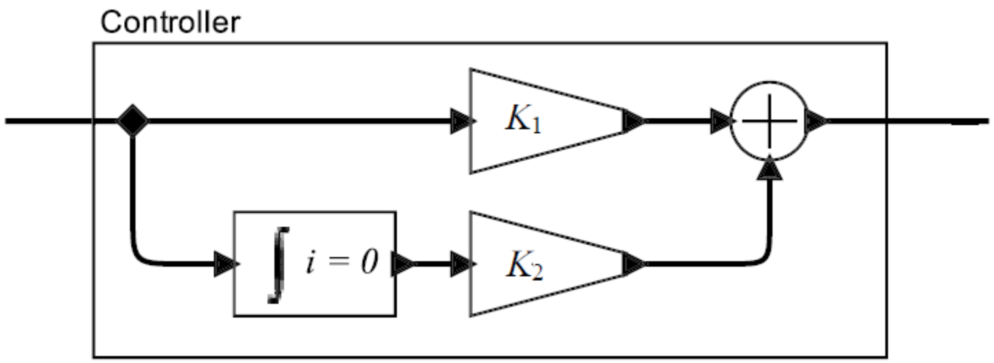
$\Rightarrow$ *Based on Slide* 14: $$\dot{\theta}_y(t) = \frac{b}{K}u(t)(1 - e^{\frac{-Kt}{I_{yy}}})$$

$$\dot{\theta}_y(t) = \frac{b}{K}u(t)(1 - e^{\frac{-Kt}{I_{yy}}})$$

The desired angular rotation is zero, but the control system asymptotically approaches a non-zero angular rotation of $\frac{b}{K}$. This tracking error can be made arbitrarily small by increasing the control system feedback gain K, but with this controller design, it cannot be made to go to zero.



Now, consider how the output will change using a proportional integrator controller, which is depicted as follows:



Suppose $T_t = bu(t)$

$\psi(t) = 0$

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

$$= \frac{1}{I_{yy}}\int_0^t (K_1\left(\psi(\tau) - \dot{\theta}_y(\tau)\right) + K_2\int_0^\tau \left(\psi(\alpha) - \dot{\theta}_y(\alpha)\right)d\alpha + T_t(\tau))d\tau$$

Using the assumption that $\psi(t) = 0$, we have

$$\dot{\theta}_y(\mathrm{t}) = \frac{1}{I_{yy}}\int_0^t (-K_1\dot{\theta}_y(\tau) - K_2\int_0^\tau \dot{\theta}_y(\alpha)d\alpha + T_t(\tau))d\tau$$

$$= \frac{1}{I_{yy}}\int_0^t \left(-K_1\dot{\theta}_y(\tau) - K_2\int_0^\tau \dot{\theta}_y(\alpha)d\alpha\right)d\tau + \frac{1}{I_{yy}}\int_0^t b\,d\tau$$

$$= \frac{1}{I_{yy}}\int_0^t \left(-K_1\dot{\theta}_y(\tau) - K_2\int_0^\tau \dot{\theta}_y(\alpha)d\alpha\right)d\tau + \frac{bt}{I_{yy}}$$

$$\Longrightarrow \ \ddot{\theta}_y(t) = \frac{1}{I_{yy}}\left(-K_1\dot{\theta}_y(t) - K_2\int_0^t \dot{\theta}_y(\alpha)d\alpha\right) + \frac{b}{I_{yy}}$$

$$\Longrightarrow \frac{d^3\theta}{dt^3} = \frac{1}{I_{yy}}\left(-K_1\ddot{\theta}_y(t) - K_2\dot{\theta}_y(\mathrm{t})\right)$$

$$\Longrightarrow s^2\dot{\Theta}(s) - s\dot{\theta}_y(0) - \ddot{\theta}_y(0) = \frac{1}{I_{yy}}\left(-K_1 s\dot{\Theta}(s) + K_1\dot{\theta}_y(0) - K_2\dot{\Theta}(s)\right)$$

$$\Longrightarrow \left(I_{yy}s^2 + K_1 s + K_2\right)\dot{\Theta}(s) = I_{yy}\left(s\dot{\theta}_y(0) + \ddot{\theta}_y(0)\right) + K_1\dot{\theta}_y(0)$$

$$\dot{\Theta}(s) = \frac{I_{yy}\dot{\theta}_y(0)\,s\; + I_{yy}\ddot{\theta}_y(0) + K_1\dot{\theta}_y(0)}{I_{yy}s^2 + K_1 s + K_2} = \frac{\dot{\theta}_y(0)\,s\; + \ddot{\theta}_y(0) + \frac{K_1\dot{\theta}_y(0)}{I_{yy}}}{s^2 + \frac{K_1}{I_{yy}}s + \frac{K_2}{I_{yy}}}$$

Now, based on the roots of the denominator, there are different strategies for fraction decomposition. Since K_1 and K_2 are unknown, we must consider all the strategies.

1) Suppose the denominator has two distinct roots. Then we can write

$$\frac{\dot{\theta}_y(0)\,s\; + \ddot{\theta}_y(0) + \frac{K_1\dot{\theta}_y(0)}{I_{yy}}}{s^2 + \frac{K_1}{I_{yy}}s + \frac{K_2}{I_{yy}}} = \frac{A}{s - r_1} + \frac{B}{s - r_2}$$

$$\dot{\Theta}(s) = \frac{\dot{\theta}_y(0)\, s \ + \ddot{\theta}_y(0) + \frac{K_1\dot{\theta}_y(0)}{I_{yy}}}{s^2 + \frac{K_1}{I_{yy}} s + \frac{K_2}{I_{yy}}} = \frac{A}{s - r_1} + \frac{B}{s - r_2}$$

Where $r_1 = \frac{-K_1 + \sqrt{K_1^2 - 4I_{yy}K_2}}{2I_{yy}}$ and $r_2 = \frac{-K_1 - \sqrt{K_1^2 - 4I_{yy}K_2}}{2I_{yy}}$

$\boxed{K_1^2 - 4I_{yy}K_2 > 0}$

$$\Longrightarrow \begin{cases} A + B = \dot{\theta}_y(0) \\ -Ar_2 - Br_1 = \ddot{\theta}_y(0) + \frac{K_1\dot{\theta}_y(0)}{I_{yy}} \end{cases}$$

$$\Longrightarrow \quad -Ar_2 - \left(\dot{\theta}_y(0) - A\right) r_1 = \ddot{\theta}_y(0) + \frac{K_1\dot{\theta}_y(0)}{I_{yy}}$$

$$\Longrightarrow -A \times \frac{-K_1 - \sqrt{K_1^2 - 4I_{yy}K_2}}{2I_{yy}} - \left(\dot{\theta}_y(0) - A\right)\frac{-K_1 + \sqrt{K_1^2 - 4I_{yy}K_2}}{2I_{yy}} = \ddot{\theta}_y(0) + \frac{K_1\dot{\theta}_y(0)}{I_{yy}}$$

$$\Rightarrow \cancel{AK_1} + A\sqrt{K_1^2 - 4I_{yy}K_2} + \cancel{\dot{\theta}_y(0)}K_1 - \dot{\theta}_y(0)\sqrt{K_1^2 - 4I_{yy}K_2} - \cancel{AK_1}$$

$$+ A\sqrt{K_1^2 - 4I_{yy}K_2} = 2I_{yy}\ddot{\theta}_y(0) + 2K_1\dot{\theta}_y\cancel{(0})$$

$$\Rightarrow 2A\sqrt{K_1^2 - 4I_{yy}K_2} = \dot{\theta}_y(0)\left(K_1 + \sqrt{K_1^2 - 4I_{yy}K_2}\right) + 2I_{yy}\ddot{\theta}_y(0)$$

$$\Rightarrow \mathrm{A} = \frac{\dot{\theta}_y(0)}{2} + \frac{K_1\dot{\theta}_y(0) + 2I_{yy}\ddot{\theta}_y(0)}{2\sqrt{K_1^2 - 4I_{yy}K_2}}$$

$$\Rightarrow \mathrm{B} = \frac{\dot{\theta}_y(0)}{2} - \frac{K_1\dot{\theta}_y(0) + 2I_{yy}\ddot{\theta}_y(0)}{2\sqrt{K_1^2 - 4I_{yy}K_2}}$$

Now, we can analyze the system behavior over time using Laplace inverse transform:

$$\dot{\theta}_y(\mathrm{t}) = Ae^{r_1 t} + Be^{r_2 t}$$

$$= \frac{1}{2}\left(\dot{\theta}_y(0) + \frac{K_1\dot{\theta}_y(0) + 2I_{yy}\ddot{\theta}_y(0)}{\sqrt{K_1^2 - 4I_{yy}K_2}}\right)e^{\frac{-K_1+\sqrt{K_1^2-4I_{yy}K_2}}{2I_{yy}}t}$$

$$+ \frac{1}{2}\left(\dot{\theta}_y(0) - \frac{K_1\dot{\theta}_y(0) + 2I_{yy}\ddot{\theta}_y(0)}{\sqrt{K_1^2 - 4I_{yy}K_2}}\right)e^{\frac{-K_1-\sqrt{K_1^2-4I_{yy}K_2}}{2I_{yy}}t}$$

2) Suppose the denominator has two imaginary roots. Then we can write:

$$\dot{\Theta}(s) = \frac{\dot{\theta}_y(0)\, s \; + \ddot{\theta}_y(0) + \frac{K_1\dot{\theta}_y(0)}{I_{yy}}}{\mathrm{s}^2 + \frac{K_1}{I_{yy}} s + \frac{K_2}{I_{yy}}} = \frac{\dot{\theta}_y(0)\left(s + \frac{K_1}{2I_{yy}}\right) + \left(\ddot{\theta}_y(0) + \frac{K_1\dot{\theta}_y(0)}{2I_{yy}}\right)}{\left(s + \frac{K_1}{2I_{yy}}\right)^2 + \frac{4I_{yy}K_2 - {K_1}^2}{4{I_{yy}}^2}}$$

$$\dot{\theta}_y(\mathrm{t}) = e^{\frac{-K_1 t}{2I_{yy}}}\left(\dot{\theta}_y(0)\cos\left(\frac{\sqrt{4I_{yy}K_2 - {K_1}^2}}{2I_{yy}} t\right) + \frac{\ddot{\theta}_y(0) + \frac{K_1\dot{\theta}_y(0)}{2I_{yy}}}{\frac{\sqrt{4I_{yy}K_2 - {K_1}^2}}{2I_{yy}}}\sin\left(\frac{\sqrt{4I_{yy}K_2 - {K_1}^2}}{2I_{yy}} t\right)\right)$$

$$\dot{\theta}_y(\mathrm{t}) = e^{\frac{-K_1 t}{2I_{yy}}}\left(\dot{\theta}_y(0)\cos\left(\frac{\sqrt{4I_{yy}K_2 - {K_1}^2}}{2I_{yy}} t\right) + \frac{K_1\dot{\theta}_y(0) + 2I_{yy}\ddot{\theta}_y(0)}{\sqrt{K_1^2 - 4I_{yy}K_2}}\sin\left(\frac{\sqrt{4I_{yy}K_2 - {K_1}^2}}{2I_{yy}} t\right)\right)$$

note that our assumption about the imaginary roots is correct only if $K_1^2 - 4I_{yy}K_2 < 0$.

3) Suppose the denominator has a double root. Notice that in such a situation the equation $K_1^2 = 4I_{yy}K_2$ holds. so we can write

$$\dot{\Theta}(s) = \frac{\dot{\theta}_y(0)\, s \ + \ddot{\theta}_y(0) + \dfrac{K_1\dot{\theta}_y(0)}{I_{yy}}}{s^2 + \dfrac{K_1}{I_{yy}}s + \dfrac{K_2}{I_{yy}}} = \frac{A}{s + \dfrac{K_1}{2I_{yy}}} + \frac{B}{\left(s + \dfrac{K_1}{2I_{yy}}\right)^2}$$

$$\Longrightarrow A\left(s + \frac{K_1}{2I_{yy}}\right) + B = \dot{\theta}_y(0)\, s \ + \ddot{\theta}_y(0) + \frac{K_1\dot{\theta}_y(0)}{I_{yy}}$$

$$\Longrightarrow \begin{cases} A = \dot{\theta}_y(0) \\ B = \dfrac{K_1\dot{\theta}_y(0) + 2I_{yy}\ddot{\theta}_y(0)}{2I_{yy}} \end{cases}$$

$$\Longrightarrow \dot{\theta}_y(\mathrm{t}) = e^{\frac{-K_1 t}{2I_{yy}}}\left(\dot{\theta}_y(0) + \frac{K_1\dot{\theta}_y(0) + 2I_{yy}\ddot{\theta}_y(0)}{2I_{yy}}\mathrm{t}\right)$$

As an outline, we rewrite what we have calculated:

$$K_1^2 - 4I_{yy}K_2 > 0$$

$$\Rightarrow \dot{\theta}_y(\mathrm{t}) = \frac{1}{2}\left(\dot{\theta}_y(0) + \frac{K_1\dot{\theta}_y(0) + 2I_{yy}\ddot{\theta}_y(0)}{\sqrt{K_1^2 - 4I_{yy}K_2}}\right)e^{\frac{-K_1+\sqrt{K_1^2-4I_{yy}K_2}}{2I_{yy}}t}$$

overdamped

$$+\frac{1}{2}\left(\dot{\theta}_y(0) - \frac{K_1\dot{\theta}_y(0) + 2I_{yy}\ddot{\theta}_y(0)}{\sqrt{K_1^2 - 4I_{yy}K_2}}\right)e^{\frac{-K_1-\sqrt{K_1^2-4I_{yy}K_2}}{2I_{yy}}t}$$

$$***$$

underdamped

$$K_1^2 - 4I_{yy}K_2 < 0 \Rightarrow$$

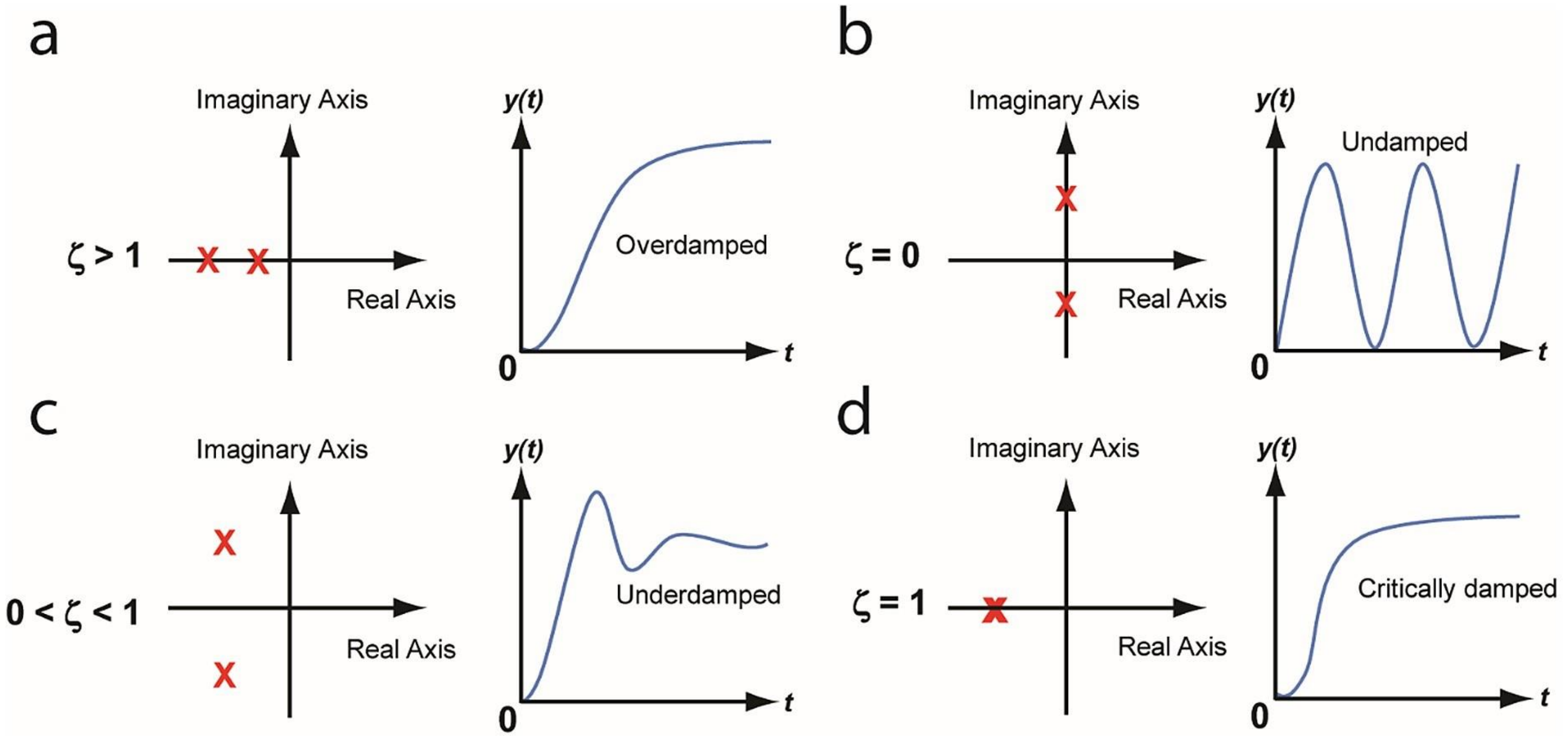
$$\dot{\theta}_y(\mathrm{t}) = e^{\frac{-K_1 t}{2I_{yy}}}\left(\dot{\theta}_y(0)\cos\left(\frac{\sqrt{4I_{yy}K_2 - {K_1}^2}}{2I_{yy}}t\right) + \frac{K_1\dot{\theta}_y(0) + 2I_{yy}\ddot{\theta}_y(0)}{\sqrt{K_1^2 - 4I_{yy}K_2}}\sin\left(\frac{\sqrt{4I_{yy}K_2 - {K_1}^2}}{2I_{yy}}t\right)\right)$$

$$***$$

$$K_1^2 = 4I_{yy}K_2$$

$$\Rightarrow \dot{\theta}_y(\mathrm{t}) = e^{\frac{-K_1 t}{2I_{yy}}}\left(\dot{\theta}_y(0) + \frac{K_1\dot{\theta}_y(0) + 2I_{yy}\ddot{\theta}_y(0)}{2I_{yy}}\mathrm{t}\right)$$

critically damped

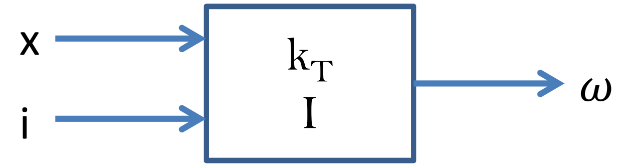

a
Imaginary Axis
$\zeta > 1$
Real Axis
y(t)
Overdamped
0
t
b
Imaginary Axis
$\zeta = 0$
Real Axis
y(t)
Undamped
0
t
c
Imaginary Axis
$0 < \zeta < 1$
Real Axis
y(t)
Underdamped
0
t
d
Imaginary Axis
$\zeta = 1$
Real Axis
y(t)
Critically damped
0
t

A DC motor produces a torque that is proportional to the current through the windings of the motor. Neglecting friction, the net torque on the motor, therefore, is this torque minus the torque applied by whatever load is connected to the motor. Newton's second law (the rotational version) gives

$$k_T\, i(t) - x(t) = I\frac{d\omega}{dt}$$

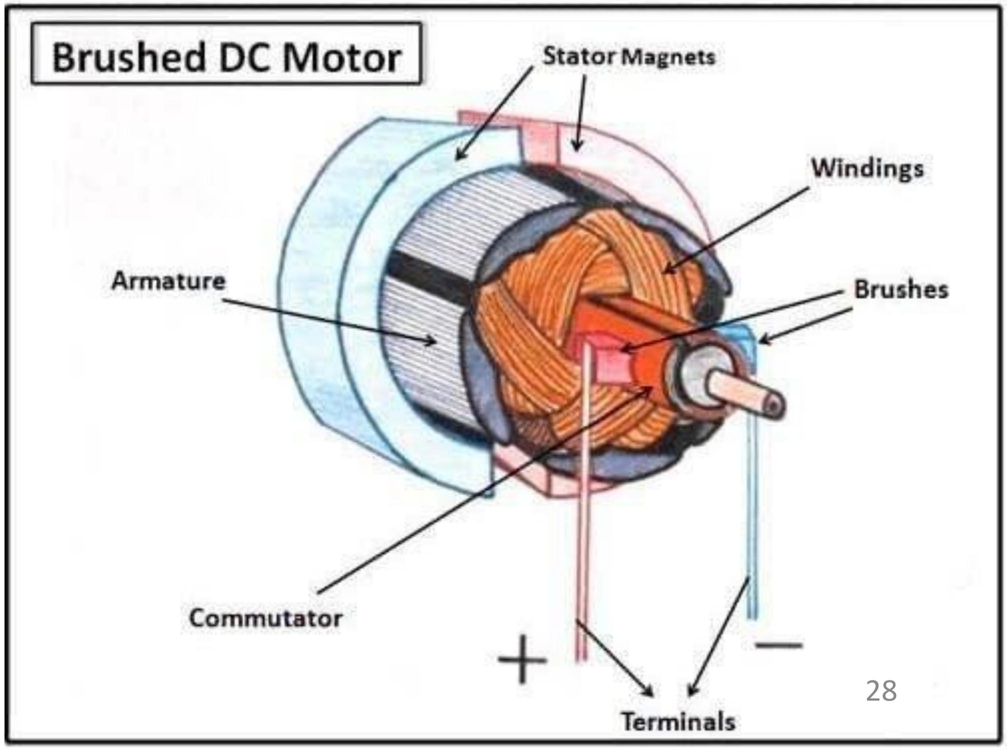
motor torque constant

torque applied by the load



Assuming the motor is initially at rest, we can rewrite the equation:

$$\omega(t) = \frac{k_T}{I}\int_0^t i(\tau)d\tau - \frac{1}{I}\int_0^t x(\tau)d\tau$$



In reality, the input to a DC motor is not a current, but is rather a voltage. If we assume that the inductance of the motor windings is negligible, then the relationship between voltage and current is given by

$$v(t) = Ri(t) + k_b\omega(\mathrm{t})$$

where R is the resistance of the motor windings and k_b is a constant called the motor back electromagnetic force constant. The second term appears because a rotating motor also functions as an electrical generator, where the voltage generated is proportional to the angular velocity.

So we can modify the previous equation:

$$\frac{k_T}{R}\left(v(t) - k_b\omega(\mathrm{t})\right) - x(t) = I\frac{d\omega}{dt}$$

$$\Longrightarrow \dot{\omega} + \frac{k_b k_T}{RI}\omega + \left(\frac{Rx(t) - k_T v(t)}{RI}\right) = 0$$

Chapter 3

- Discrete dynamics

A discrete system operates in a sequence of discrete steps and is said to have discrete dynamics. Some systems are inherently discrete.

Example) A system counting the number of cars entering and leaving a parking garage, in order to display the number of cars occupying the garage.

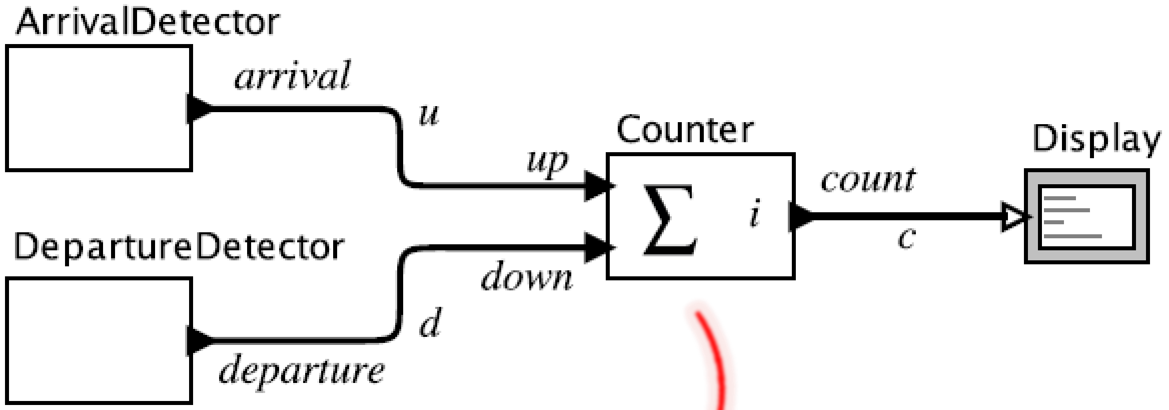
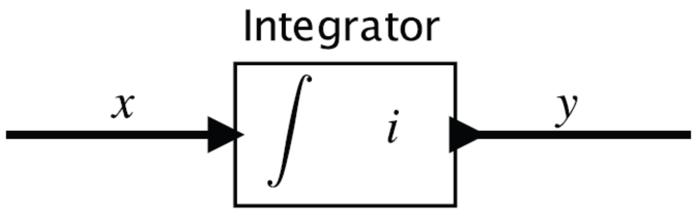
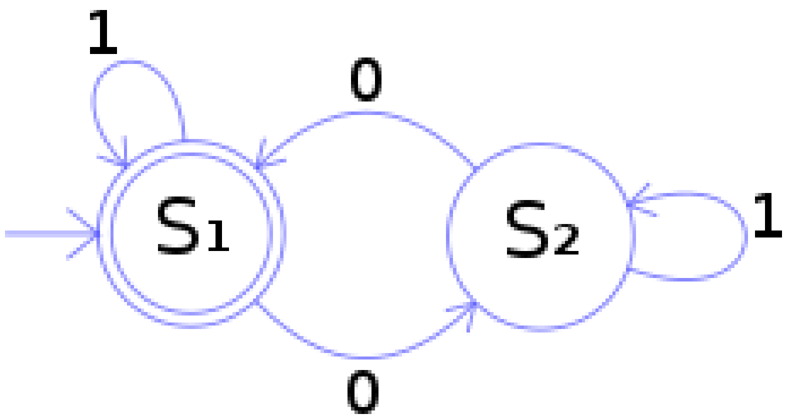


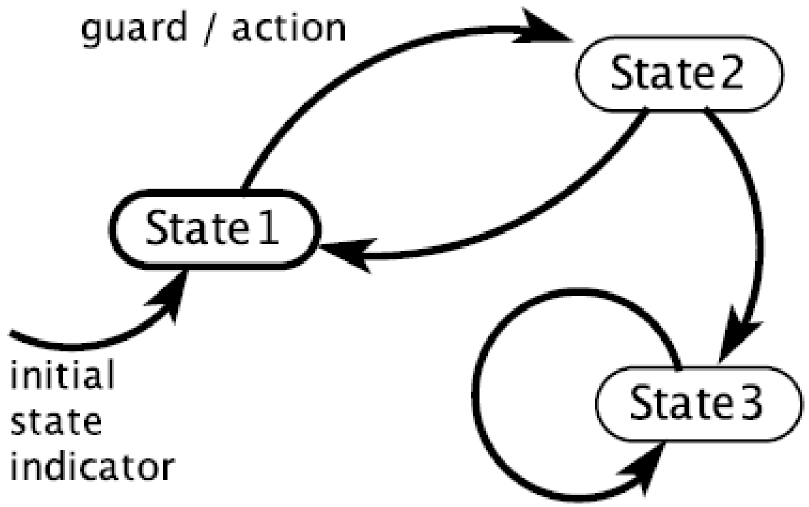
Figure 3.1: Model of a system that keeps track of the number of cars in a parking garage.



The corresponding system that we had in previous chapter

How to model the discrete systems?





Finite state machine (automata)

However, it is just for showing the states and there may not be a final state.

inputs: *up*, *down* : pure
output: *count* : $\{0, \cdots, M\}$

A pure signal is either present or absent

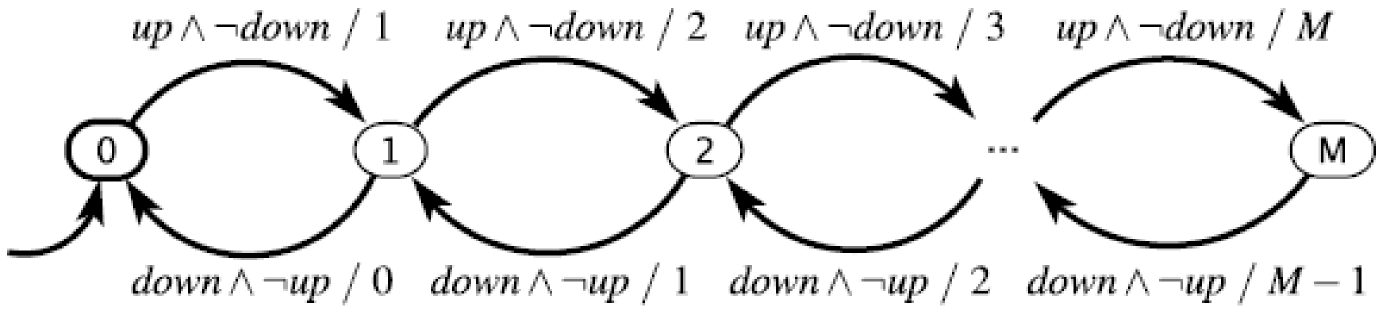
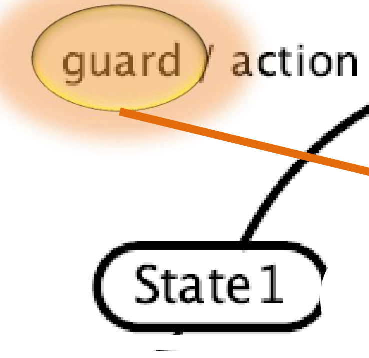

$up \wedge \neg down / 1$
$up \wedge \neg down / 2$
$up \wedge \neg down / 3$
$up \wedge \neg down / M$
0
1
2
...
M
$down \wedge \neg up / 0$
$down \wedge \neg up / 1$
$down \wedge \neg up / 2$
$down \wedge \neg up / M-1$

Figure 3.4: FSM model for the garage counter of Figure 3.1.



Boolean predicate

For example:

True: Transition is always enabled.

p1: Transition is enabled if p1 is present.

¬ p1: Transition is enabled if p1 is absent.

p1 ∧ p2: Transition is enabled if both p1 and p2 are present.

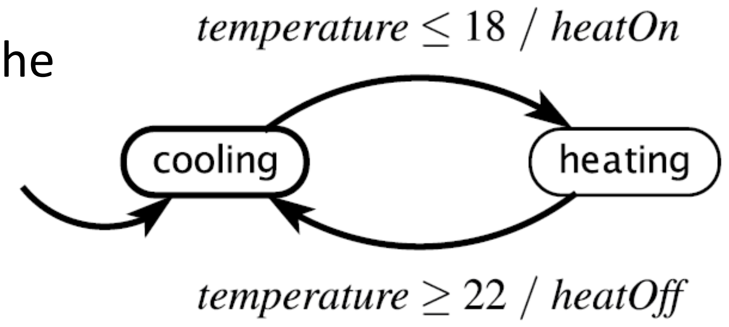
p1 ∨ p2 : Transition is enabled if either p1 or p2 is present.

p1 ∧ ¬ p2: Transition is enabled if p1 is present and p2 is absent.

p3 = 1

p3 > 5

Suppose the setpoint is 20 degrees Celsius. If the heater is on, then the thermostat allows the temperature to rise past the setpoint to 22 degrees. If the heater is off, then it allows the temperature to drop past the setpoint to 18 degrees. This strategy is called hysteresis.


$temperature \leq 18 \ / \ heatOn$
cooling
heating
$temperature \geq 22 \ / \ heatOff$

A model of a thermostat with hysteresis.

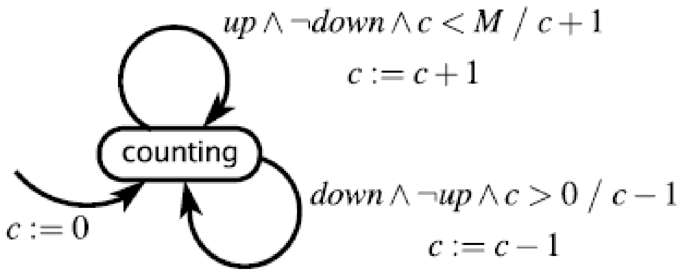
Hysteresis avoids chattering, where the heater would turn on and off rapidly when the temperature is close to the setpoint temperature.

$$\begin{tikzcd}[column sep=large] & \textbf{Event triggered} & \text{like garage counter, HVAC (heating, ventilation, and air conditioning)} \\ \textbf{FSM} \arrow[ur] \arrow[dr] & & \\ & \textbf{Time triggered} & \text{timed automata like traffic light} \end{tikzcd}$$

Extended State machine

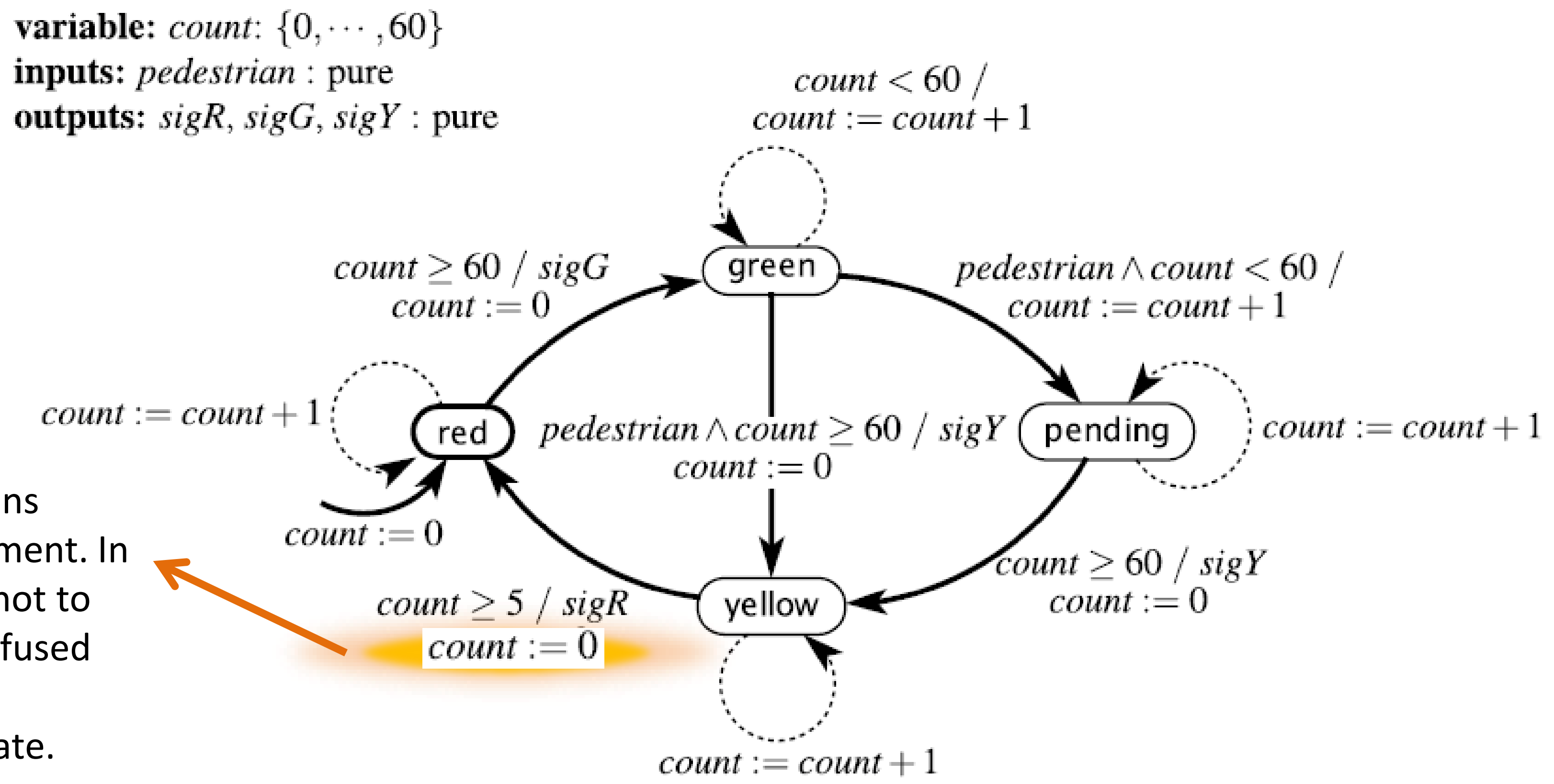
The notation for FSMs becomes awkward when the number of states gets large. The garage counter example illustrates this point clearly.

variable: c: $\{0, \cdots, M\}$
inputs: up, $down$: pure
output: $count$: $\{0, \cdots, M\}$



$up \wedge \neg down \wedge c < M \ / \ c+1$
$c := c+1$
counting
$down \wedge \neg up \wedge c > 0 \ / \ c-1$
$c := c-1$
$c := 0$

Figure 3.8: Extended state machine for the garage counter of Figure 3.4.



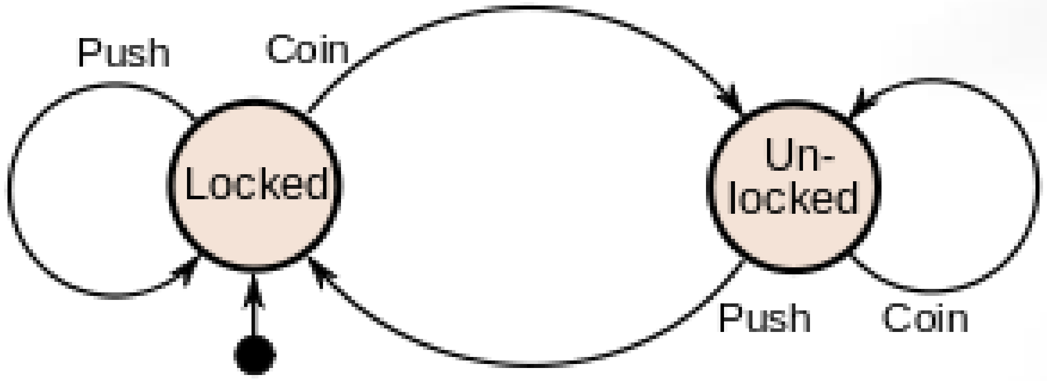
variable: $count$: $\{0, \cdots, 60\}$
inputs: $pedestrian$: pure
outputs: $sigR$, $sigG$, $sigY$: pure
$count < 60$ / $count := count + 1$
green
red
pending
yellow
$count \geq 60$ / $sigG$ $count := 0$
$pedestrian \wedge count < 60$ / $count := count + 1$
$count := count + 1$
$pedestrian \wedge count \geq 60$ / $sigY$ $count := 0$
$count := count + 1$
$count := 0$
$count \geq 60$ / $sigY$ $count := 0$
$count \geq 5$ / $sigR$ $count := 0$
$count := count + 1$

:= means assignment. In order not to be confused with a predicate.

Figure 3.10: Extended state machine model of a traffic light controller that keeps track of the passage of time, assuming it reacts at regular intervals.

When pedestrian is present, the machine transitions to yellow if it has been in state green for at least 60 seconds. Otherwise, it transitions to pending, where it stays for the remainder of the 60 second interval. This ensures that once the light goes green, it stays green for at least 60 seconds.

Another example from out of the book



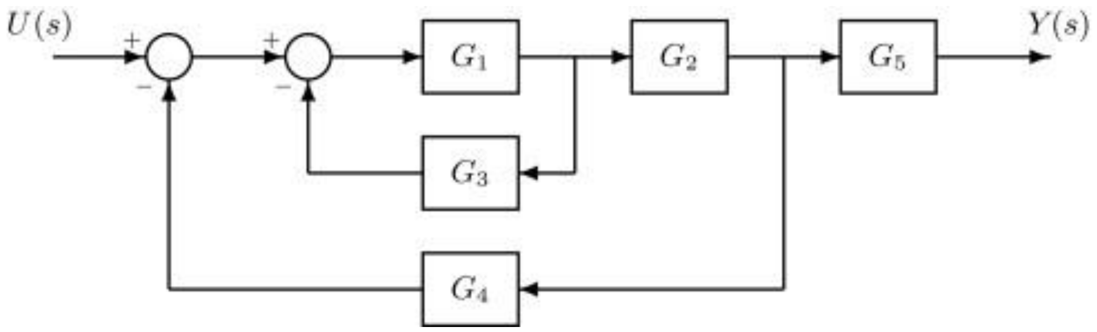
turnstile

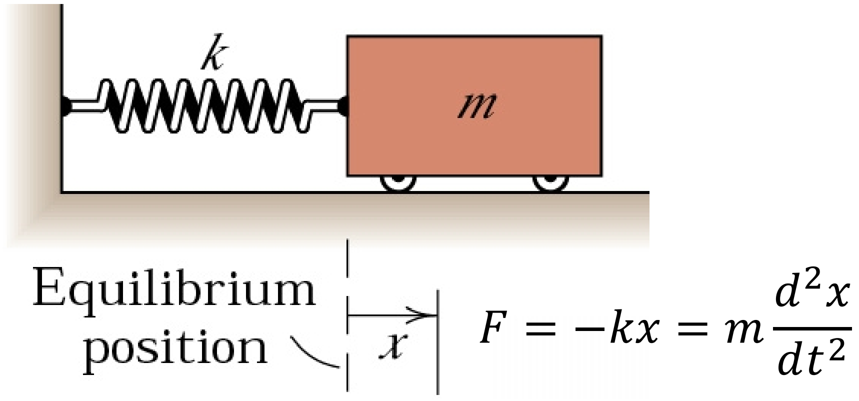
Chapter 4

- Hybrid systems

We have learned so far

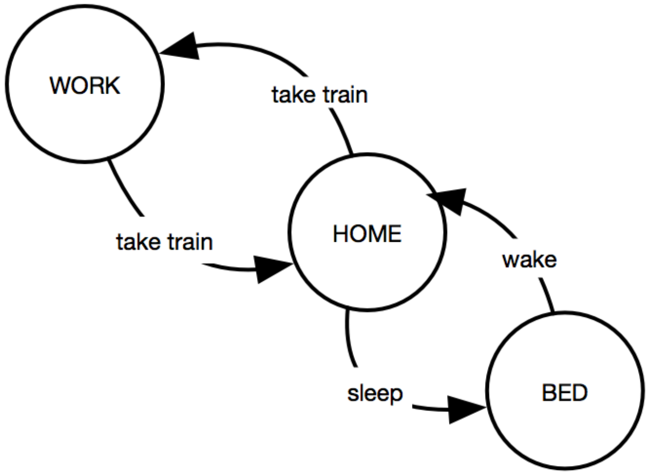
Continuous systems





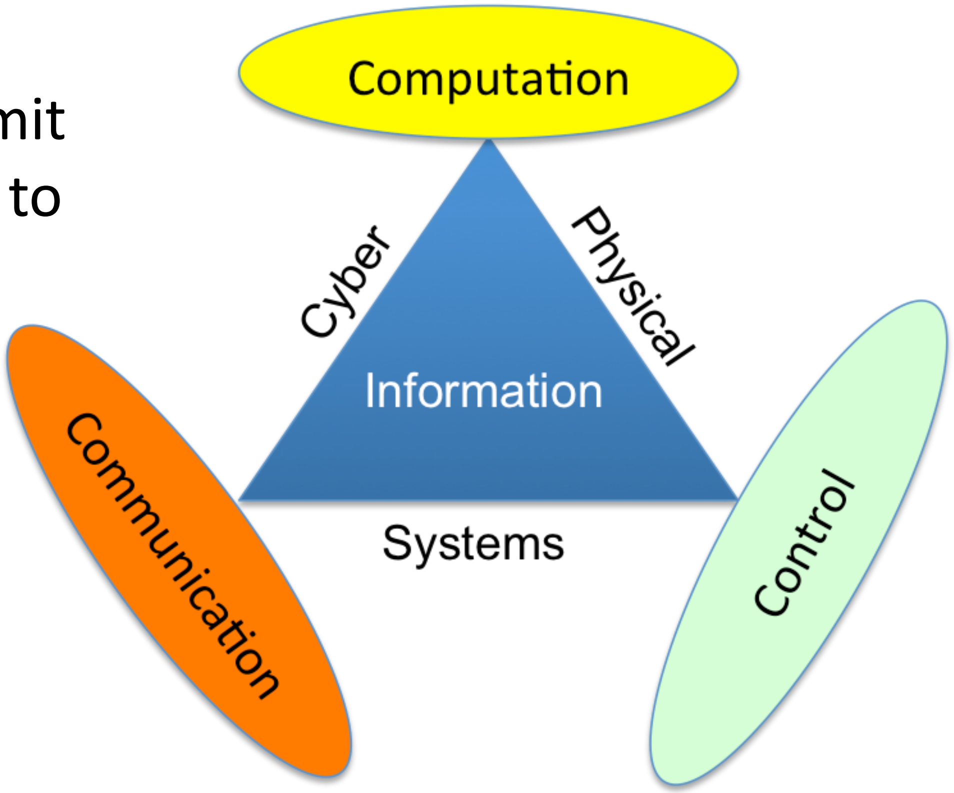
$$F = -kx = m\frac{d^2x}{dt^2}$$

Discrete systems



Now we want to combine these systems. Why?

In this section, we show that state machines can be generalized to admit continuous inputs and outputs and to combine discrete and continuous dynamics.



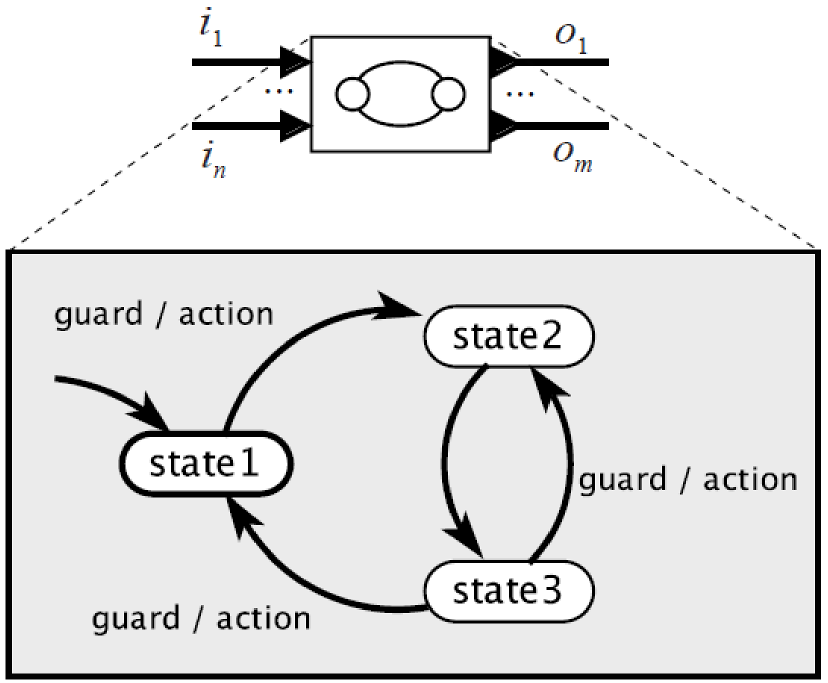


Figure 4.1: An FSM represented as an actor.

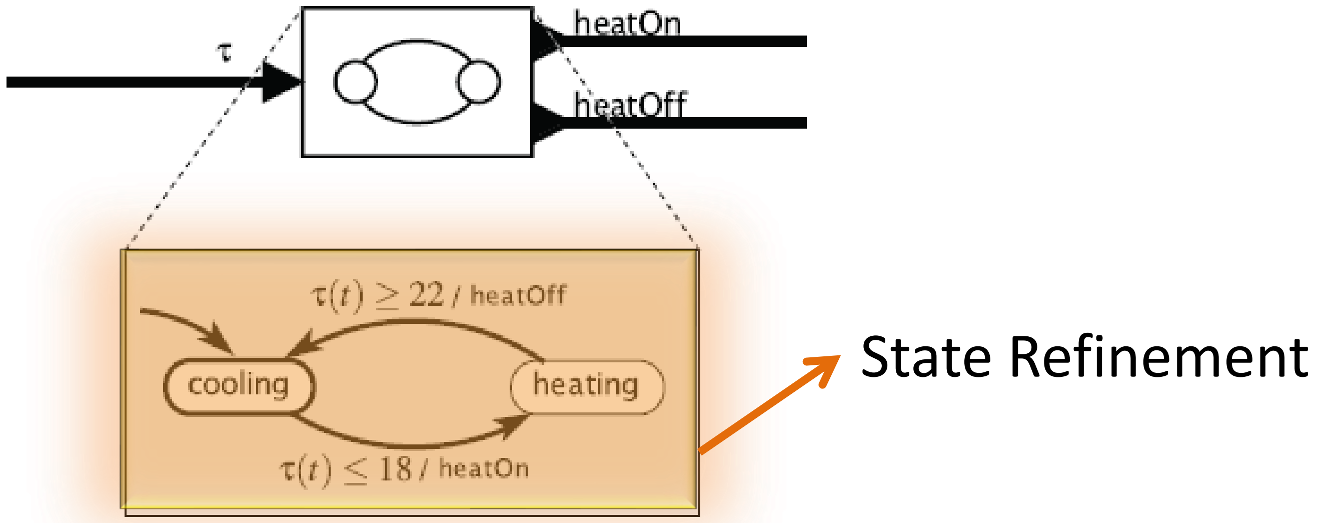
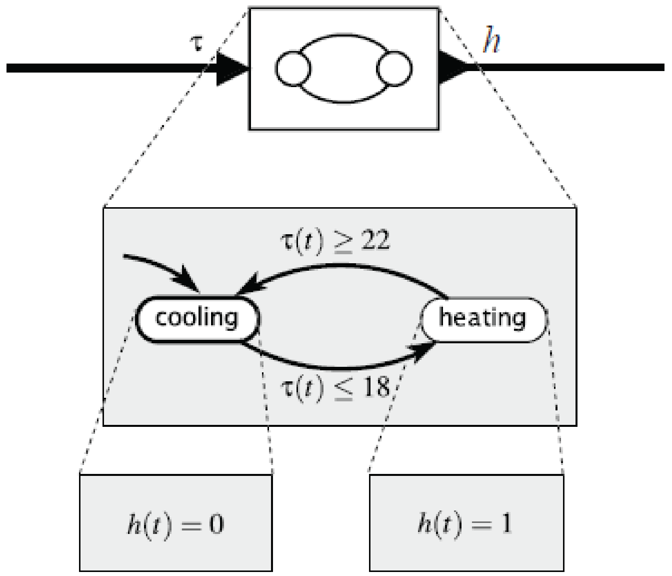


Figure 4.2: A thermostat modeled as an FSM with a continuous-time input signal.



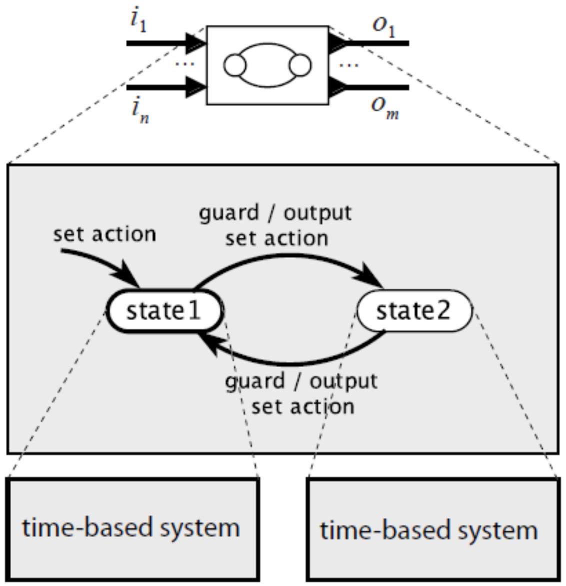


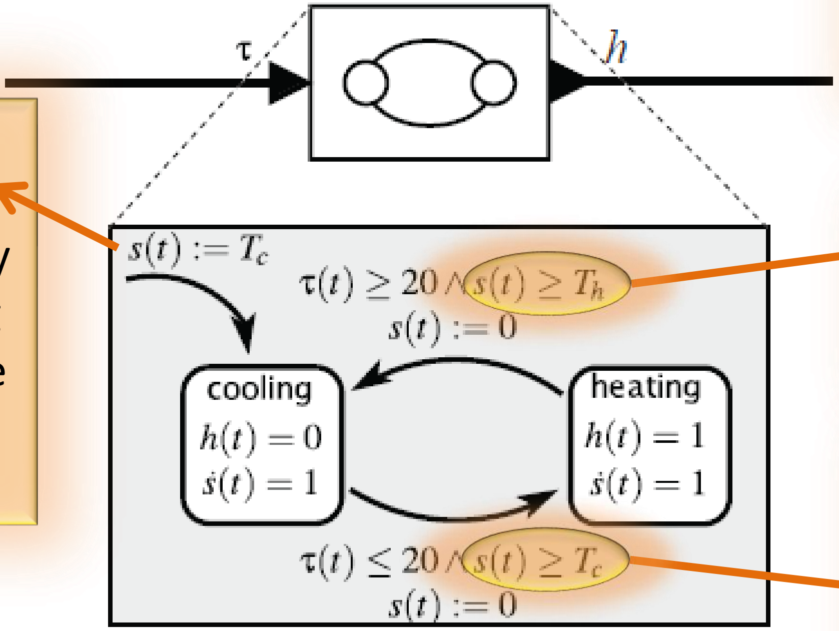
Figure 4.3: A thermostat with continuous-time output.

Figure 4.4: Notation for hybrid systems.

Recall the thermostat of Example 4.1, which uses hysteresis to prevent chattering. An alternative implementation that would also prevent chattering would use a single temperature threshold, but instead would require that the heater remain on or off for at least a minimum amount of time, regardless of the temperature. This design would not have the hysteresis property, but may be useful nonetheless.

s is a continuous-time signal with dynamics given by $\dot{s}$ (t) = 1 representing the clock

This action ensures that when the thermostat starts, it can immediately transition to the heating mode if the temperature τ(t) is less than or equal to 20 degrees.



ensures that the heater will always be on for at least time T_h.

specifies that once the heater goes off, it will remain off for at least time T_c.

Figure 4.5: A timed automaton modeling a thermostat with a single temperature threshold, 20, and minimum times T_c and T_h in each mode.

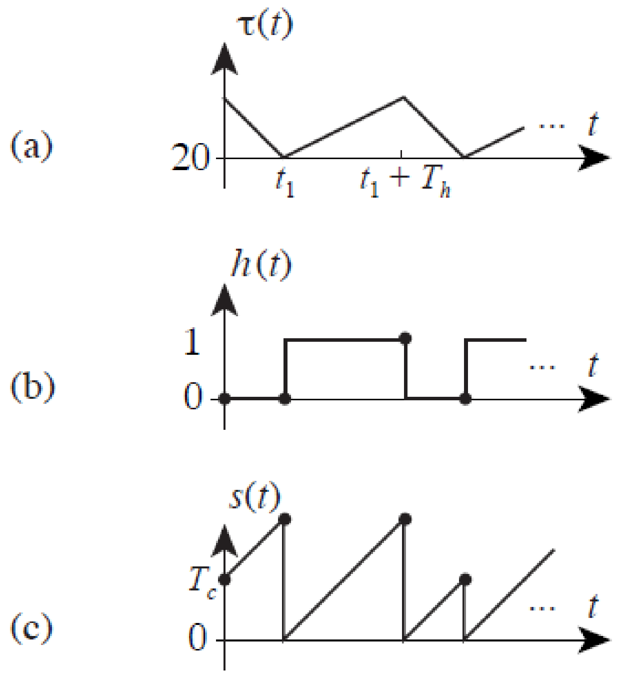
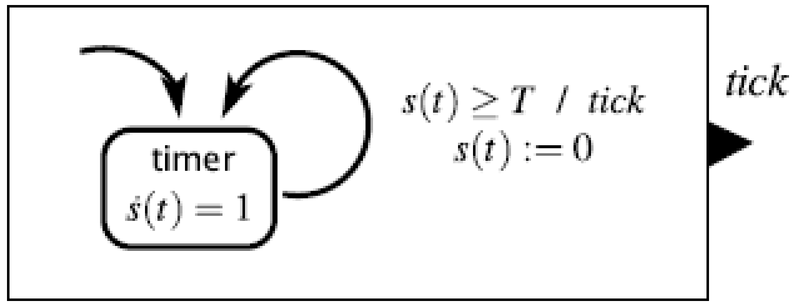
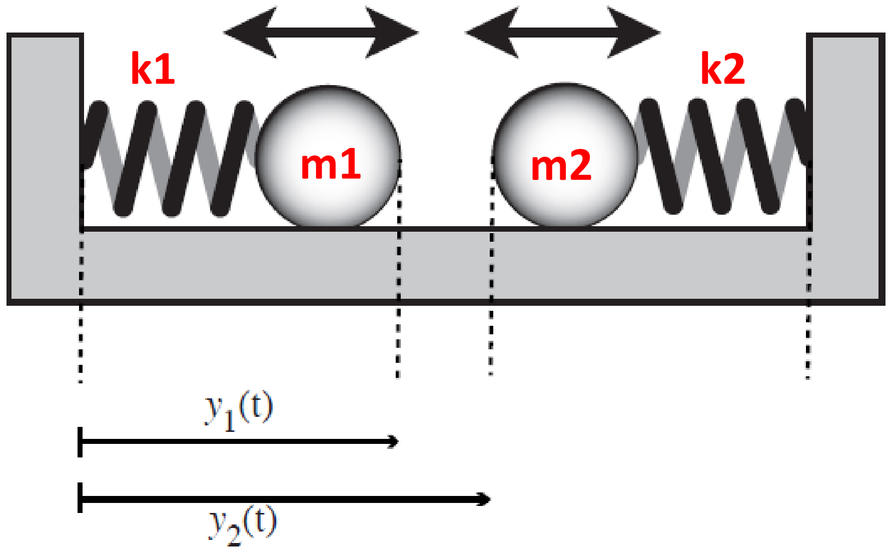


Figure 4.6: (a) A temperature input to the hybrid system of Figure 4.5, (b) the output h, and (c) the refinement state s.



A timed automaton that generates a pure output event every T time units.

Consider the physical system depicted below. Two **sticky** round masses are attached to springs. The springs are compressed or extended and then released. The masses oscillate on a **frictionless** table. If they collide, **they stick together and oscillate together.** After some time, the **stickiness decays**, and masses pull apart again.



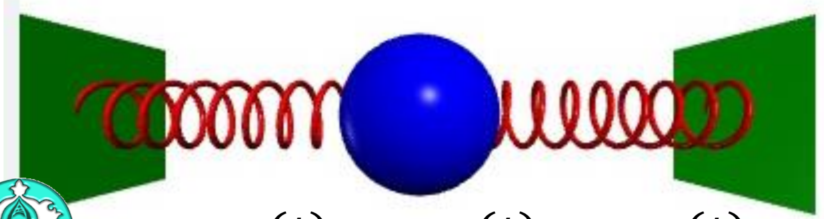
p_1 , p_2 : neutral position of two masses respectively (when the springs are neither extended nor compressed)

1) If the masses are separate:

$$F = k(p - y) = m\frac{d^2y}{dt^2}$$

$$\Rightarrow \ddot{y}_i(t) = \frac{k_i(p_i - y_i(t))}{m_i}$$

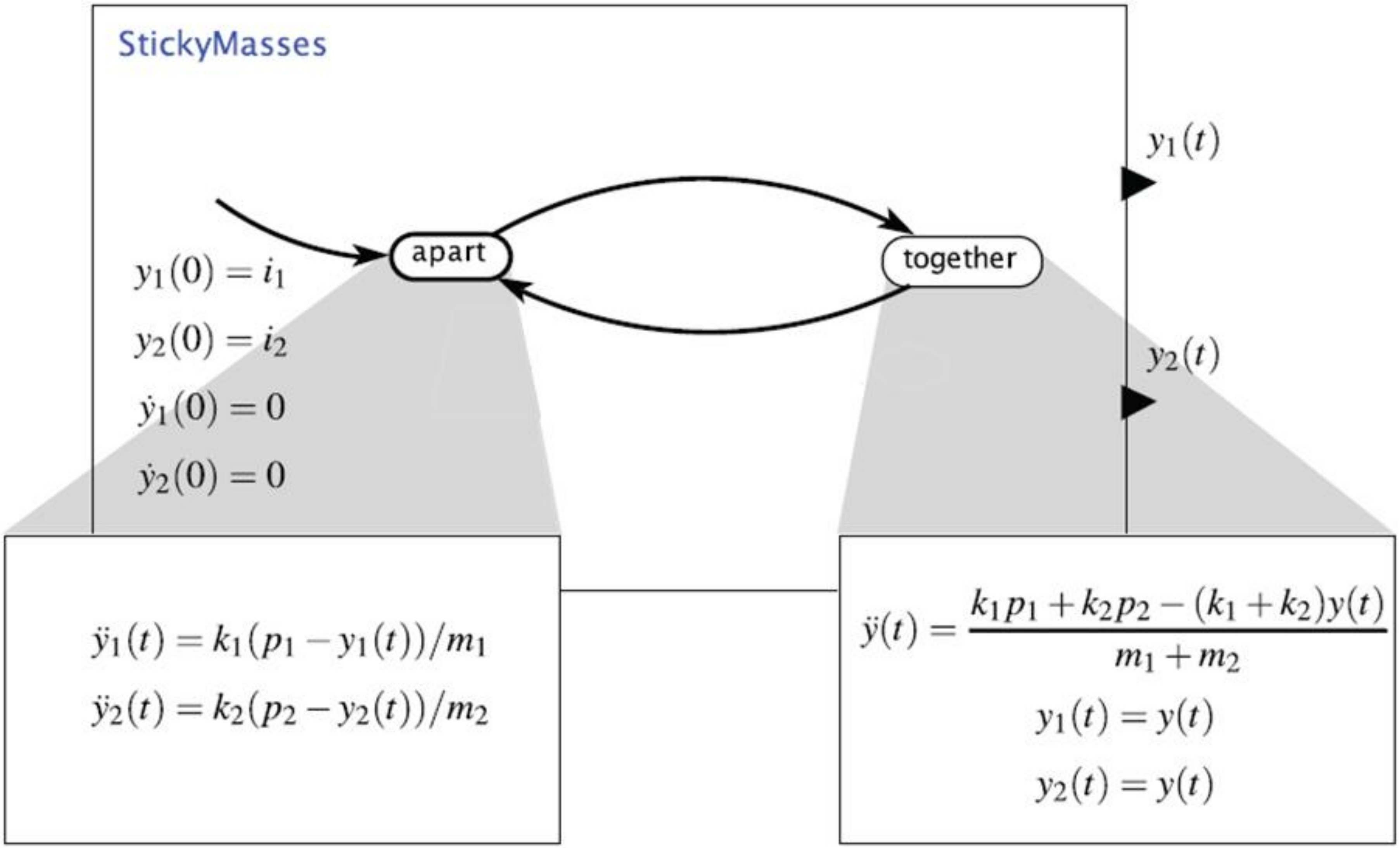
2) With the masses stuck together, they behave as a single object with mass m1 +m2.



$$F = k_1(p_1 - y) + k_2(p_2 - y) = m\frac{d^2y}{dt^2}$$

$$y(t) = y_1(t) = y_2(t)$$

$$\Rightarrow \ddot{y}(t) = \frac{k_1p_1 + k_2p_2 - (k_1 + k_2)y(t)}{m_1 + m_2}$$


StickyMasses
apart
together
$y_1(t)$
$y_2(t)$
$y_1(0) = i_1$
$y_2(0) = i_2$
$\dot{y}_1(0) = 0$
$\dot{y}_2(0) = 0$
$\ddot{y}_1(t) = k_1(p_1 - y_1(t))/m_1$
$\ddot{y}_2(t) = k_2(p_2 - y_2(t))/m_2$
$\ddot{y}(t) = \frac{k_1 p_1 + k_2 p_2 - (k_1 + k_2) y(t)}{m_1 + m_2}$
$y_1(t) = y(t)$
$y_2(t) = y(t)$

Now, we must determine when the system changes it's state. Then we should find the velocity in each action, since in order to describe the motion of system in each state, we need the position and velocity at the time that the state changes.

The transition from apart to together has the guard $y_1(t) = y_2(t)$

When the masses stick together, based on the conservation of momentum we have:

$$m_1\dot{y}_1(t) + m_2\dot{y}_2(t) = (m_1 + m_2)\dot{y}(t)$$

$$\Longrightarrow \dot{y}(t) = \frac{m_1\dot{y}_1(t) + m_2\dot{y}_2(t)}{m_1 + m_2}$$

The transition from together to apart has the more complicated guard:

$$(k_1 - k_2)y(t) + k_2p_2 - k_1p_1 > s$$

stickiness of two masses

That is, the right-pulling force on the right mass exceeds the right-pulling force on the left mass by more than the stickiness.

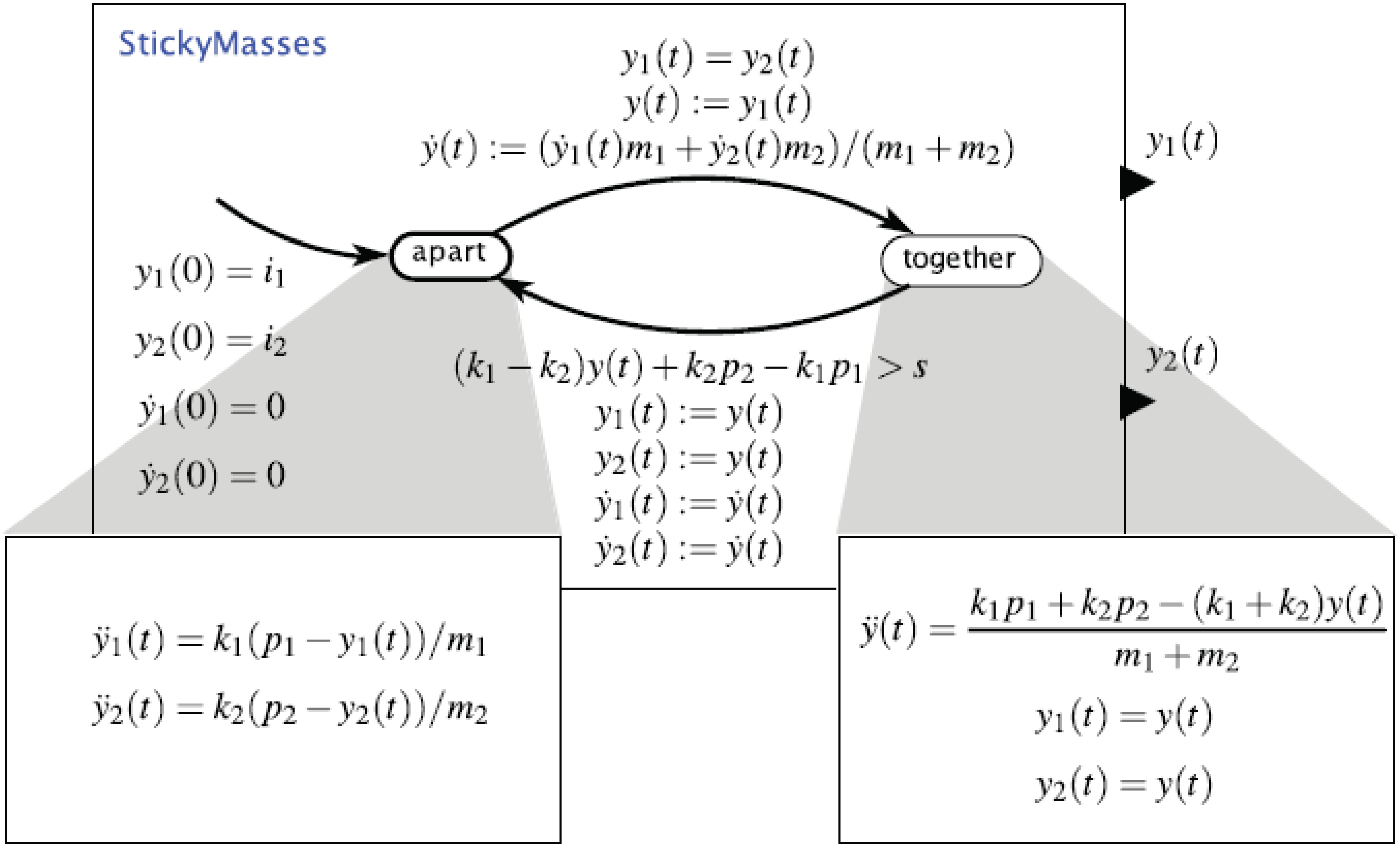
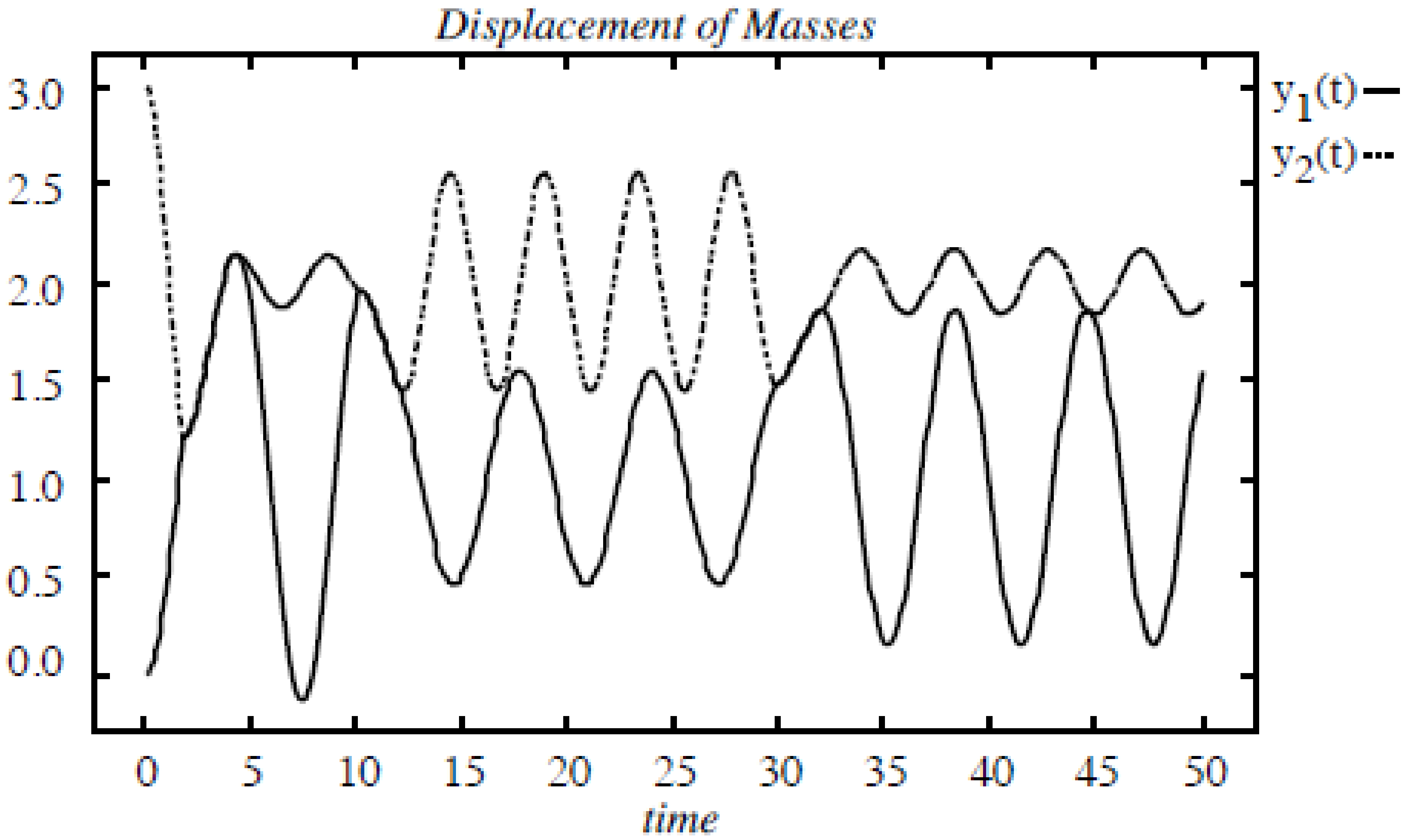
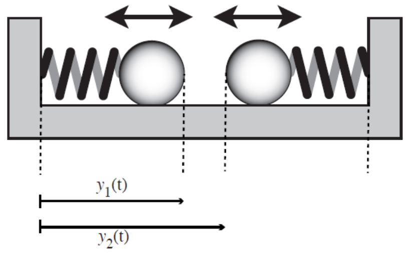

StickyMasses
$y_1(t) = y_2(t)$
$y(t) := y_1(t)$
$\dot{y}(t) := (\dot{y}_1(t)m_1 + \dot{y}_2(t)m_2)/(m_1+m_2)$
$y_1(t)$
$y_2(t)$
apart
together
$y_1(0) = i_1$
$y_2(0) = i_2$
$\dot{y}_1(0) = 0$
$\dot{y}_2(0) = 0$
$(k_1 - k_2)y(t) + k_2 p_2 - k_1 p_1 > s$
$y_1(t) := y(t)$
$y_2(t) := y(t)$
$\dot{y}_1(t) := \dot{y}(t)$
$\dot{y}_2(t) := \dot{y}(t)$
$\ddot{y}_1(t) = k_1(p_1 - y_1(t))/m_1$
$\ddot{y}_2(t) = k_2(p_2 - y_2(t))/m_2$
$\ddot{y}(t) = \frac{k_1 p_1 + k_2 p_2 - (k_1+k_2)y(t)}{m_1+m_2}$
$y_1(t) = y(t)$
$y_2(t) = y(t)$

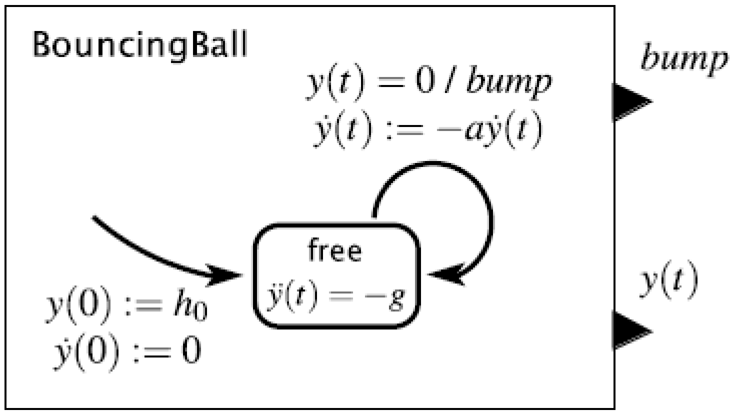
Figure 4.10: Hybrid system model for the sticky masses system considered in Example 4.6.

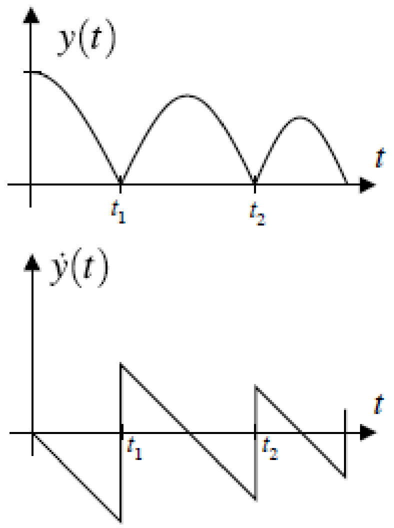




After determining all of the system features, we can predict the system behavior by programming or using a simulator.

Example 4.7: Consider a bouncing ball. At time $t = 0$, the ball is dropped from a height $y(0) = h_0$, where h_0 is the initial height in meters. It falls freely. At some later time t_1 it hits the ground with a velocity $\dot{y}(t_1) < 0$ m/s (meters per second). A *bump* event is produced when the ball hits the ground. The collision is **inelastic** (meaning that kinetic energy is lost), and the ball bounces back up with velocity $-a\dot{y}(t_1)$, where a is constant with $0 < a < 1$. The ball will then rise to a certain height and fall back to the ground repeatedly.





The system outputs a bump each time the ball hits the ground, and also outputs the position of the ball.

This system is depicted as an event triggered system (y(t)=0). As an exercise, think about converting it to a time triggered one. What makes it problematic?

Control system

The **plant**, the physical process that is to be controlled

the **environment** in which the plant operates

the **sensors** that measure some variables of the plant and the environment

the **controller** that determines the mode transition structure and selects the time-based inputs to the plant.

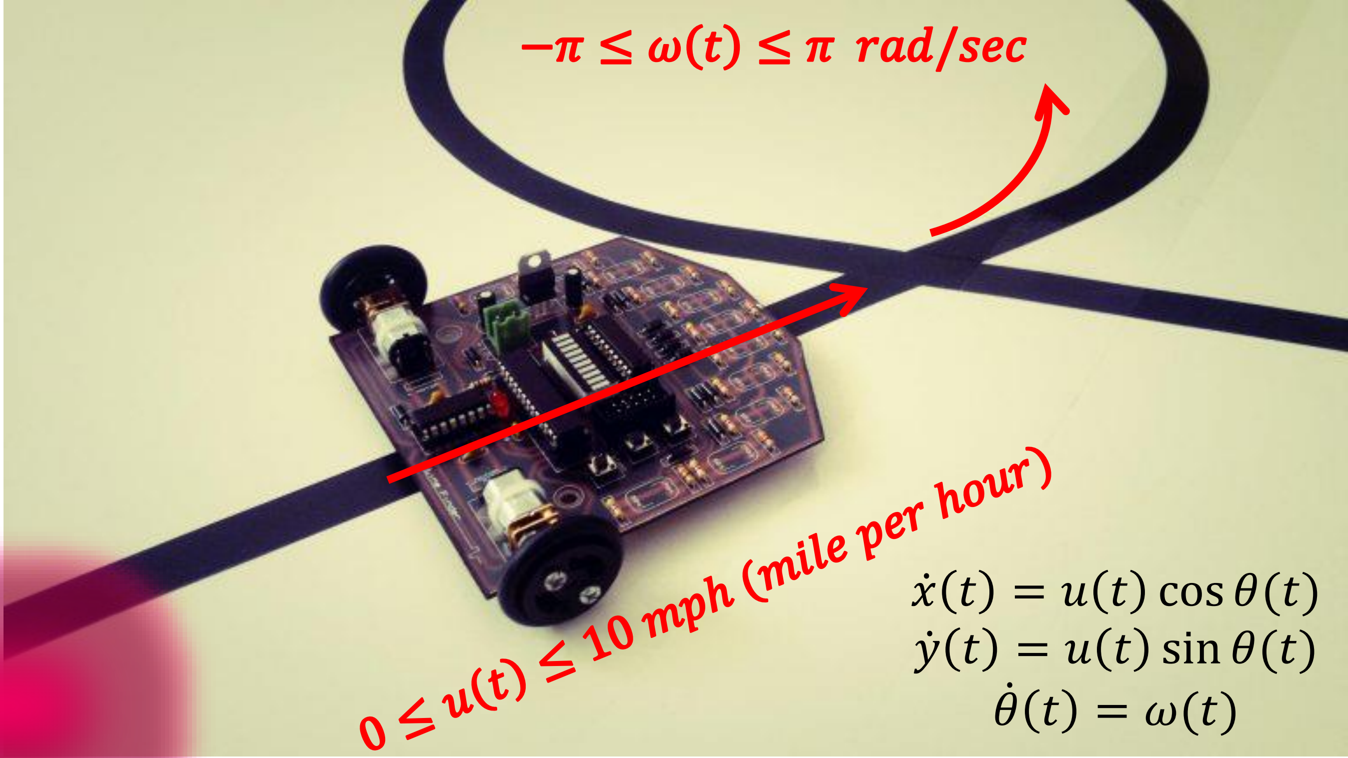
It has two levels:

1. **supervisory control** that determines the mode transition structure
determines which of several strategies should be followed

2. **low-level control** that determines the time-based inputs to the plant
implements the selected strategy

Hybrid systems are ideal for modeling such two-level controllers.

Consider an automated guided vehicle (AGV). We will design a controller so that the vehicle closely follows the track.


$-\pi \leq \omega(t) \leq \pi \;\; rad/sec$
$0 \leq u(t) \leq 10$ mph (mile per hour)

$$\dot{x}(t) = u(t)\cos\theta(t)$$
$$\dot{y}(t) = u(t)\sin\theta(t)$$
$$\dot{\theta}(t) = \omega(t)$$

It has two degrees of freedom: Linear and angular velocity

We ignore the inertia of the vehicle, so we assume that we can instantaneously change the velocity or angular speed.

The two-level controller design is based on a simple idea. The vehicle always moves at its maximum speed of 10 mph. If the vehicle strays too far to the left of the track, the controller steers it towards the right; if it strays too far to the right of the track, the controller steers it towards the left. If the vehicle is close to the track, the controller maintains the vehicle in a straight direction. Thus the controller guides the vehicle in four modes, left, right, straight, and stop. In stop mode, the vehicle comes to a halt.

straight

$$\begin{aligned}\dot{x}(t) &= 10\cos\theta(t)\\ \dot{y}(t) &= 10\sin\theta(t)\\ \dot{\theta}(t) &= 0\end{aligned}$$

left

$$\begin{aligned}\dot{x}(t) &= 10\cos\theta(t)\\ \dot{y}(t) &= 10\sin\theta(t)\\ \dot{\theta}(t) &= \pi\end{aligned}$$

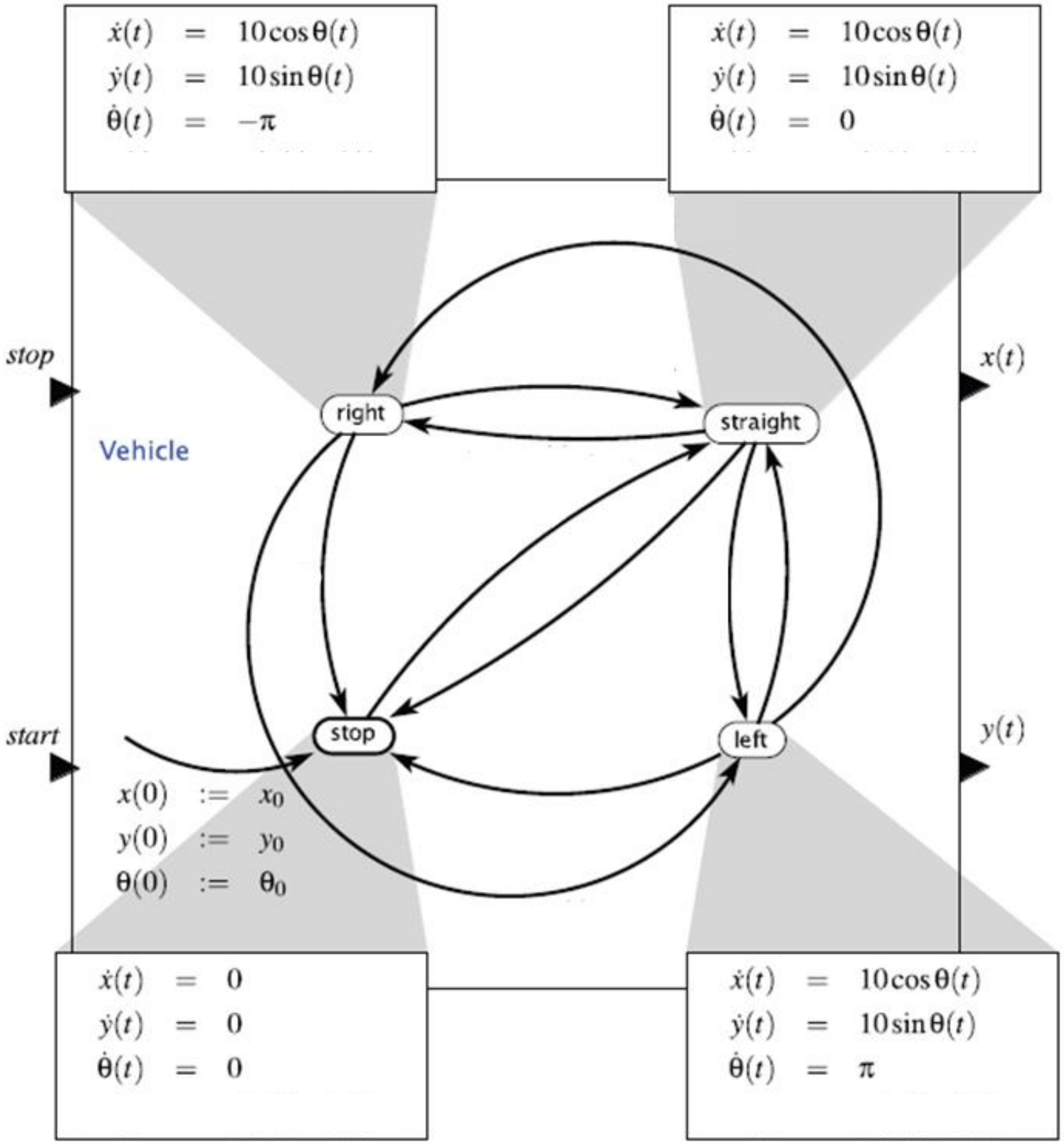
right

$$\begin{aligned}\dot{x}(t) &= 10\cos\theta(t)\\ \dot{y}(t) &= 10\sin\theta(t)\\ \dot{\theta}(t) &= -\pi\end{aligned}$$

stop

$$\begin{aligned}\dot{x}(t) &= 0\\ \dot{y}(t) &= 0\\ \dot{\theta}(t) &= 0\end{aligned}$$

Now, we've designed a low level control, specifying what to do in each decision.

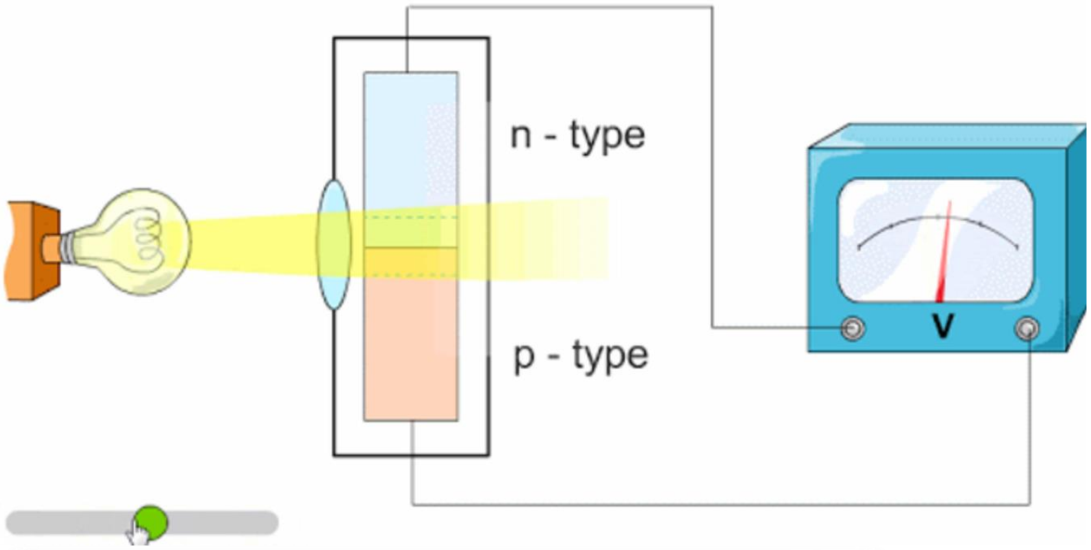
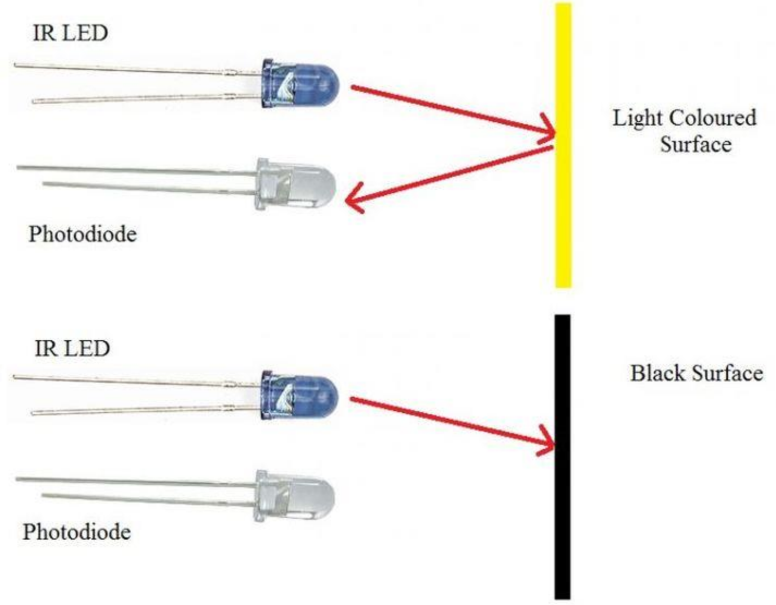
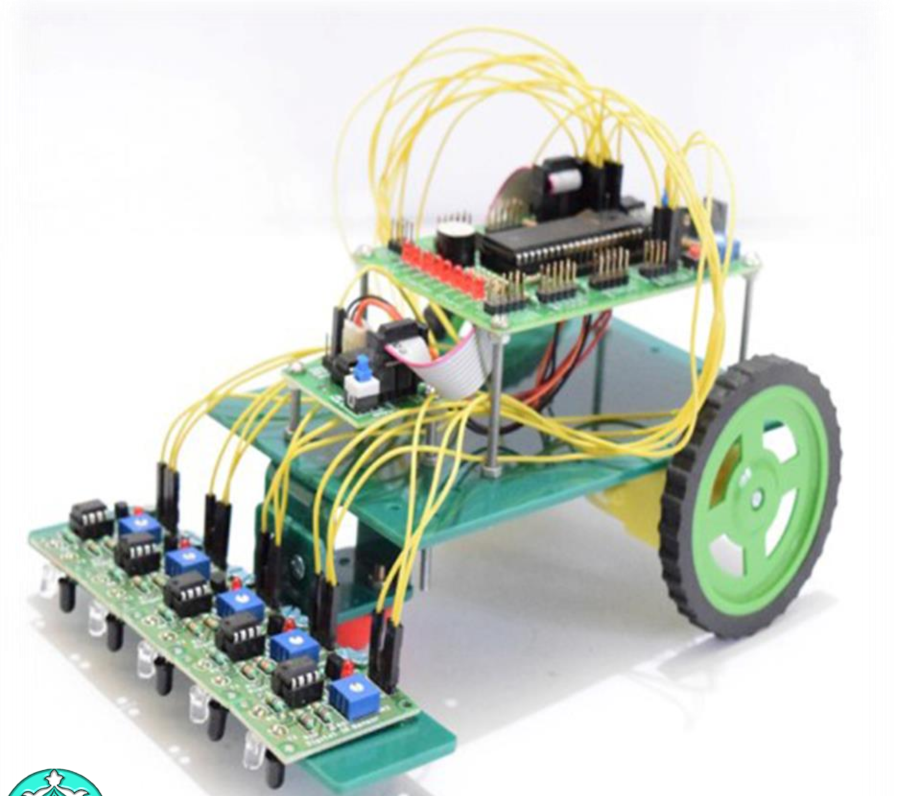
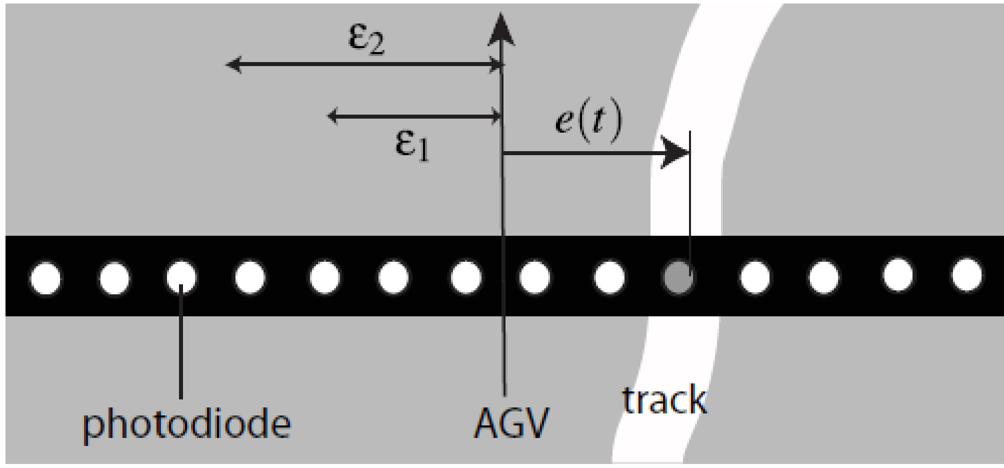


$\dot{x}(t) = 10\cos\theta(t)$, $\dot{y}(t) = 10\sin\theta(t)$, $\dot{\theta}(t) = -\pi$
$\dot{x}(t) = 10\cos\theta(t)$, $\dot{y}(t) = 10\sin\theta(t)$, $\dot{\theta}(t) = 0$
stop
start
Vehicle
right
straight
stop
left
$x(t)$
$y(t)$
$x(0) := x_0$
$y(0) := y_0$
$\theta(0) := \theta_0$
$\dot{x}(t) = 0$, $\dot{y}(t) = 0$, $\dot{\theta}(t) = 0$
$\dot{x}(t) = 10\cos\theta(t)$, $\dot{y}(t) = 10\sin\theta(t)$, $\dot{\theta}(t) = \pi$

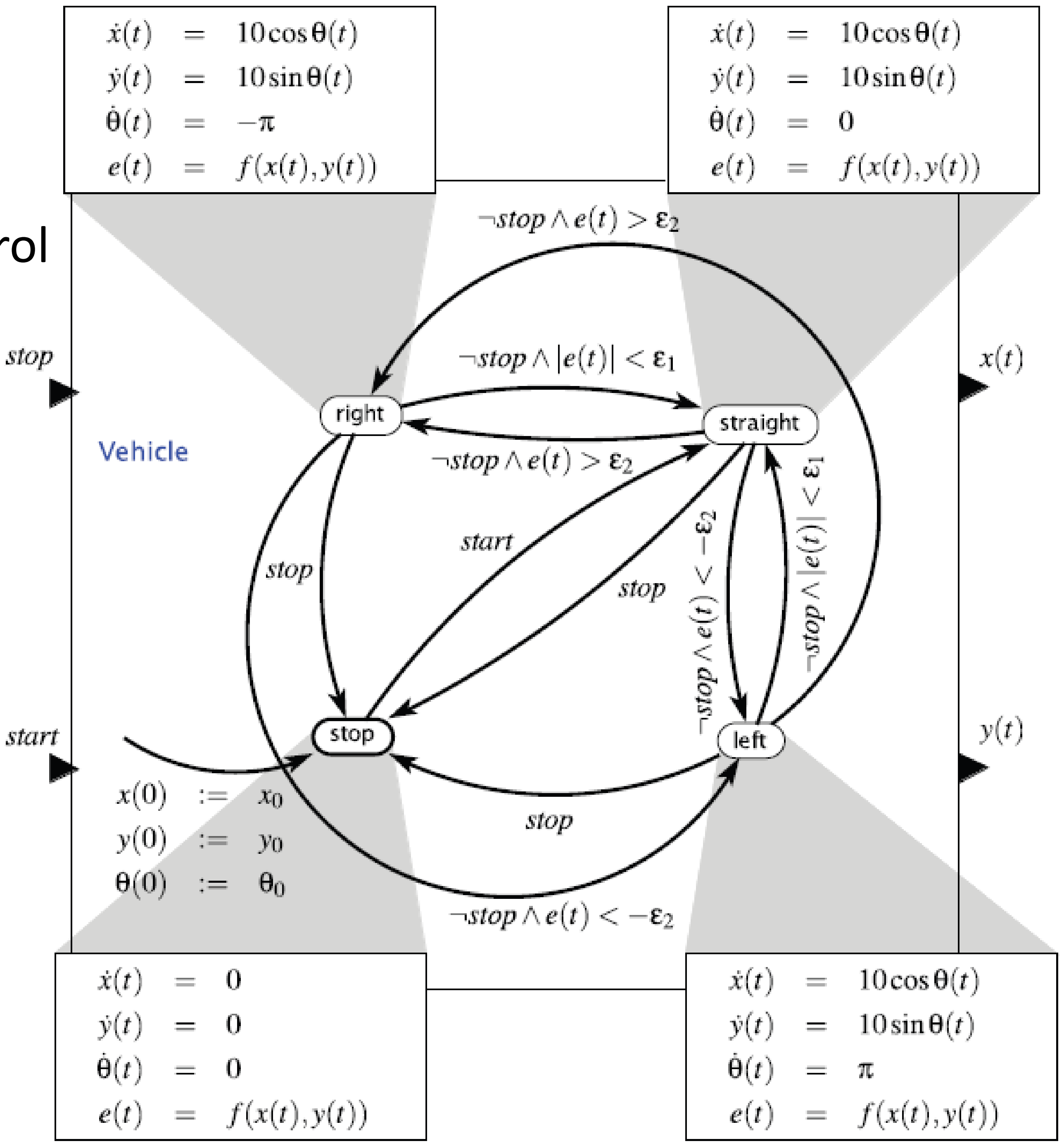
We design the supervisory control governing transitions between modes in such a way that the vehicle closely follows the track, using a sensor that determines how far the vehicle is to the left or right of the track. We can build such a sensor using photodiodes. Let's suppose the track is painted with a light-reflecting color, whereas the floor is relatively dark. Underneath the AGV we place an array of photodiodes as shown in Figure 4.14. The array is perpendicular to the AGV body axis. As the AGV passes over the track, the diode directly above the track generates more current than the other diodes. By comparing the magnitudes of the currents through the different diodes, the sensor estimates the displacement $e(t)$ of the center of the array (hence, the center of the AGV) from the track. We adopt the convention that $e(t) < 0$ means that the AGV is to the right of the track and $e(t) > 0$ means it is to the left. We model the sensor output as a function f of the AGV's position,

$$\forall t, \quad e(t) = f(x(t), y(t)).$$

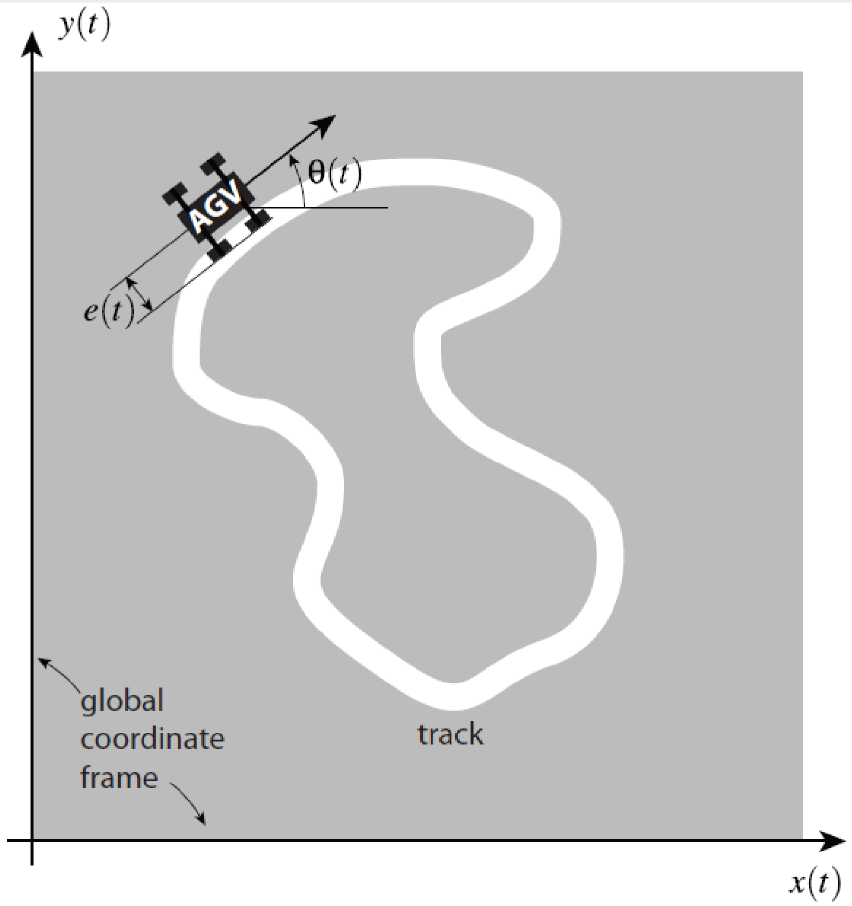
The function f of course depends on the environment—the track. We now specify the supervisory controller precisely. We select two thresholds, $0 < \epsilon_1 < \epsilon_2$, as shown in Figure 4.14. If the magnitude of the displacement is small, $|e(t)| < \epsilon_1$, we consider that the AGV is close enough to the track, and the AGV can move straight ahead, in straight mode. If $e(t) > \epsilon_2$ ($e(t)$ is large and positive), the AGV has strayed too far to the left and must be steered to the right, by switching to right mode. If $e(t) < -\epsilon_2$ ($e(t)$ is large and negative), the AGV has strayed too far to the right and must be steered to the left, by switching to left mode.


n - type
p - type
V

IR LED
Photodiode
Light Coloured Surface
IR LED
Photodiode
Black Surface


ε_2
ε_1
$e(t)$
photodiode
AGV
track

Transitions are added as supervisory control



$\dot{x}(t) = 10\cos\theta(t)$
$\dot{y}(t) = 10\sin\theta(t)$
$\dot{\theta}(t) = -\pi$
$e(t) = f(x(t), y(t))$
$\dot{x}(t) = 10\cos\theta(t)$
$\dot{y}(t) = 10\sin\theta(t)$
$\dot{\theta}(t) = 0$
$e(t) = f(x(t), y(t))$
$\neg stop \wedge e(t) > \varepsilon_2$
$\neg stop \wedge |e(t)| < \varepsilon_1$
$\neg stop \wedge e(t) > \varepsilon_2$
stop
start
Vehicle
right
straight
stop
start
stop
left
$\neg stop \wedge e(t) < -\varepsilon_2$
$\neg stop \wedge |e(t)| < \varepsilon_1$
$x(t)$
$y(t)$
$x(0) := x_0$
$y(0) := y_0$
$\theta(0) := \theta_0$
$\neg stop \wedge e(t) < -\varepsilon_2$
$\dot{x}(t) = 0$
$\dot{y}(t) = 0$
$\dot{\theta}(t) = 0$
$e(t) = f(x(t), y(t))$
$\dot{x}(t) = 10\cos\theta(t)$
$\dot{y}(t) = 10\sin\theta(t)$
$\dot{\theta}(t) = \pi$
$e(t) = f(x(t), y(t))$



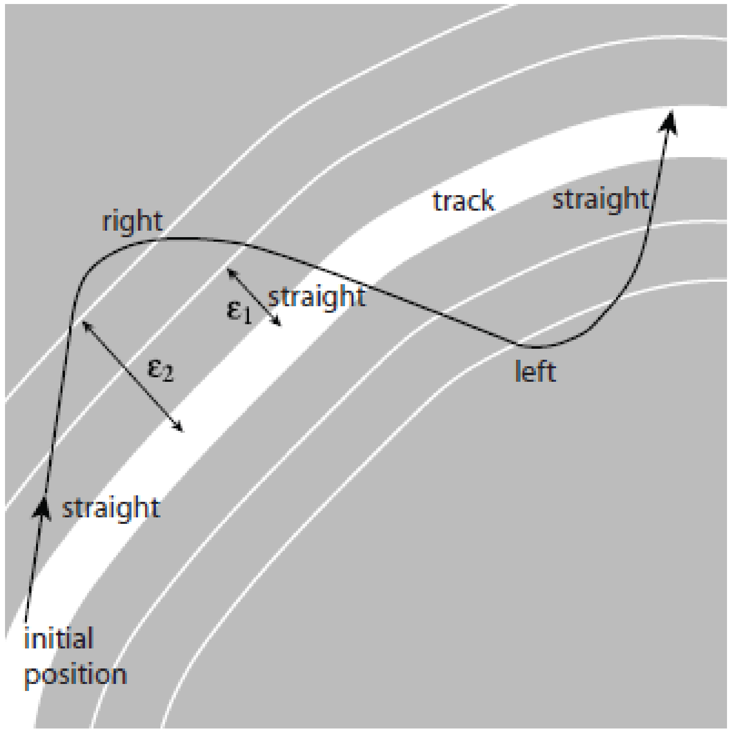


Figure 4.15: A trajectory of the AGV, annotated with modes.

In this example:

1) The plant is described by differential equations $\begin{cases}\dot{x}(t) = u(t)\cos\theta(t)\\ \dot{y}(t) = u(t)\sin\theta(t)\\ \dot{\theta}(t) = \omega(t)\end{cases}$ that govern the evolution of the continuous state at time t, $(x(t);\ y(t);\ \theta(t))$, in terms of the plant inputs u and ω.

2) The environment is the closed track.

3) The sensor is the photodiode array, measuring the position of the AGV relative to the track (e(t)).

4)

i) The supervisory controller comprises the four modes and the guards that determine when to switch between modes.

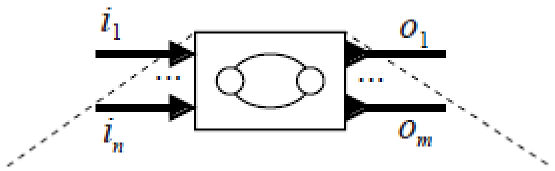
ii) The low-level controller specifies how the time-based inputs to the plant, u and ω, are selected in each mode.

Hybrid Systems
Computer Science
Finite state machines
Control theory
Continuous dynamical systems

Chapter 5

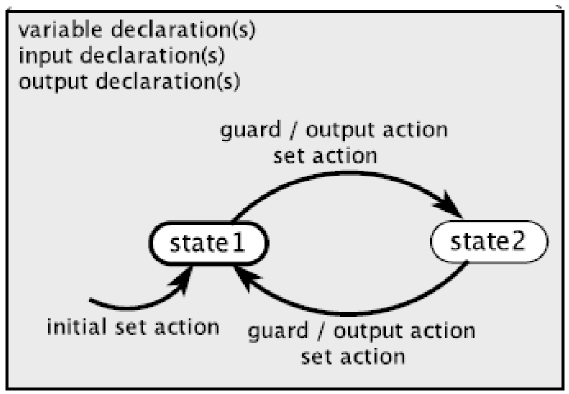
- Composition of State Machines

State machines provide a convenient way to model behaviors of systems. One disadvantage that they have is that for most interesting systems, the number of states is very large, often even infinite. Automated tools can handle large state spaces, but humans have more difficulty with any direct representation of a large state space. A time-honored principle in engineering is that complicated systems should be described as compositions of simpler systems. This chapter gives a number of ways to do this with state machines.



Composition techniques

- concurrent composition
  - machines react simultaneously (synchronous)
  - machines react independently (asynchronous
- hierarchy



Synchrony has different meanings in different contexts. The meaning used in this chapter underlies the synchronous languages.

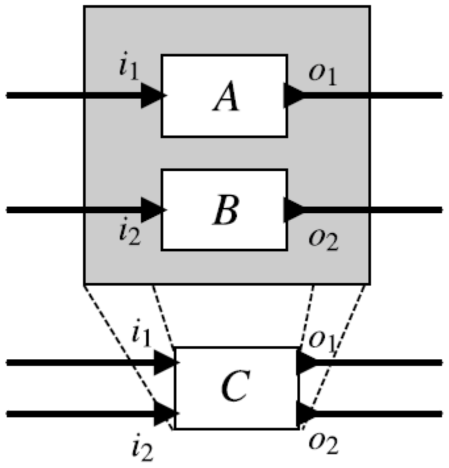
- the outputs of components in a program are (conceptually) simultaneous with their inputs (synchrony hypothesis).
- components in a program execute (conceptually) simultaneously and instantaneously.

concurrent composition

1) Side-by-Side Synchronous Composition:

The state machines being composed do not communicate.

The inputs and outputs of the two actors are disjoint.



Side-by-side composition of two actors.

Synchronous side-by-side composition is simple for several reasons.

- we know that **the environment determines when a state machine reacts**. **In synchronous side-by-side composition, the environment need not be aware that *C* is a composition of two state machines**. Such compositions are modular in the sense that the composition itself becomes a component that can be further composed as if it were itself an atomic component.

- **if the two state machines A and B are deterministic, then the synchronous side-by-side composition is also deterministic**. **We say that a property is compositional if a property held by the components is also a property of the composition**. For synchronous side-by-side composition, **determinism** is a compositional property.

- a synchronous side-by-side composition of finite state machines is itself an FSM.

Notice that (s1; s4) and (s2; s3) are not **reachable states**.

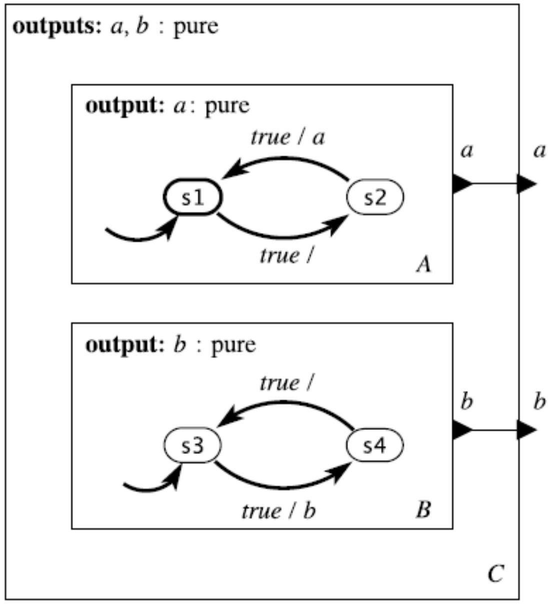


Figure 5.3: Example of side-by-side composition of two actors.

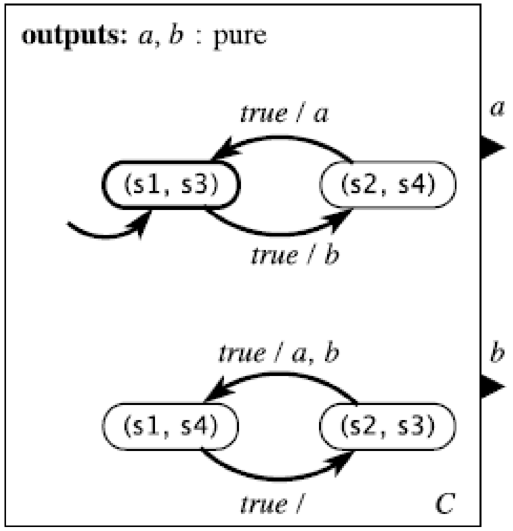


Figure 5.4: Single state machine giving the semantics of synchronous side-by-side composition of the state machines in Figure 5.3.

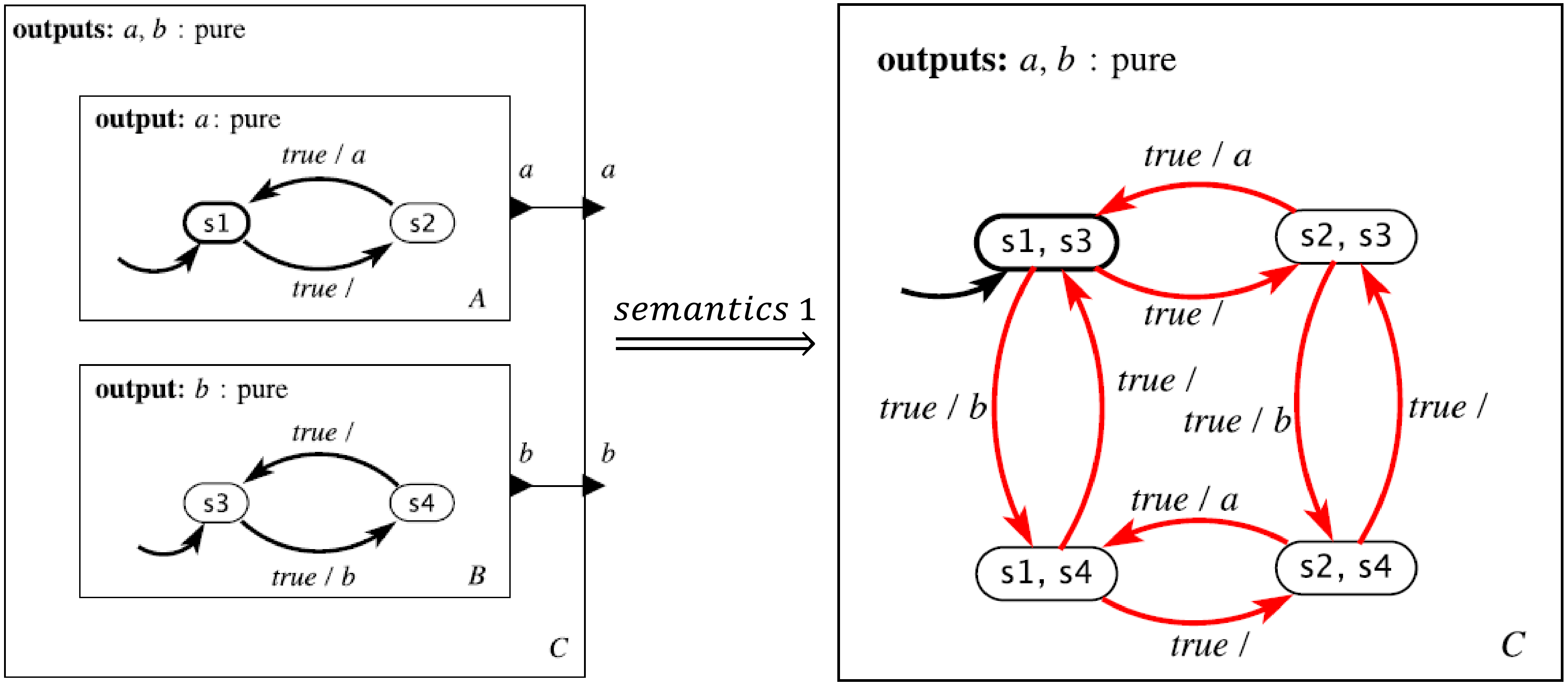
2) Side-by-Side Asynchronous Composition:

In an **asynchronous composition** of state machines, the component machines react independently. This statement is rather vague, and in fact, it has several different interpretations. Each interpretation gives a semantics to the composition. The key to each semantics is how to define a reaction of the composition ***C***. Two possibilities are:

Semantics 1) A reaction of ***C*** is a reaction of one of A or B, where the choice is nondeterministic. (interleaving semantics, i.e. A or B never react simultaneously. Their reactions are interleaved in some order.)

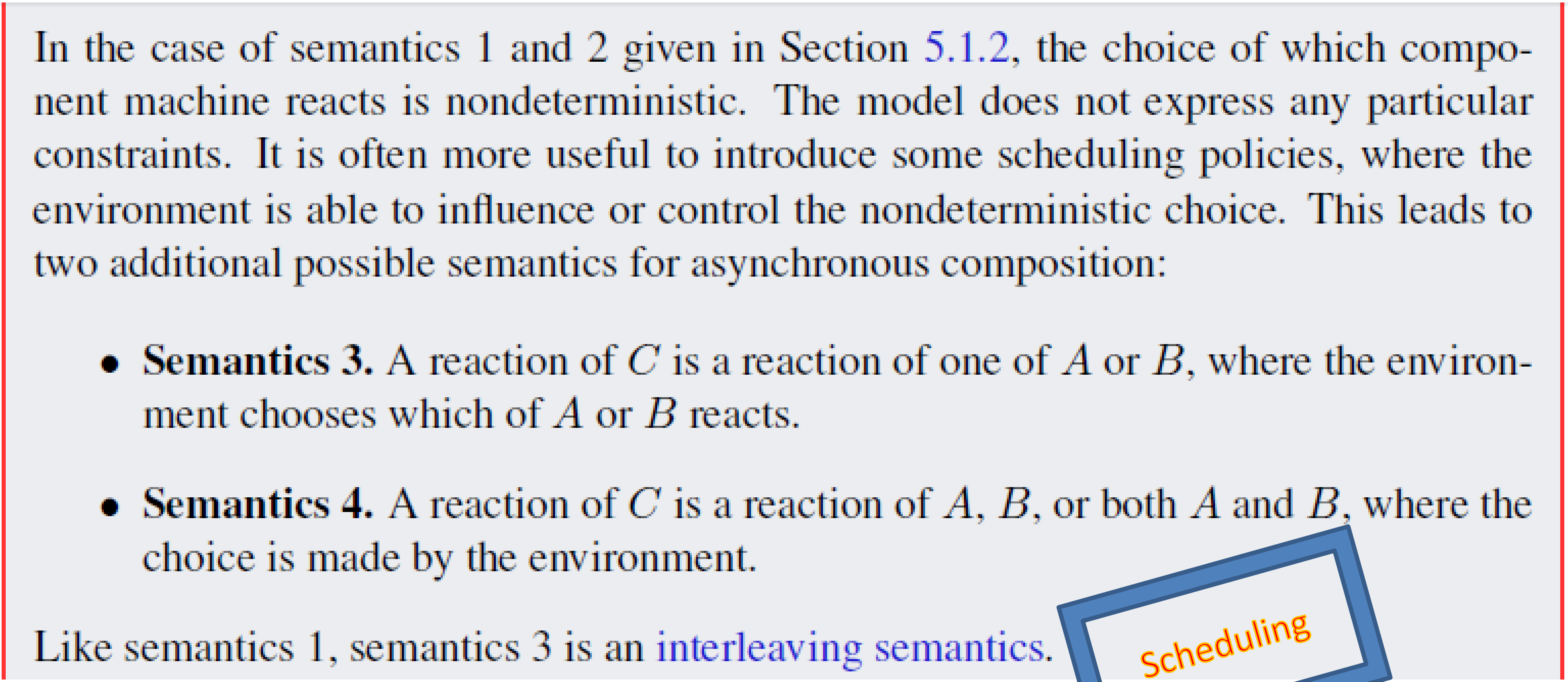
Semantics 2) A reaction of ***C*** is a reaction of A, B, or both A and B, where the choice is nondeterministic. A variant of this possibility might allow neither to react.

under these semantics machines **A and B may completely miss input events.** That is, an input to *C* destined for machine A may be present in a reaction where the nondeterministic choice results in B reacting rather than A. If this is not desirable, then some control over scheduling or synchronous composition becomes a better choice.



red arrows show nondeterminism

This machine is nondeterministic. Note that if we had chosen semantics 2, then it would also be able to move to (s2; s4).



In the case of semantics 1 and 2 given in Section 5.1.2, the choice of which component machine reacts is nondeterministic. The model does not express any particular constraints. It is often more useful to introduce some scheduling policies, where the environment is able to influence or control the nondeterministic choice. This leads to two additional possible semantics for asynchronous composition:

- **Semantics 3.** A reaction of C is a reaction of one of A or B, where the environment chooses which of A or B reacts.
- **Semantics 4.** A reaction of C is a reaction of A, B, or both A and B, where the choice is made by the environment.

Like semantics 1, semantics 3 is an interleaving semantics.

Scheduling

Shared Variables

An **extended state machine** has **local variables** that can be **read and written** as part of taking a transition. Sometimes it is useful when composing state machines to allow these variables to be shared among a group of machines. In particular, such shared variables can be useful for **modeling interrupts and threads**. Many complications arise, including the memory consistency model and the notion of atomic operations.

Consider two servers that can receive requests from a network. Each request requires an unknown amount of time to service, so the servers share a queue of requests. If one server is busy, the other server can respond to a request, even if the request arrives at the network interface of the first server.

shared variable ***pending***: number of pending job requests

When a request arrives at the composite machine *C*, one of the two servers is nondeterministically chosen to react, assuming asynchronous composition under semantics 1. If that server is idle, then it proceeds to serve the request. If the server is serving another request, then one of two things can happen:

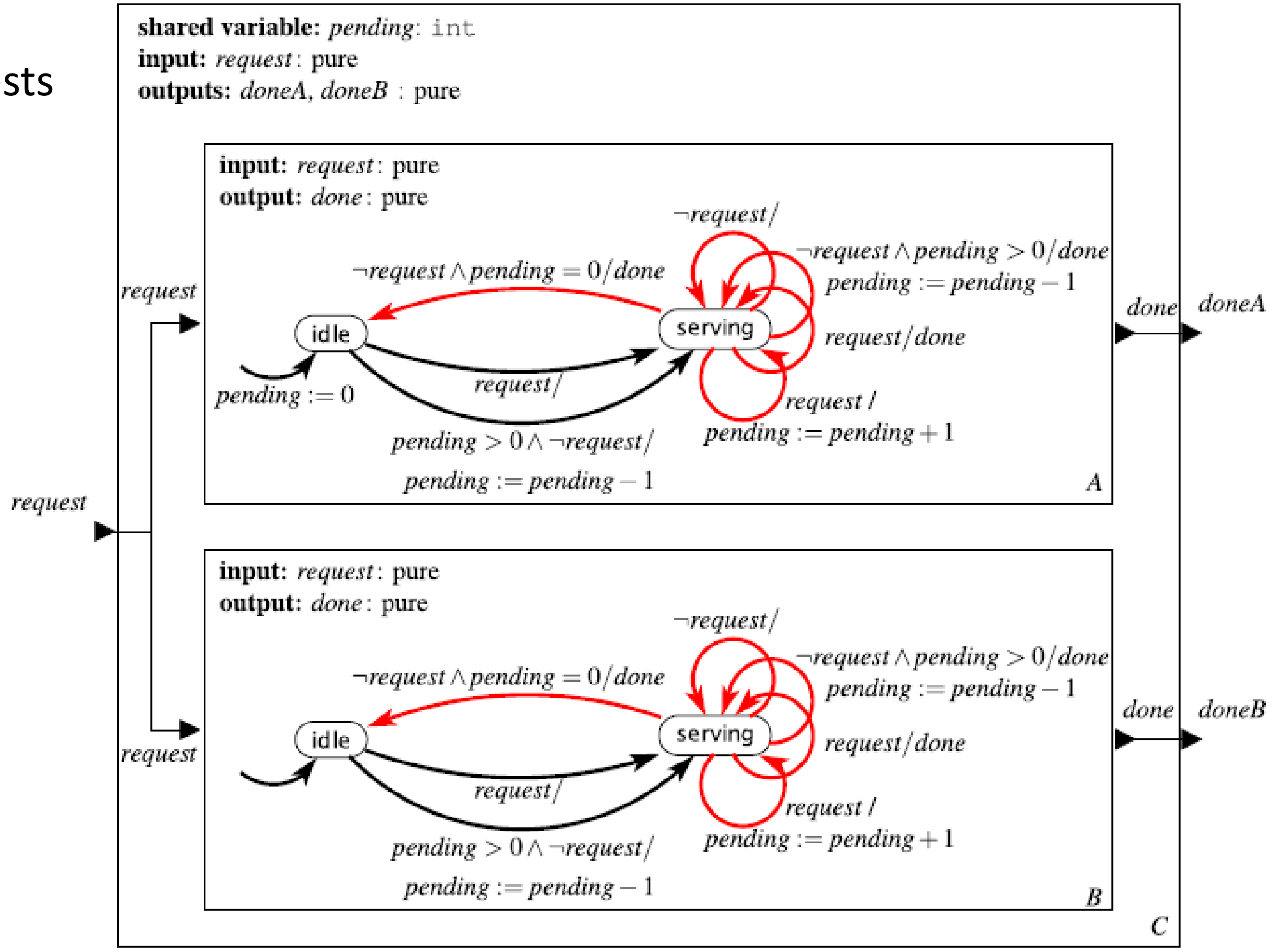


Figure 5.6: Model of two servers with a shared task queue, assuming asynchronous composition under semantics 1.

it can coincidentally finish serving the request it is currently serving, issuing the output done, and proceed to serve the new one, or it can increment the count of pending requests and continue to serve the current request. The choice between these is nondeterministic, to model the fact that the time it takes to service a request is unknown.

If *C* reacts when there is no request, then again either server A or B will be selected non-deterministically to react. If the server that reacts is idle and there are one or more pending requests, then the server transitions to serving and decrements the variable pending. If the server that reacts is not idle, then one of three things can happen.

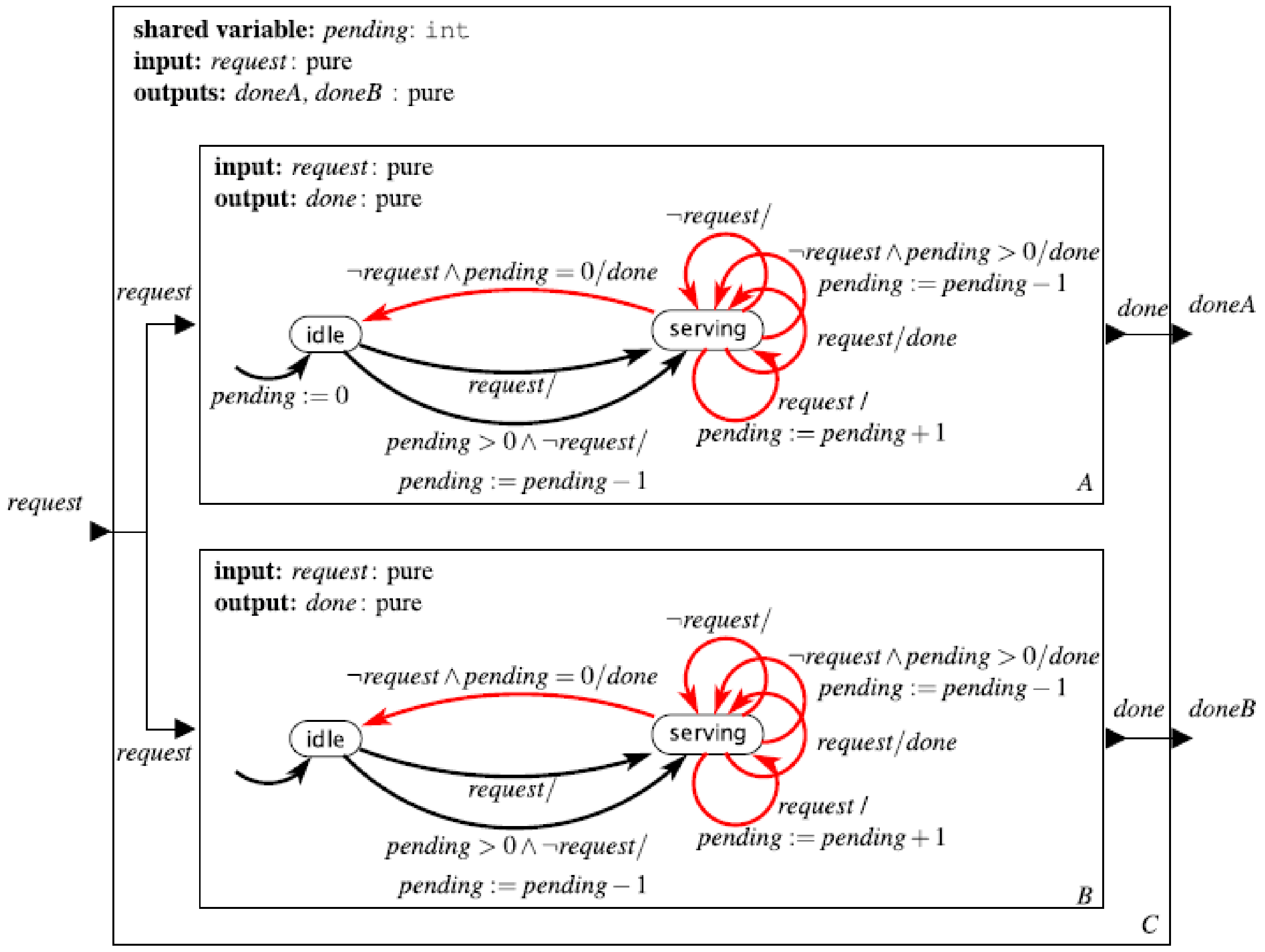


Figure 5.6: Model of two servers with a shared task queue, assuming asynchronous composition under semantics 1.

It may continue serving the current request, in which case it simply transitions on the self transition back to serving. Or it may finish serving the request, in which case it will transition to idle if there are no pending requests, or transition back to serving and decrement pending if there are pending requests.

Notice that because of the interleaving semantics, accesses to the shared variable are atomic operations

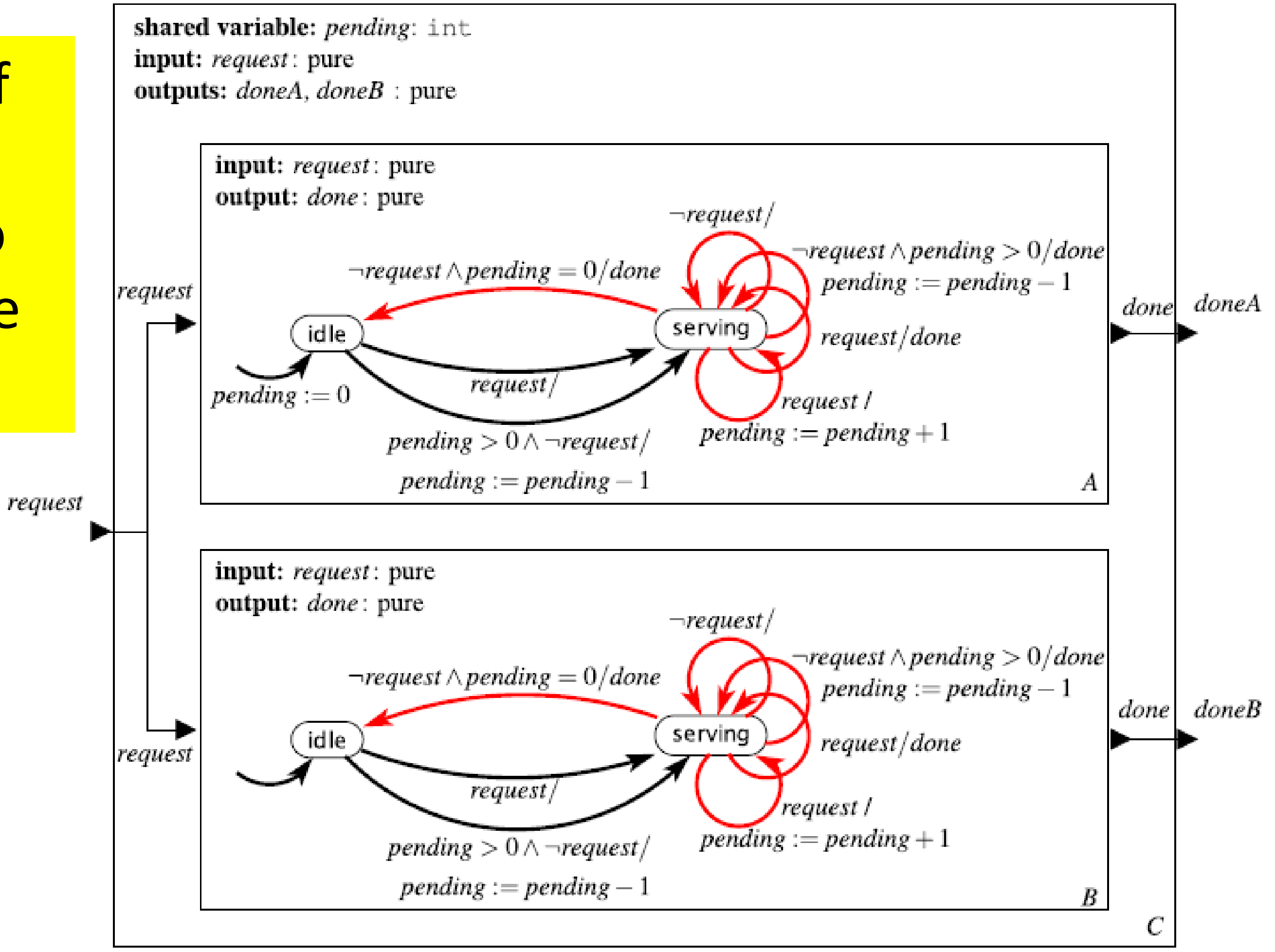
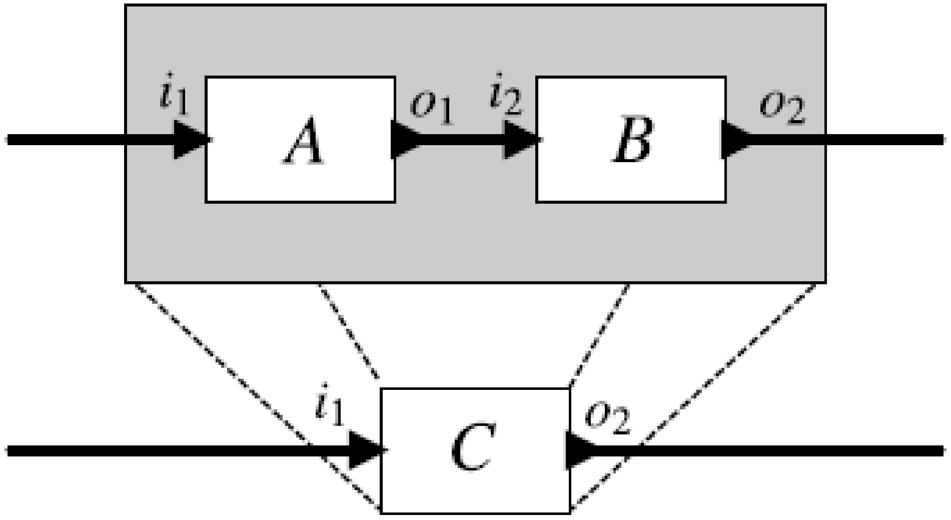


Figure 5.6: Model of two servers with a shared task queue, assuming asynchronous composition under semantics 1.

The choice of semantics 1 is reasonable in this case because the input goes to both of the component machines, so regardless of which component machine reacts, no input event will be missed. However, this semantics would not work if the two machines had independent inputs, because then requests could be missed. Semantics 2 can help prevent that, but what strategy should be used by the environment to determine which machine reacts? What if the two independent inputs both have requests present at the same reaction of *C*? If we choose semantics 4 to allow both machines to react simultaneously, then what is the meaning when both machines update the shared variable? The updates are no longer atomic, as they are with an interleaving semantics. **Note further that** choosing asynchronous composition under semantics 1 allows behaviors that do not make good use of idle machines. In particular, suppose that machine A is serving, machine B is idle, and a request arrives. If the nondeterministic choice results in machine A reacting, then it will simply increment pending. Not until the nondeterministic choice results in B reacting will the idle machine be put to use. In fact, semantics 1 allows behaviors that never use one of the machines.

Cascade Composition (serial composition)
The output of A must be valid is an input for B.

For cascade composition, if we wish the composition to be asynchronous, then we need to introduce some machinery for buffering the data that is sent from A to B. We defer discussion of such asynchronous composition to Chapter 6, where **dataflow** and **process network** models of computation will provide such asynchronous composition. In this chapter, we will only consider synchronous composition for cascade systems.



Cascade composition of two actors.

(s1,s4) and (s2, s3) are not reachable states!

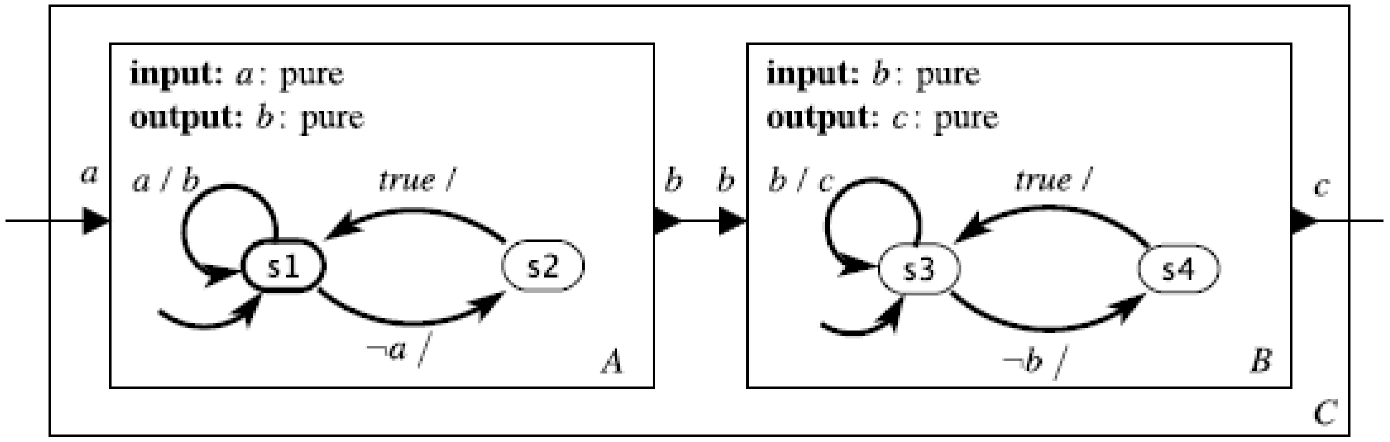


Figure 5.8: Example of a cascade composition of two FSMs.

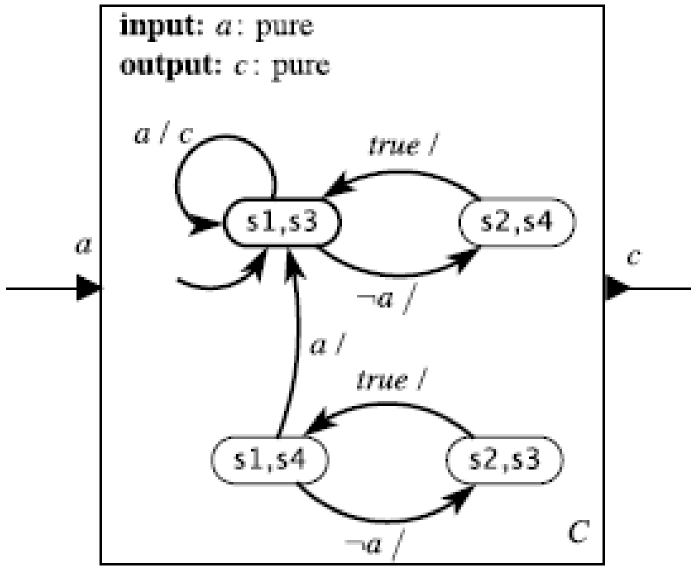


Figure 5.9: Semantics of the cascade composition of Figure 5.8, assuming synchronous composition.

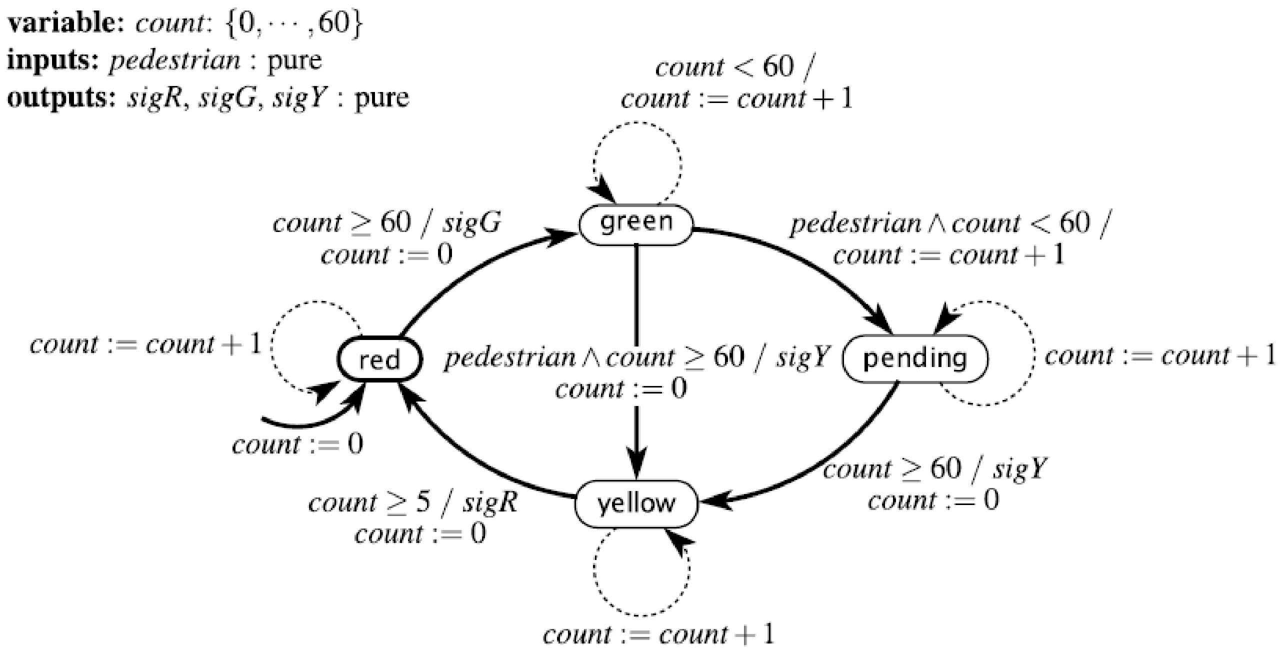

variable: count: {0, ⋯ ,60}
inputs: pedestrian : pure
outputs: sigR, sigG, sigY : pure
count < 60 / count := count + 1
green
count ≥ 60 / sigG count := 0
pedestrian ∧ count < 60 / count := count + 1
count := count + 1
red
pedestrian ∧ count ≥ 60 / sigY count := 0
pending
count := count + 1
count := 0
count ≥ 60 / sigY count := 0
count ≥ 5 / sigR count := 0
yellow
count := count + 1

Figure 3.10: Extended state machine model of a traffic light controller that keeps track of the passage of time, assuming it reacts at regular intervals.

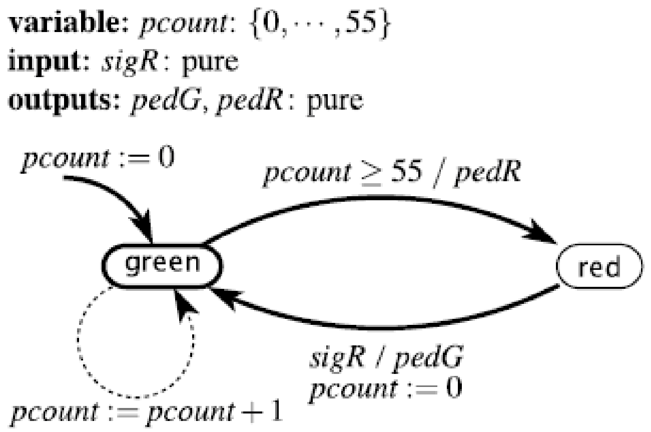
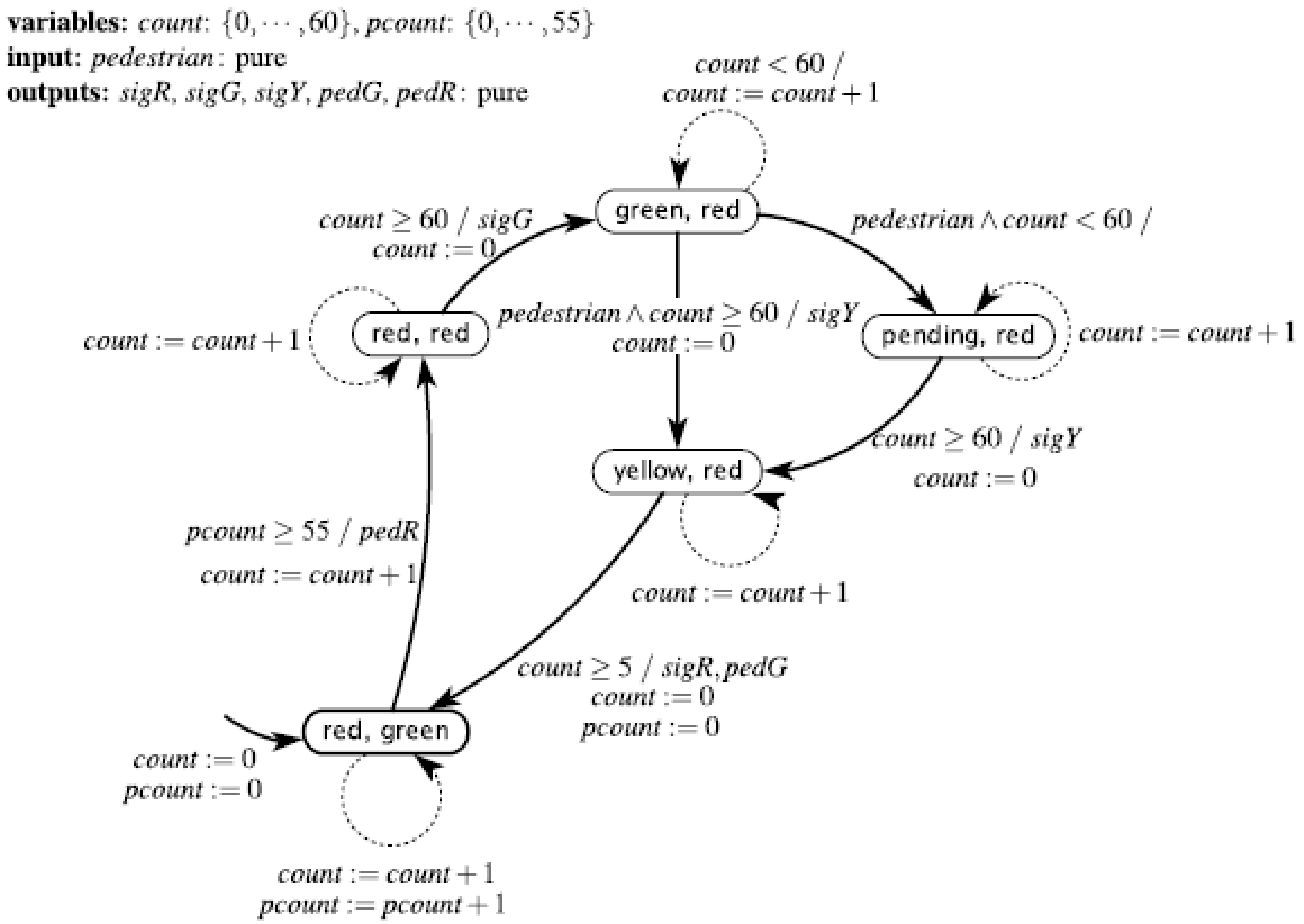


Figure 5.10: A model of a pedestrian crossing light, to be composed in a synchronous cascade composition with the traffic light model of Figure 3.10.

We want to compose these models

The output $sigR$ of the traffic light can provide the input $sigR$ of the pedestrian light.



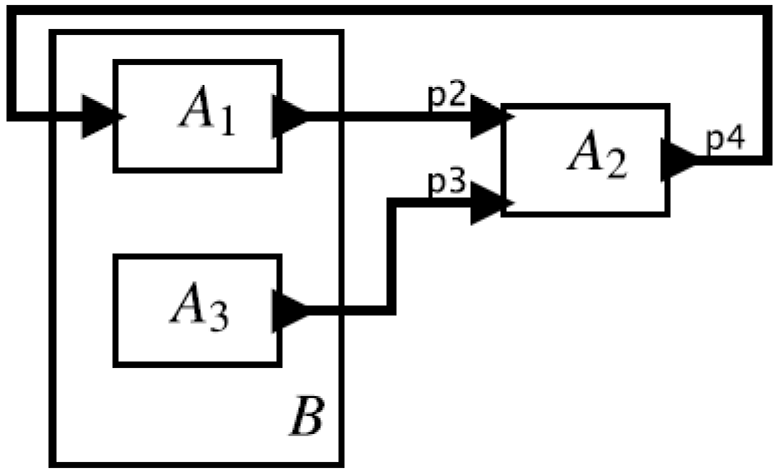
variables: count: {0, ··· ,60}, pcount: {0, ··· ,55}
input: pedestrian: pure
outputs: sigR, sigG, sigY, pedG, pedR: pure
count < 60 / count := count + 1
green, red
count ≥ 60 / sigG count := 0
pedestrian ∧ count < 60 /
red, red
count := count + 1
pedestrian ∧ count ≥ 60 / sigY count := 0
pending, red
count := count + 1
count ≥ 60 / sigY count := 0
yellow, red
pcount ≥ 55 / pedR count := count + 1
count := count + 1
count ≥ 5 / sigR, pedG count := 0 pcount := 0
red, green
count := 0 pcount := 0
count := count + 1 pcount := pcount + 1

Figure 5.11: Semantics of a synchronous cascade composition of the traffic light model of Figure 3.10 with the pedestrian light model of Figure 5.10.

Note that unsafe states, such as (green, green), are not reachable states, and hence are not shown.

General Composition

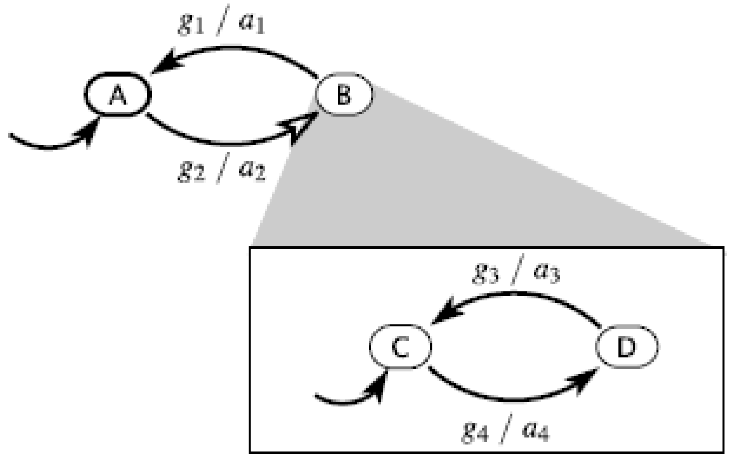
Side-by-side and cascade composition provide the basic building blocks for building more complex compositions of machines.



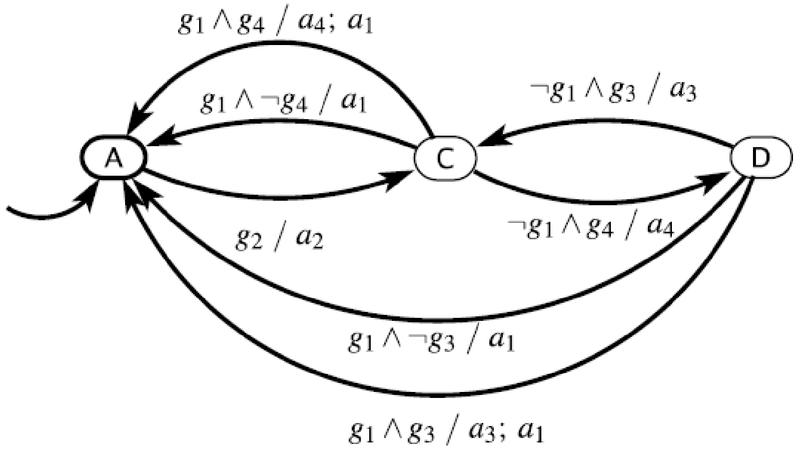
which machine should react first, B or A_2? This conundrum will be resolved in the next chapter when we explain the synchronous-reactive model of computation.

Hierarchical State Machines

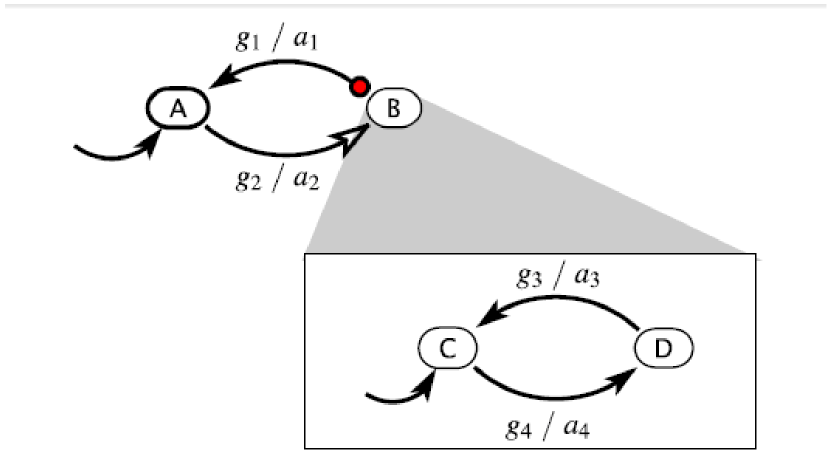
State refinements can be another FSM



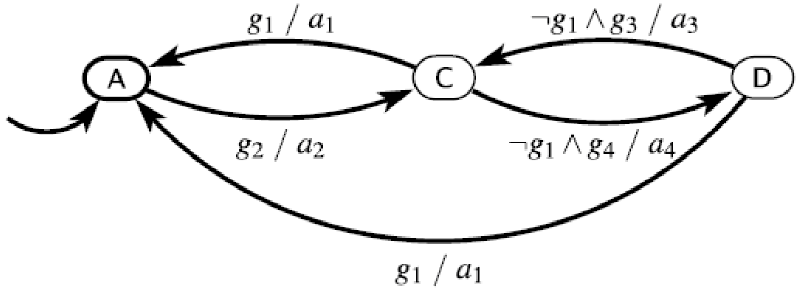
$\equiv$

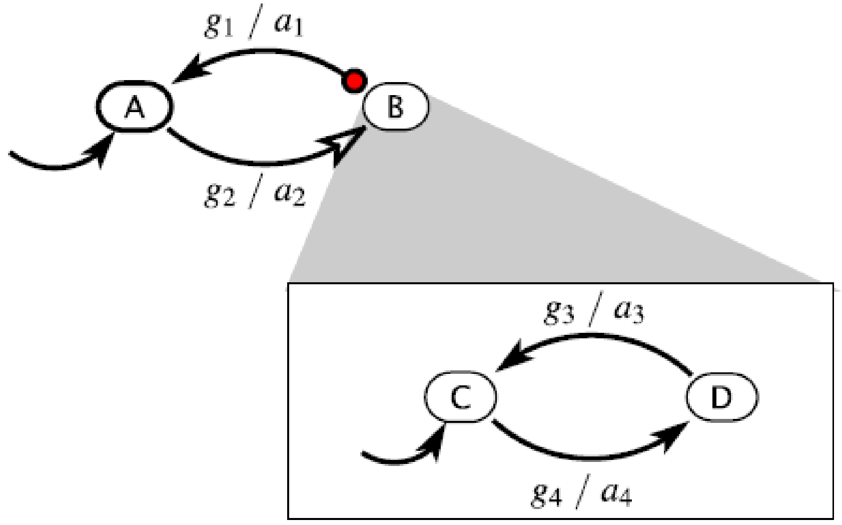

$g_1 \wedge g_4 / a_4; a_1$
$g_1 \wedge \neg g_4 / a_1$
$\neg g_1 \wedge g_3 / a_3$
g_2 / a_2
$\neg g_1 \wedge g_4 / a_4$
$g_1 \wedge \neg g_3 / a_1$
$g_1 \wedge g_3 / a_3; a_1$
A
C
D

Consider the hierarchical FSM depicted below is in state C. A subtle question is what happens if both guards g_1 and g_4 evaluate to true. Such subtle questions help account for the proliferation of different variants of Statecharts.
We choose a particular semantics that has attractive modularity properties. In this semantics, a reaction of a hierarchical FSM is defined in a depth-first fashion. The deepest refinement of the current state reacts first, then its container state machine, then its container, etc. this means that if the machine is in state B, then the refinement machine reacts first.

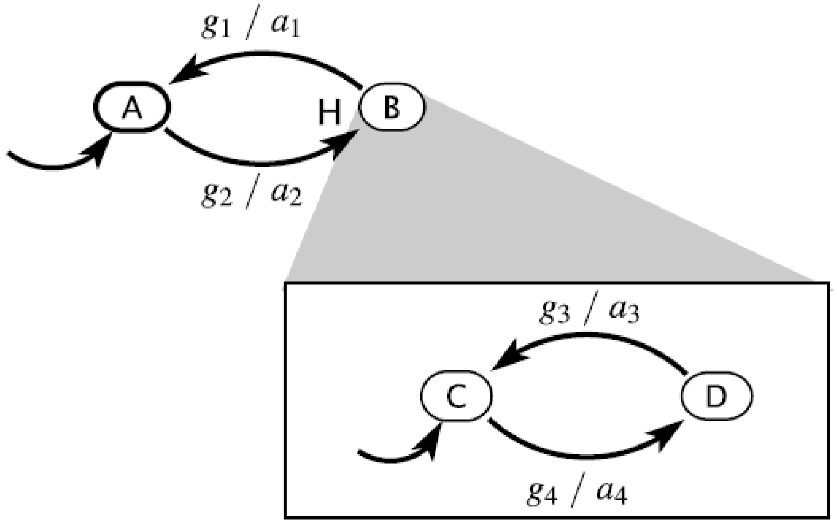


$\equiv$

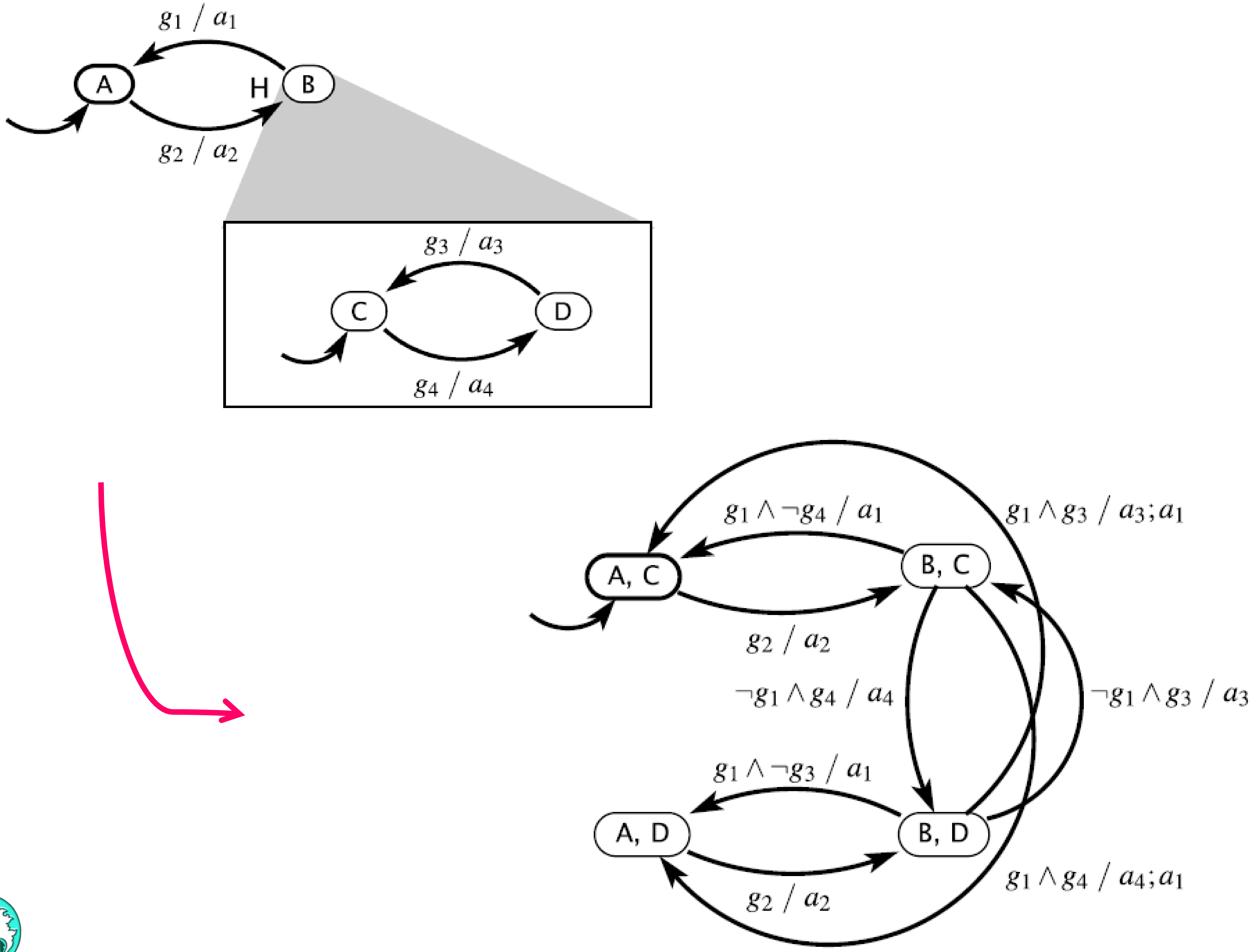




The transition from A to B is called a reset transition because the destination refinement is reset to its initial state, regardless of where it had previously been. A reset transition is indicated in our notation with a hollow arrowhead at the destination end of a transition.



the transition from A to B is a history transition, an alternative to a reset transition. In our notation, a solid arrowhead denotes a history transition. It may also be marked with an “H” for emphasis. When a history transition is taken, the destination refinement resumes in whatever state it was last in (or its initial state on the first entry).



g_1 / a_1
A
H
B
g_2 / a_2
g_3 / a_3
C
D
g_4 / a_4
$g_1 \wedge \neg g_4$ / a_1
$g_1 \wedge g_3$ / $a_3;a_1$
A, C
B, C
g_2 / a_2
$\neg g_1 \wedge g_4$ / a_4
$\neg g_1 \wedge g_3$ / a_3
$g_1 \wedge \neg g_3$ / a_1
A, D
B, D
g_2 / a_2
$g_1 \wedge g_4$ / $a_4;a_1$