Logic, Learning, and Decision
Lecture 1

Bengt Lennartson

Division of Systems and Control
Chalmers University of Technology

Outline

- Discrete event systems
- Modelling
- Design of feedback controllers

What is a discrete event system?

Systems of Systems

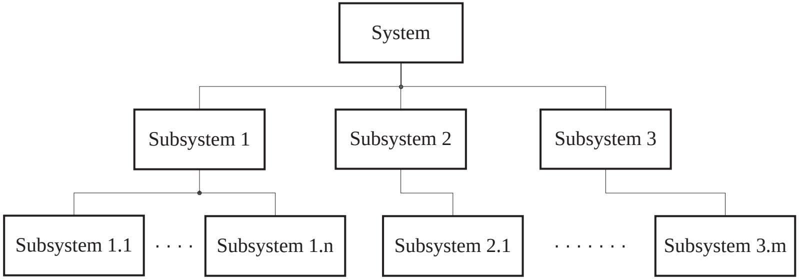


Figure 1.1 A system consists of a number of interacting components/subsystems which together perform a common objective.

Inputs and Outputs

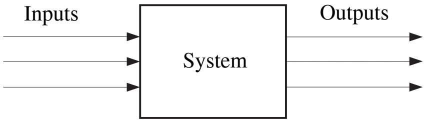


Figure 1.2 A system is influenced by inputs, and the resulting system behavior can be observed by its outputs.

Continuous State-Space Models

Continuous time

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t), t) \\ y(t) &= g(x(t), u(t), t) \end{aligned}$$

Discrete time

$$\begin{aligned} x(t_{k+1}) &= f(x(t_k), u(t_k), t_k) \\ y(t_k) &= g(x(t_k), u(t_k), t_k) \end{aligned}$$

Discrete State-Space Models

Discrete states in a set Ω, e.g. $\Omega = \mathbb{N}$

$$\begin{aligned} x(t_k^+) &= f(x(t_k), u(t_k), t_k) \\ y(t_k) &= g(x(t_k), u(t_k), t_k) \end{aligned}$$

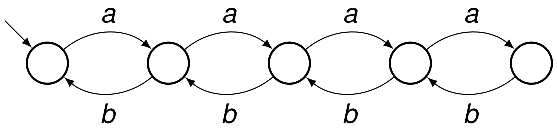
$x \in \Omega_x = \{x^{(1)}, x^{(2)}, x^{(3)}, \ldots\}$, $u \in \Omega_u$, $y \in \Omega_y$.

Example: Buffer $\Omega_x = \{0, 1, 2, 3, 4\}$. When an element is moved to or from the buffer, the state update becomes

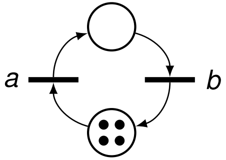
$$x(t_k^+) = x(t_k) + u(t_k)$$

Limited Buffer – Graphical models

Automaton



Petri Net



Limited Buffer – Predicate Transition Model

Discrete state space: $x \in X = \{0, 1, 2, 3, 4\}$

$$\begin{array}{ll} \acute{x} = x + 1 & \text{when event } a \text{ is executed and } x < 4 \\ \acute{x} = x - 1 & \text{when event } b \text{ is executed and } x > 0 \end{array}$$

Include the event variable $e \in \{a, b\}$, and the buffer can be expressed by the predicate

$$(\acute{x} = x + 1 \wedge x < 4 \wedge e = a) \vee (\acute{x} = x - 1 \wedge x > 0 \wedge e = b)$$

Temperature

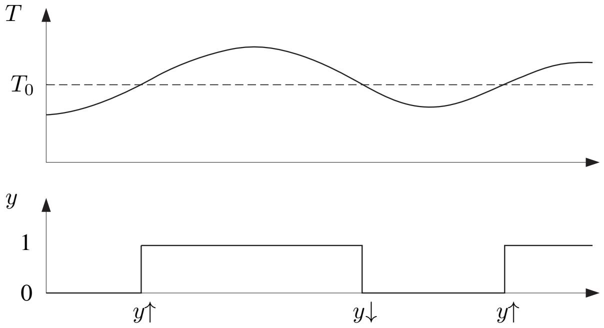


Figure 1.3 Continuous-time temperature signal T and a related discrete binary signal y, where $y = 1$ when $T \geq T_0$ and $y = 0$ when $T < T_0$.

Automaton for Temperature

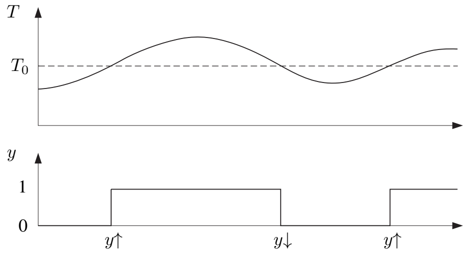


Figure 1.3 Continuous-time temperature signal T and a related discrete binary signal y, where $y = 1$ when $T \geq T_0$ and $y = 0$ when $T < T_0$.

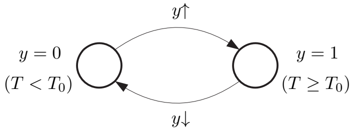


Figure 1.4 An automaton modelling the discrete binary signal y in Fig. 1.3. The circles represent the states and the directed arcs represent the transitions and the associated discrete events.

Summary: Discrete Event Systems

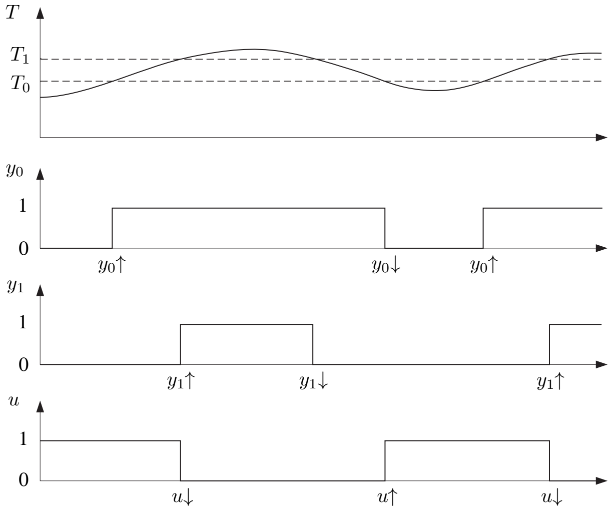
Summary: Discrete event systems

- A discrete event system consists of a set of discrete states, where the transitions between the different states and the associated instantaneous events constitute the dynamic system behavior.
- Typical examples of discrete events are mode changes e.g. to start an operation, send and receive messages, discrete signal changes and state jumps.
- Although a discrete event system normally consists of a number of small subsystems, the total integrated system often becomes quite large in terms of the number of states ($> 10^6$).
- Discrete event models typically represent abstractions and simplifications, which implies that both complex physical and software systems can be handled with reasonable effort.

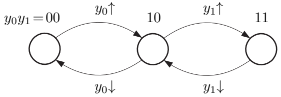
Outline

- Discrete event systems
- Modelling
- Design of feedback controllers

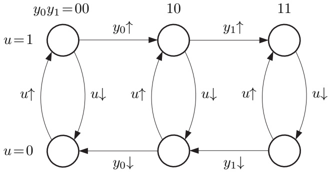
Temperature Control



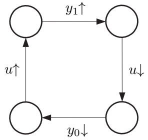
Automata for Temperature Control Example



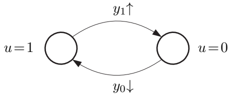
(a)



(b)



(c)



(d)

Tank

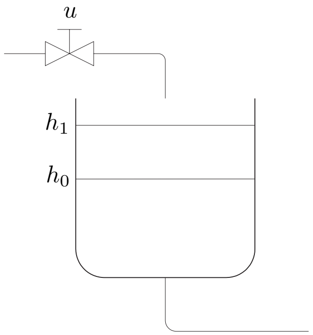


Figure 1.7 Tank level controlled by a binary valve control signal u, based on two level sensors h_0 and h_1.

Cylinder

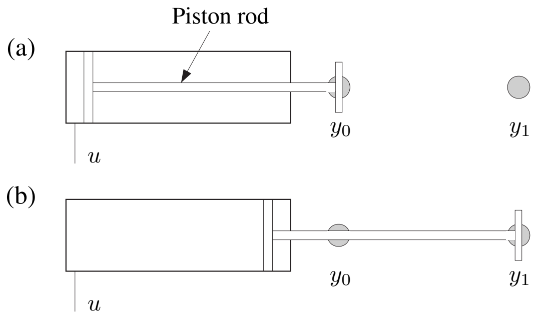


Figure 1.8 (a) Cylinder in its 0-position, indicated by the binary signal y_0 (gray circle), and (b) its 1-position indicated by y_1. The piston rod moves forward to the 1-position when $u = 1$ and returns to the 0-position when $u = 0$.

AGV Automatic Guided Vehicle

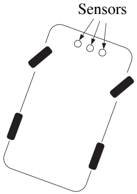


Figure 1.9 Automated guided vehicle with three binary sensor signals (left, zero, right) and three directions (left, straight on, right) of the steering wheels.

$$\begin{tikzcd}
y_\ell\, y_0 y_r = 100 \arrow[r, bend left, "y_\ell\downarrow"] & 000 \arrow[l, bend left, "y_\ell\uparrow"] \arrow[r, bend left, "y_0\uparrow"] & 010 \arrow[l, bend left, "y_0\downarrow"] \arrow[r, bend left, "y_0\downarrow"] & 000 \arrow[l, bend left, "y_0\uparrow"] \arrow[r, bend left, "y_r\uparrow"] & 001 \arrow[l, bend left, "y_r\downarrow"]
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
q_1 \; (u_\ell = 1) \arrow[r, bend left, "y_0\uparrow"] & q_2 \; (u_0 = 1) \arrow[l, bend left, "y_\ell\uparrow"] \arrow[r, bend left, "y_r\uparrow"] & q_3 \; (u_r = 1) \arrow[l, bend left, "y_0\uparrow"]
\end{tikzcd}$$

(b)

Summary: Modelling

Summary: Modelling

- Generic discrete event models can achieved, which can be used to describe completely different physical behaviors from different application areas.
- Models for control purposes can often be simplified, where all details of the system behavior are not necessary to include in the design of a control function.

Outline

- Discrete event systems
- Modelling
- Design of feedback controllers

Closed Loop System

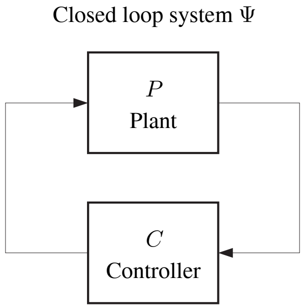


Figure 1.11 Closed loop system Ψ, where the plant P is influenced by the controller C based on feedback from the plant.

Closed Loop System Model

Continuous feedback

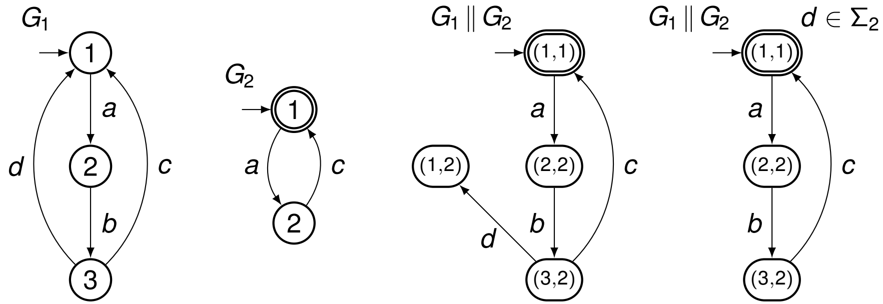
$$\Psi(P, C) = \frac{P(s)C(s)}{1 + P(s)C(s)}$$

P = plant, C = controller, Ψ = closed loop system

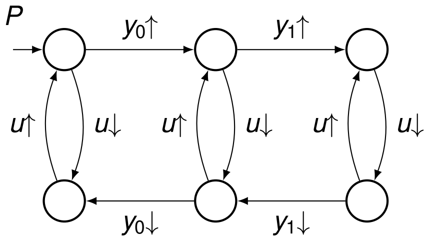
Discrete event closed loop system by synchronization

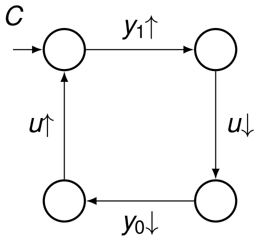
$$\Psi(P, C) = P||C$$

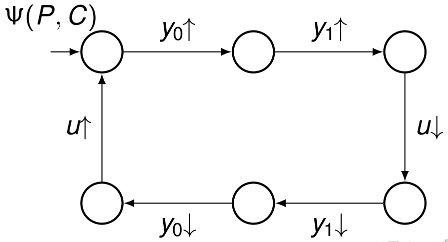
Synchronization of automata



Temperature Example







Robot System

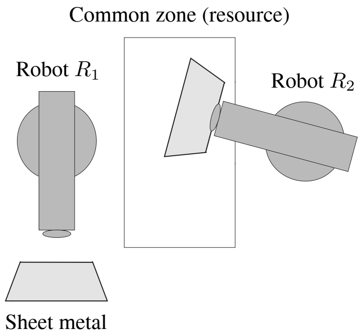
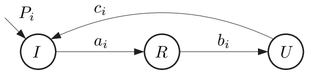
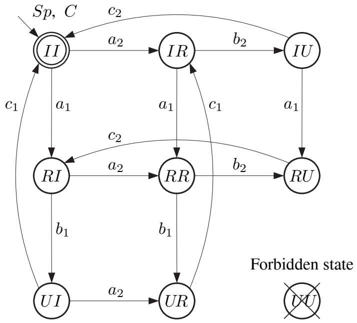


Figure 1.13 Robots moving sheet metals. Each robot needs to book the common zone before the zone is passed. This booking of the common resource guarantees that the robots including the sheet metals avoid collisions.

Automata for Robot System

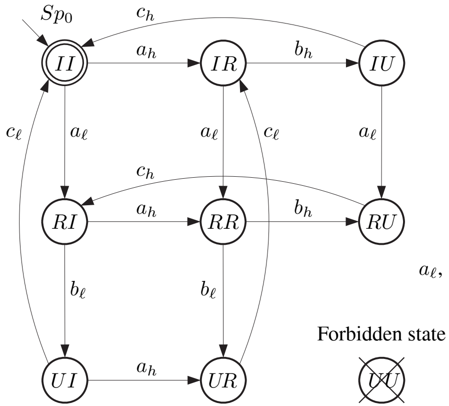


(a)

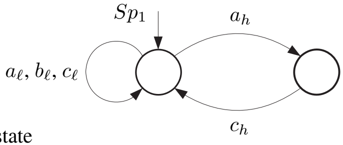


(b)

Mars Pathfinder



(a)



(b)

Mars Pathfinder: Deadlock

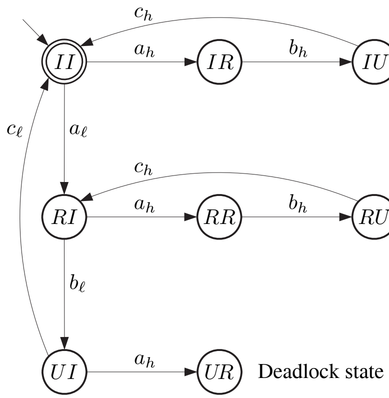


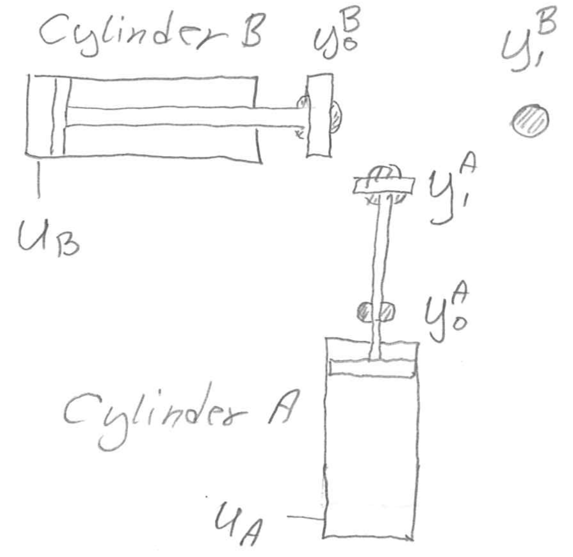
Figure 1.16 Total specification $Sp = Sp_0||Sp_1$. A correct controller C is obtained by removing the deadlock state from Sp

Summary: Design of Feedback Controllers

Summary: Design of feedback controllers

- The closed loop system $\Psi(P, C) = P||C$, where P = plant automaton, C = controller automaton and || is the synchronous composition. C normally restricts the behavior of P
- The closed loop specification Sp expresses desired properties of $\Psi(P, C)$. Sp is based on combinations of marked and forbidden states, and desired and forbidden event sequences.
- Verification: Given P, C and Sp, evaluate if $\Psi(P, C)$ satisfies Sp.
- Controller synthesis: Given P and Sp, compute C such that $\Psi(P, C)$ satisfies Sp.

Controller implementation



Controller automaton

$$\begin{tikzcd}
C \arrow[r] & \overset{u_A=0}{0} \arrow[r, "y_0^A\uparrow"] & \overset{u_A=1}{1} \arrow[d, "y_1^A\uparrow"] \\
& \underset{u_B=0}{3} \arrow[u, "y_0^B\uparrow"] & \underset{u_B=1}{2} \arrow[l, "y_1^B\uparrow"]
\end{tikzcd}$$

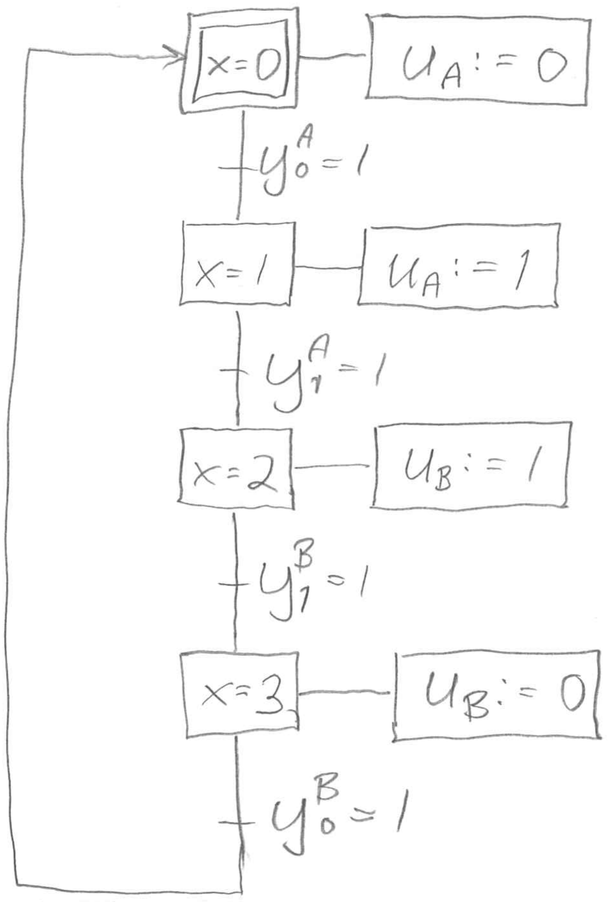
Init $x=0 \wedge y_0^B=1$

Implementation

```
if  $x=0 \wedge y_0^A=1$  then  $x'=1$
if  $x=1 \wedge y_1^A=1$  then  $x'=2$
if  $x=2 \wedge y_1^B=1$  then  $x'=3$
if  $x=3 \wedge y_0^B=0$  then  $x'=0$
else  $x'=x$
if  $x \geq 1$  then  $u_A=1$  else  $u_A=0$
if  $x=2$  then  $u_B=1$  else  $u_B=0$
```

Implementation in sequential Function Chart (SFC)

SFC is one of the standard languages for programmable logic controllers (PLCs)



$x=0$
$u_A := 0$
$y_0^A = 1$
$x=1$
$u_A := 1$
$y_1^A = 1$
$x=2$
$u_B := 1$
$y_1^B = 1$
$x=3$
$u_B := 0$
$y_0^B = 1$