



SAPIENZA
UNIVERSITÀ DI ROMA



Sistemi dinamici a eventi discreti – DEDS

Automazione I

2/12/2014

Alessandro De Luca

Sistemi e modelli matematici

Sistema: un insieme di componenti cooperanti e interagenti che realizzano una funzionalità complessiva, impossibile da realizzare da ciascuna parte separatamente (IEEE)

- ➔ definizione applicabile a sistemi fisici e non (comportamento umano, economia)

Si utilizzano modelli matematici per

- ❑ formalizzare il comportamento del sistema (descrizione e valutazione)
- ❑ riprodurre tale comportamento (simulazione al calcolatore)
- ❑ studiare proprietà di interesse (anche per il progetto del sistema di controllo)

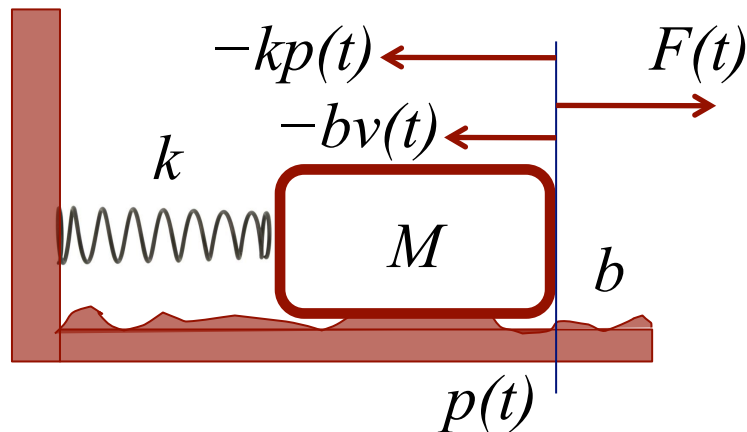
Si possono distinguere due tipi di modelli

- ❑ guidati dal tempo
 - ➔ evolvono come soluzioni nel tempo di equazioni differenziali o alle differenze
- ❑ guidati da eventi
 - ➔ evolvono in corrispondenza all'accadere di eventi (in modo asincrono)

Modelli guidati dal tempo

evoluzioni temporali ($t \in \mathbb{R}$, $k \in \mathbb{Z}$) con relazioni dinamiche/algebriche tra variabili di ingresso, stato, uscita (“classico” Teoria dei Sistemi, Controlli Automatici)

esempio: massa-molla-attrito

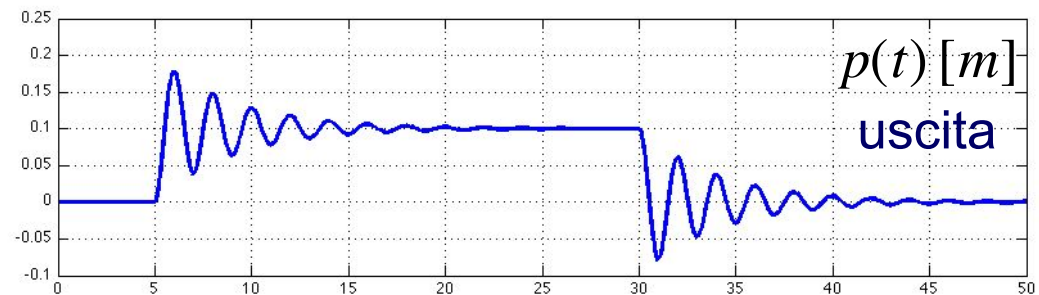
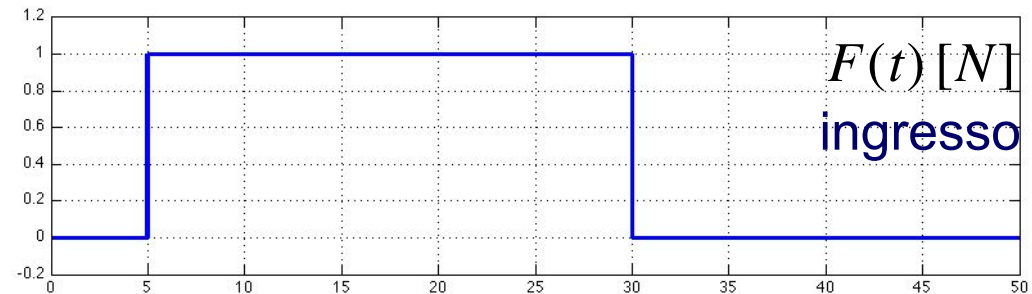


$$F(t) - kp(t) - bv(t) = Ma(t)$$

$$u(t) = F(t)$$

$$x(t) = \begin{bmatrix} p(t) & v(t) \end{bmatrix}^T$$

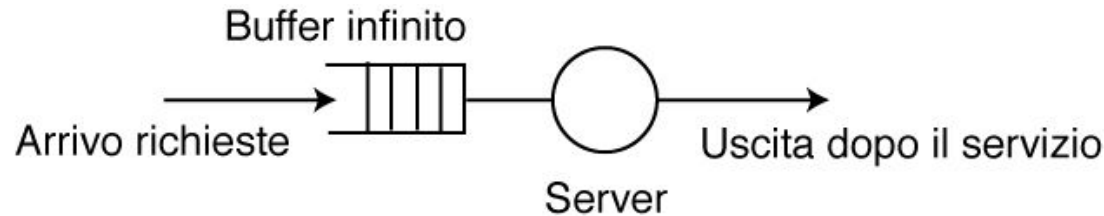
$$y(t) = p(t)$$



$$\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ -k/M & -b/M \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$
$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(t)$$

Modelli guidati da eventi

esempio: sistema client/server con buffer a capacità infinita



Molti sistemi fisici non vengono ben descritti da variabili temporali

- ❑ sistemi manifatturieri
- ❑ sistemi di comunicazione
- ❑ sistemi di controllo del traffico
- ❑ ...

In questi casi infatti

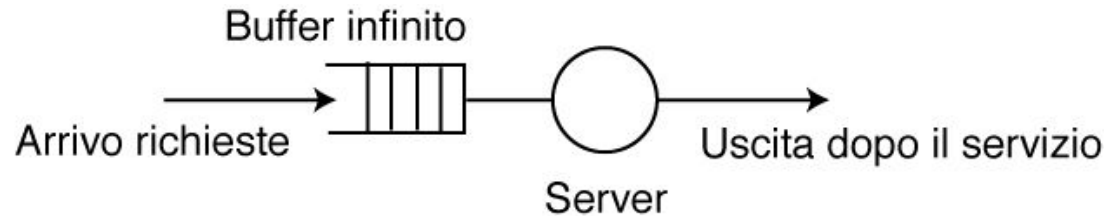
- ❑ le variabili possono assumere valori simbolici (es: on/off) in un set finito
- ❑ il sistema cambia stato (e le variabili il loro valore) in modo asincrono
- ❑ ... in corrispondenza di opportuni eventi

Si parla allora di Sistemi (Dinamici) a Eventi Discreti

- ❑ **DEDS** = Discrete Event Dynamic Systems

Modelli guidati da eventi

esempio: sistema client/server con buffer a capacità infinita



❑ qual è lo stato del sistema?

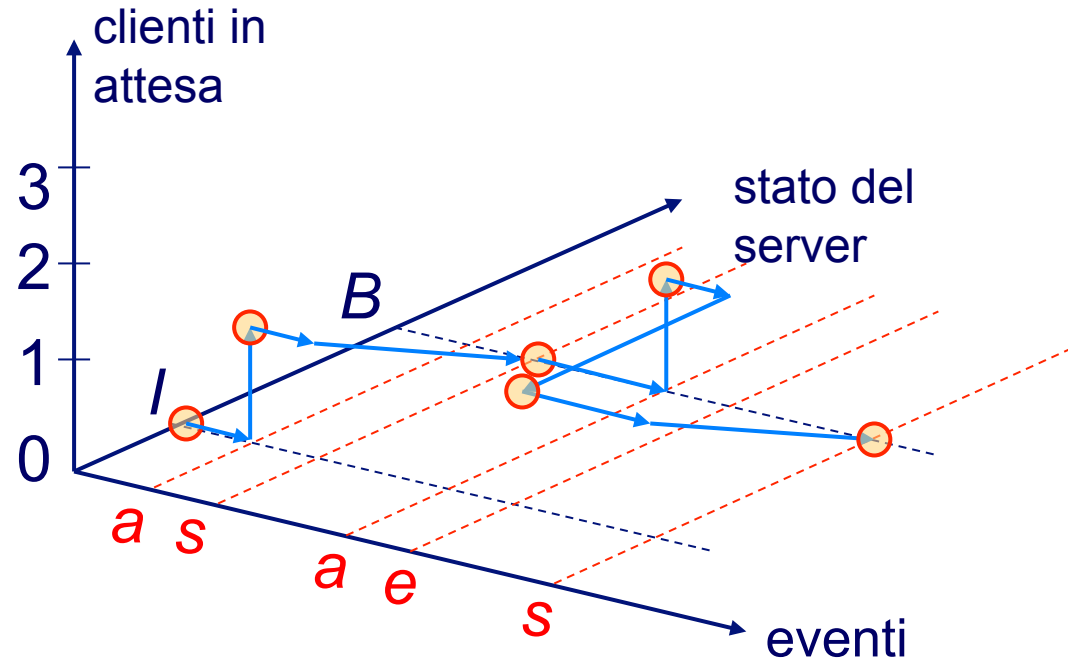
- ❑ numero di clienti in attesa nel buffer (numero naturale)
- ❑ stato del server (*Idle* = *I*, *Busy* = *B*)
- ❑ $X = \{I, B\} \times N$

❑ quando cambia lo stato?

- ❑ se arriva un nuovo cliente (evento *a*)
- ❑ se inizia un servizio (evento *s*)
- ❑ se finisce un servizio (evento *e*)

“evento” = un’azione *istantanea* sul sistema che ne provoca l’immediato cambiamento di stato

come evolve lo stato del sistema?

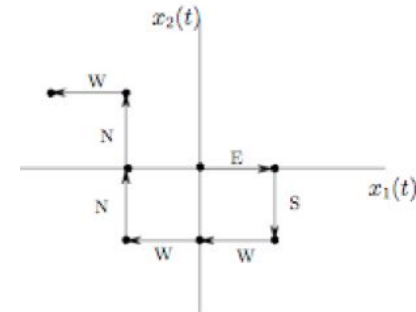


un sistema ad eventi discreti è caratterizzato dalle seguenti proprietà

- ❑ lo spazio degli stati è un insieme discreto (di cardinalità finita o infinita)
- ❑ l'evoluzione dello stato è guidato da eventi

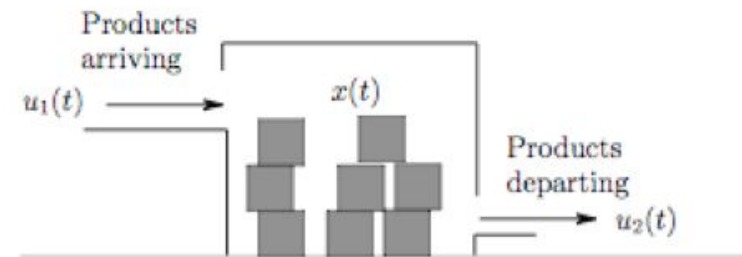
“Random walk” con passi unitari lungo assi coordinati nel piano

- insieme degli stati $X = (x_1, x_2) \in \mathbb{Z} \times \mathbb{Z}$
- insieme degli eventi $E = \{N, S, W, E\}$
- in questo caso, eventi concorrenti “commutano”



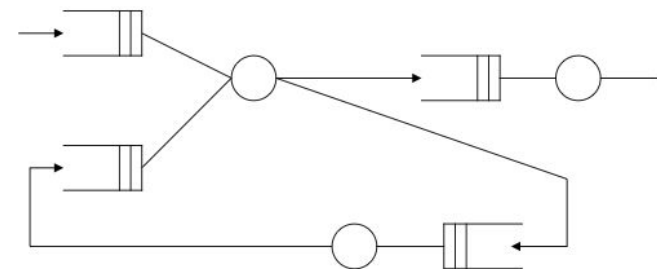
Magazzino di prodotti

- stato $X = \#$ prodotti nel magazzino
- insieme degli eventi $E = \{A, P\}$



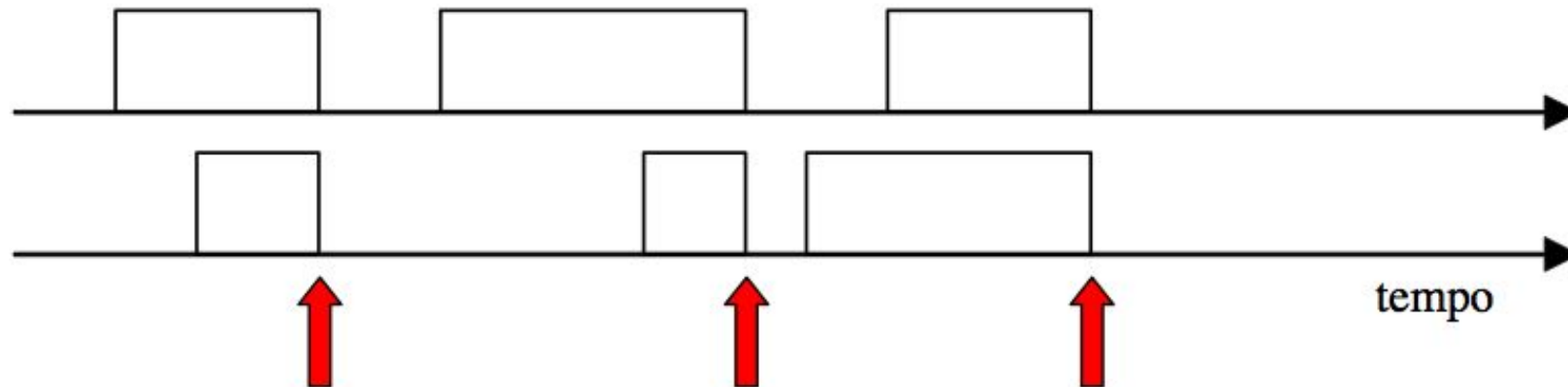
Code e reti di code (di servizio)

- stato $X = \#$ clienti in ciascuna coda
- eventi $E = \{A, S, P\}$
- capacità max di ciascuna coda
- disciplina di servizio (FIFO, ...)



Time- o Event-driven?

Il tempo è presente, ma non fondamentale nell'evoluzione del sistema ...

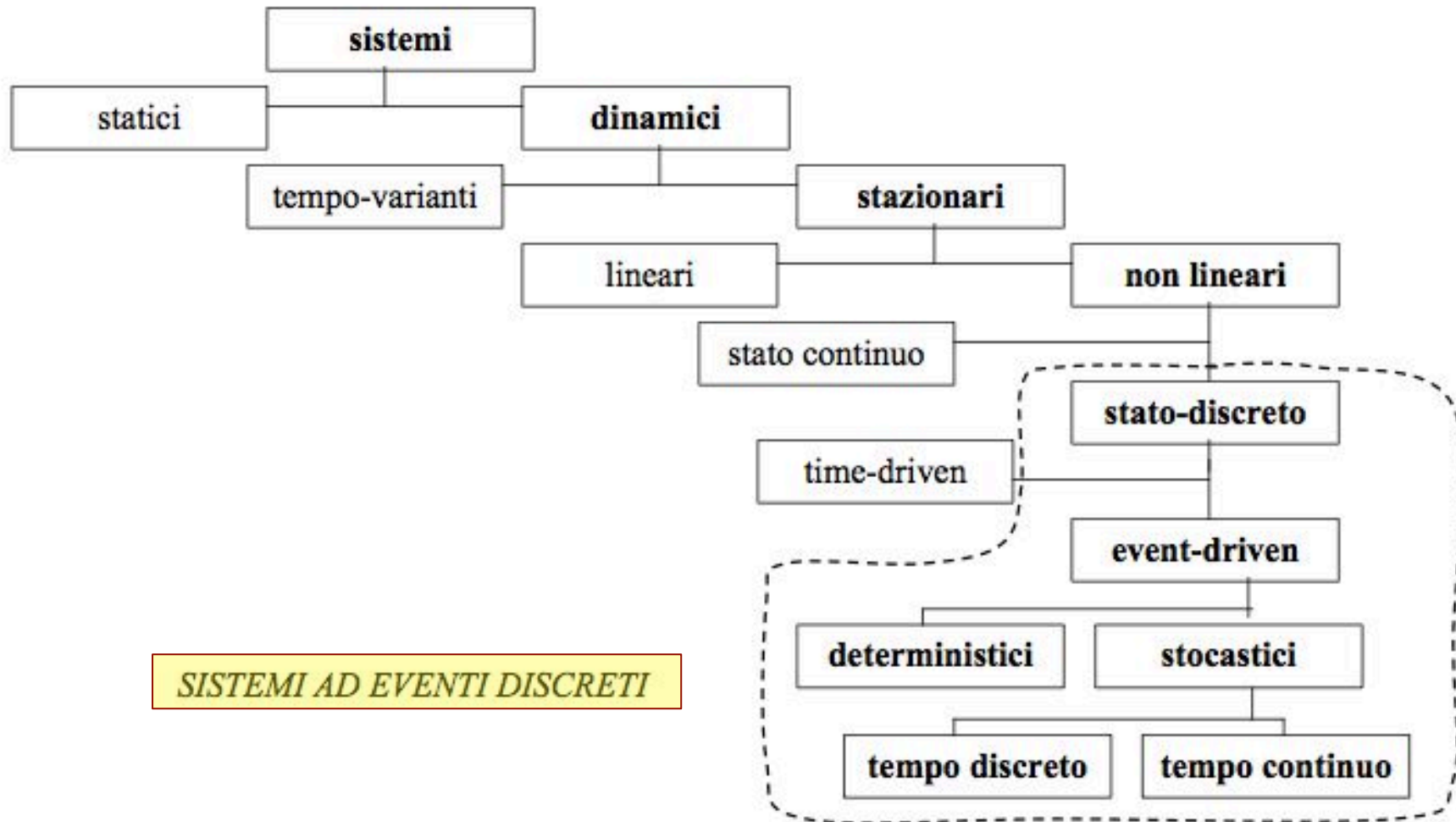


pulsante di comando manuale (oppure risposta del mouse)

- ❑ 0 = non premuto, 1 = premuto
- ❑ il sistema reagisce al rilascio del pulsante (premuta più o meno a lungo)
- ❑ a partire dallo stesso stato iniziale, i due ingressi *diversi nel tempo* producono lo *stesso* risultato (evoluzione dello stato)

Spesso si è comunque interessati a informazioni temporali, per poter valutare alcune caratteristiche o prestazioni: tempi medi di attesa in coda, tempi medi tra guasti, tempi di lavorazione, pezzi prodotti/unità di tempo (= throughput) negli FMS, ...

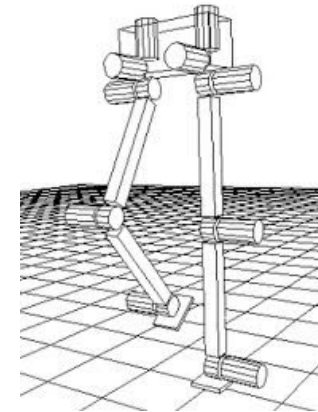
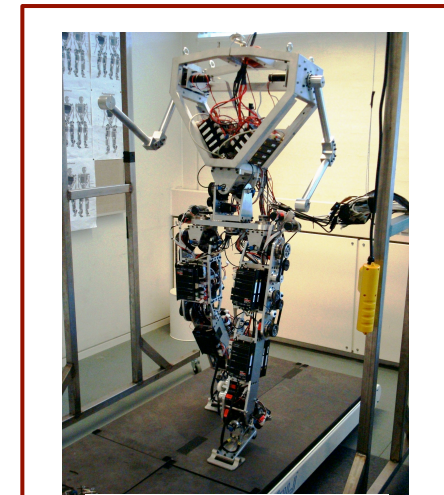
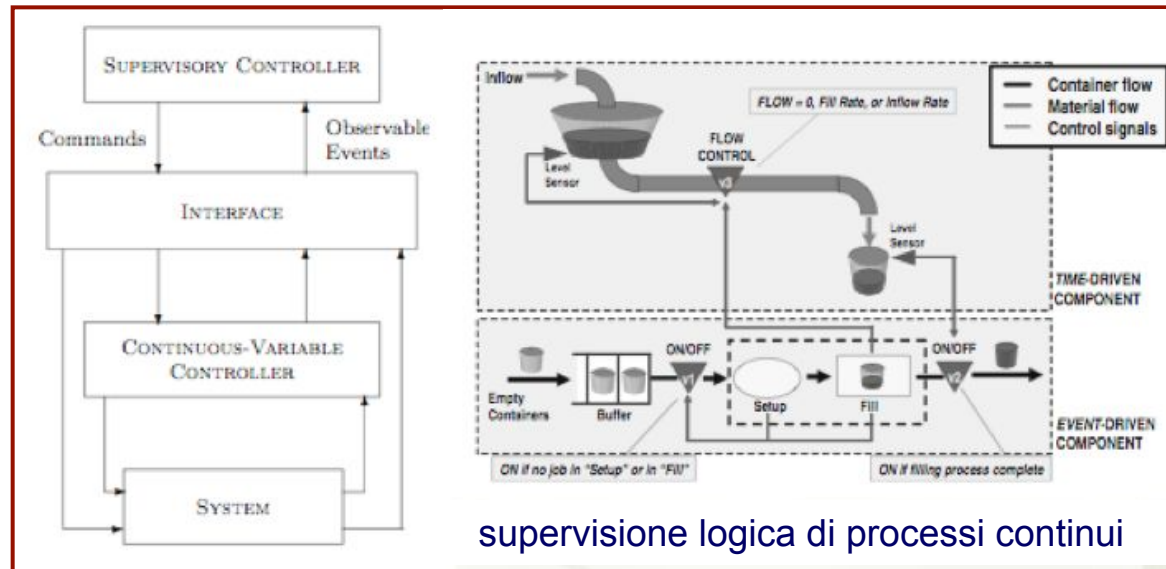
Classificazione dei sistemi



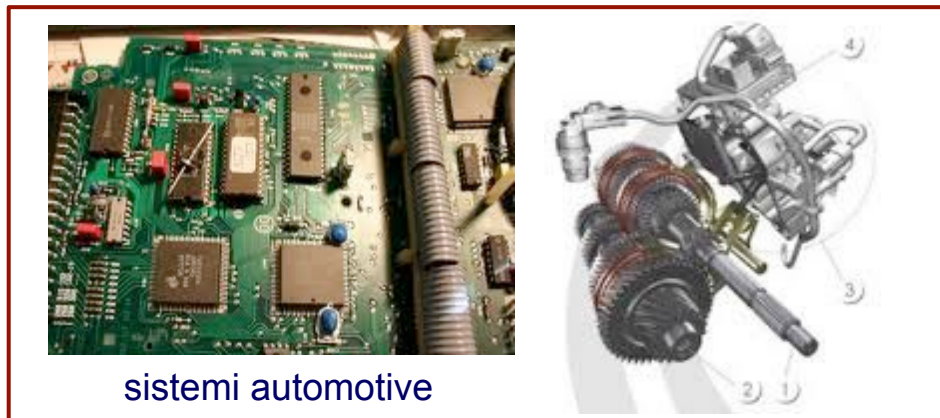
SISTEMI AD EVENTI DISCRETI

SISTEMI IBRIDI = combinazioni di sistemi time-driven e event-driven

Esempi di sistemi ibridi



sequenza di supporti nella locomozione



sistemi automotive

SISTEMI IBRIDI = combinazioni di sistemi time-driven e event-driven

Modelli per l'automazione

Cosa si vuole descrivere con i modelli nel contesto del “controllo logico”

- ❑ il funzionamento di *impianti complessi ed eterogenei*: celle robotizzate, centri di lavorazione meccanica, impianti batch/chimici, ...
- ❑ un impianto/processo può essere visto come un *insieme di sotto-sistemi*, dispositivi e macchinari opportunamente interconnessi, per ognuno dei quali può essere sviluppato un modello in dettaglio (motore, laminatoio, robot, ...)
- ❑ interessa studiare questi processi a un *livello di astrazione più elevato*, in cui si evidenziano le **sequenze di operazioni**, con i relativi problemi di **sincronizzazione, parallelismo, e conflitti nella allocazione delle risorse**

Alcune domande a cui si cerca una risposta

- ❑ che operazione deve essere svolta dopo l'operazione X?
- ❑ le operazioni X e Y possono essere fatte in parallelo in modo sincrono/asincrono?
- ❑ in quali condizioni non si deve eseguire l'operazione X?
- ❑ ci sono risorse sufficienti per svolgere le operazioni che occorrono?
- ❑ cosa bisogna fare (o non fare) per evitare di finire in un blocco critico (deadlock)?
- ❑ come si devono gestire delle risorse condivise?

Modelli per l'automazione

Lo studio delle possibili evoluzioni di stato di un DEDS non consente di comprendere e rappresentare esplicitamente i meccanismi interni di funzionamento del sistema

⇒ *uso di diversi tipi di modelli formali*

□ modelli **operazionali** (con strutture di transizione)

- ➔ automi
- ➔ reti di Petri (di varie tipologie)
- ➔ Grafcat/Sequential Functional Chart (SFC)



vediamo essenzialmente questi tipi di modelli

□ modelli **dichiarativi** (con dichiarazione dei comportamenti del sistema)

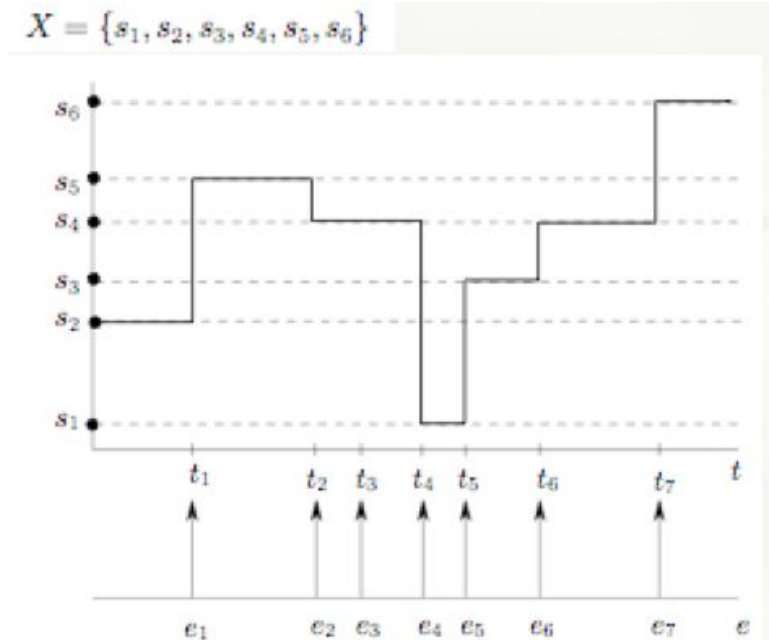
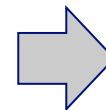
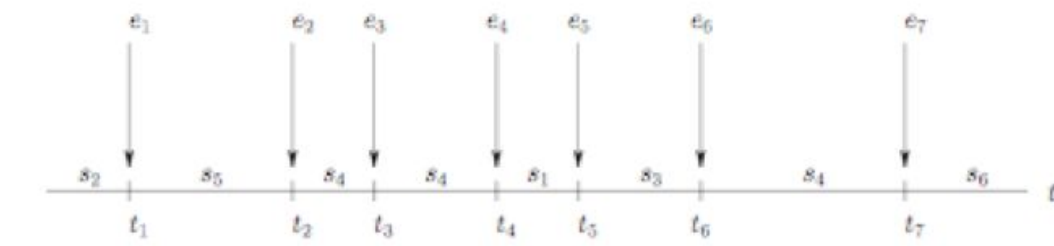
- ➔ basati su regole
- ➔ basati su equazioni

In linea di principio, per descrivere (e studiare) un DEDS si potrebbe usare anche un *linguaggio di programmazione* con i relativi pregi e difetti:

- | | |
|--|--|
| + è un modello formale | – modellistica troppo dettagliata |
| + è eseguibile (su una data macchina) | – difficile astrazione dei concetti principali |
| + disponibili supporti di sviluppo (debug) | – non ha strutture standard |
| + non va tradotto: pronto per il controllo | – in generale, difficile portabilità del “modello” |

Aspetti salienti nei DEDS

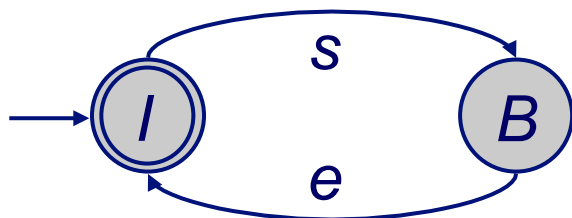
- sequenza di eventi $\{e_1, e_2, e_3, e_4, \dots\}$
- sequenza di eventi temporizzati $\{(t_1, e_1), (t_2, e_2), (t_3, e_3), (t_4, e_4), \dots\}$
- dato lo stato iniziale, una sequenza di eventi definisce in modo *univoco* l'evoluzione dello stato
- uso di “teoria dei linguaggi”: dato un *alfabeto* E di eventi (simboli), tutte le possibili sequenze di eventi (*stringhe*) costituiscono un *linguaggio* (evtl. temporale)
- DEDS a vari livelli di astrazione/interesse
 - ➔ **logico** (solo ordine degli eventi)
 - ➔ **temporale** (anche gli istanti di occorrenza, tipicamente deterministici)
 - ➔ **stocastico** (distribuzioni probabilistiche)



Un automa è una macchina a stati $G = (X, E, f, \Gamma, x_0, X_m)$ caratterizzata da

- insieme degli stati X (i singoli stati dell'automata sono **globali** per il sistema)
- insieme degli eventi E
- funzione di transizione $f: X \times E \rightarrow X$ che ad ogni coppia (stato, evento) associa lo stato successivo
- funzione di evento attivo $\Gamma: X \rightarrow 2^E$, per cui $\Gamma(x_0)$ definisce tutti gli eventi e per i quali è definita $f(x, e)$
- stato iniziale $x_0 \in X$ (rappresentato con una freccia accanto al nodo)
- *eventuali* stati "marcati" $X_m \in X$ di interesse (rappresentati da doppi cerchi)

es: server



un automa è anche un generatore di linguaggio:

“alfabeto” di eventi $E = \{s, e\}$

“linguaggio” L generato dall'automata G

$L(G) = \{\epsilon, s, se, ses, sese, seses, \dots\}$

↑
stringa
vuota

tutte le stringhe
generabili

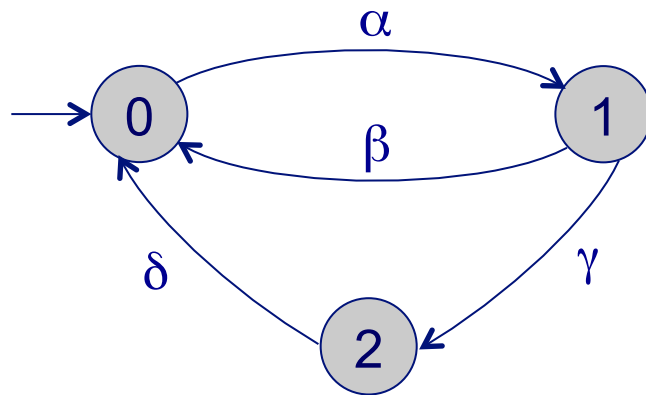
Un automa a stati finiti **con ingressi e uscite** è la sestupla $H = (U, X, Y, f, h, x_0)$ dove

- $U = \{u_1, u_2, u_3, \dots\}$ è l'insieme degli eventi in ingresso
- $X = \{x_0, x_1, x_2, x_3, \dots\}$ è l'insieme (finito) degli stati \Rightarrow **finite state machine**
- $Y = \{y_1, y_2, y_3, \dots\}$ è l'insieme degli eventi in uscita
- $f: X \times U \rightarrow X$ è la funzione di transizione dello stato (eventualmente non definita per alcune coppie $(x_i, u_k) \Rightarrow$ eventi in ingresso non ammissibili in certi stati)
- $h: X \times U \rightarrow Y$ oppure $h: X \rightarrow Y$ è la funzione di uscita
- x_0 è lo stato iniziale

Automi a stati finiti con funzione di transizione deterministica sono i più comuni

- **automi di Mealy**: la funzione di uscita dipende sia dallo stato che dall'ingresso (sistema dinamico proprio, con legame diretto in-out): $h: X \times U \rightarrow Y$
- **automi di Moore**: la funzione di uscita dipende solo dallo stato e non dall'ingresso (sistema dinamico strettamente proprio): $h: X \rightarrow Y$
- la trasformazione di un automa di Moore in uno di Mealy (e viceversa) è sempre possibile

macchina utensile soggetta a guasti

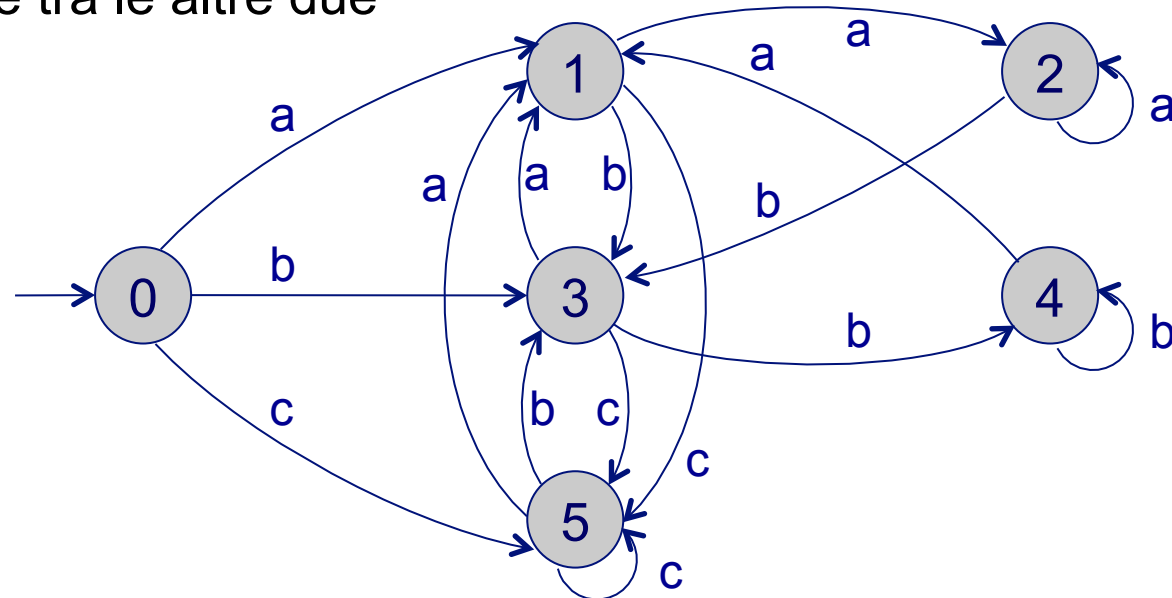


$$L(G) = \{\varepsilon, \alpha, \alpha\beta, \alpha\beta\alpha\beta, \dots, \alpha\gamma\delta, \alpha\gamma\delta\alpha\beta, \dots\}$$

- ❑ stato $X = \{0 = \text{macchina libera}, 1 = \text{macchina occupata}, 2 = \text{macchina guasta}\}$
- ❑ eventi $E = \{\alpha = \text{inizio lavorazione}, \beta = \text{fine lavorazione}, \gamma = \text{la macchina si guasta}, \delta = \text{la macchina è riparata}\}$
- ❑ hp di lavoro
 - ➔ all'inizio del periodo di osservazione, la macchina è libera
 - ➔ la macchina si può guastare solo quando esegue una lavorazione (= macchina occupata)
 - ➔ il pezzo in lavorazione al momento del guasto viene scartato (= fine lavorazione)

Modellistica con automi

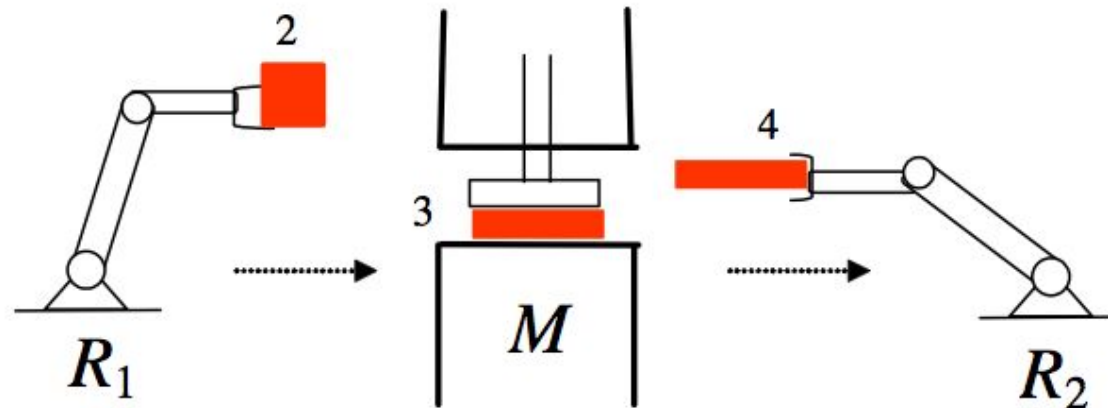
una macchina utensile esegue tre tipi di operazioni; per vincoli tecnologici, una delle tre operazioni non può essere eseguita subito dopo due consecutive esecuzioni della stessa operazione tra le altre due



- eventi $E = \{a, b, c\}$ \Leftarrow l'esecuzione delle tre diverse lavorazioni sulla macchina utensile
 - ➔ sia c l'operazione che presenta dei vincoli tecnologici
- stato $X = \{0 = \text{nessuna operazione eseguita};$
 - 1 = l'ultima operazione eseguita è di tipo a, la penultima non è di tipo a;
 - 2 = le ultime due operazioni eseguite sono di tipo a;
 - 3 = l'ultima operazione eseguita è di tipo b, la penultima non è di tipo b;
 - 4 = le ultime due operazioni eseguite sono di tipo b;
 - 5 = l'ultima operazione eseguita è di tipo c }

Modellistica con automi

due robot che caricano e scaricano pezzi in/da una macchina (ad es., una pressa)

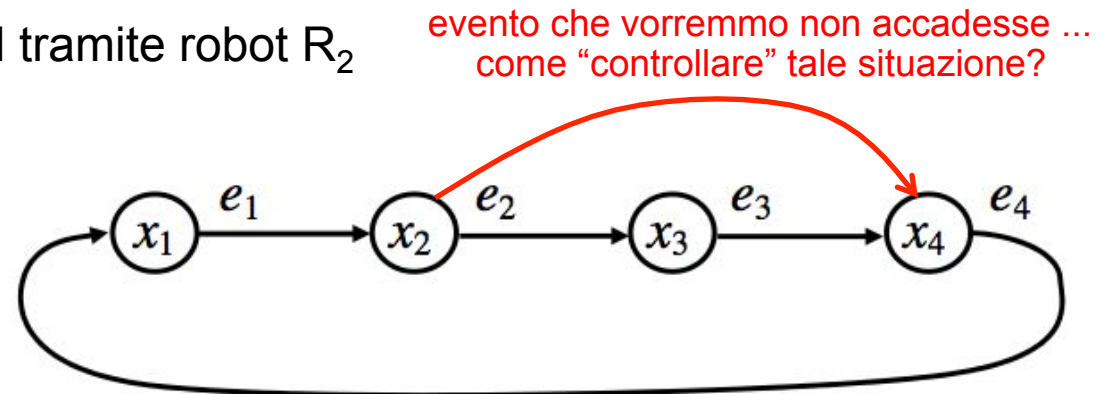


□ stati

- ➔ x_1 macchina M in attesa, disponibile per la lavorazione
- ➔ x_2 carico del pezzo su M da parte di robot R_1
- ➔ x_3 lavorazione
- ➔ x_4 scarico del pezzo lavorato da M tramite robot R_2

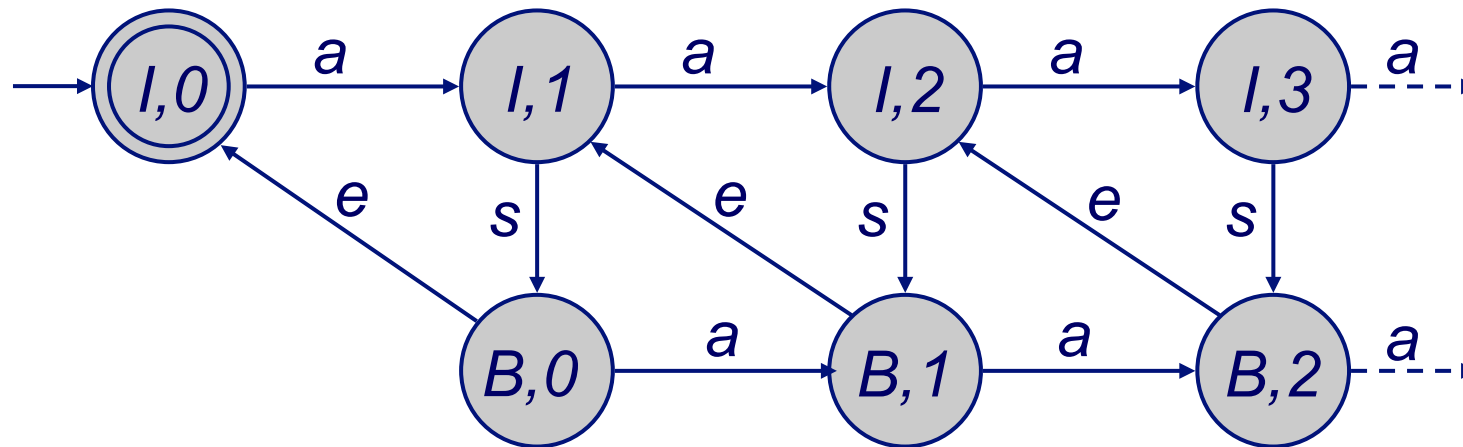
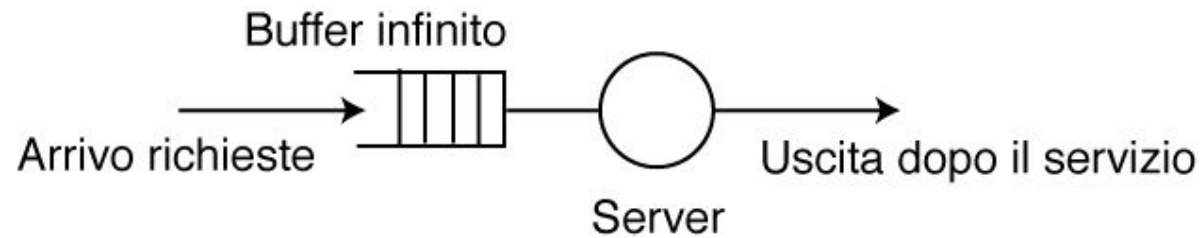
□ eventi (fasi del ciclo produttivo)

- ➔ e_1 inizio carico
- ➔ e_2 inizio lavorazione
- ➔ e_3 inizio scarico
- ➔ e_4 inizio attesa



Modellistica con automi

modello del sistema client/server (con buffer illimitato)



un numero
infinito di stati

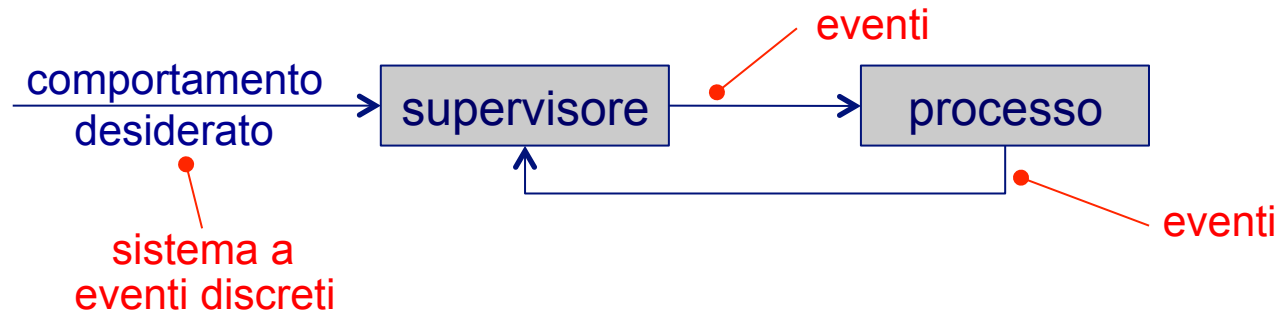
problema: gli automi **non** sono in grado di rappresentare con un numero finito di stati alcuni "linguaggi" a cardinalità infinita

Controllo di DEDS

Per un sistema dinamico a eventi discreti, si può progettare un controllore (anche esso guidato da eventi) in modo da ottenere un comportamento desiderato; due alternative:

- si implementa **direttamente** il comportamento logico desiderato del sistema automatizzato
- si cerca di modificare opportunamente il comportamento originale del processo (includendo tutti i conflitti, malfunzionamenti e guasti di interesse) in modo da ottenerne uno la cui correttezza è formalmente verificabile (**progetto di un supervisore DEDS**)

processo DEDS (ad anello aperto) + supervisore DEDS = DEDS (ad anello chiuso)



servono

- un modello del processo da controllare
- le specifiche sul comportamento desiderato, espresse tramite modello DEDS (“omogeneo” al modello del processo), vincoli da non violare, “restrizioni” sulle evoluzioni dinamiche, ...
- un supervisore guidato da eventi/*misure* in ingresso e con eventi/*comandi* in uscita
- strumenti di progetto che permettano di combinare/integrare con relativa facilità diversi sottosistemi DEDS e di verificarne il comportamento

Implementazione del supervisore

- il supervisore (= controllore) è un DEDS che evolve in modo **asincrono**, interagendo con il processo da controllare attraverso **eventi**
- lo strumento di calcolo dedicato al controllo (PIC, PLC o PC) è invece **sincrono** (ha un clock \Rightarrow tempo di ciclo T_c) e interagisce con il processo scambiando **segnali**

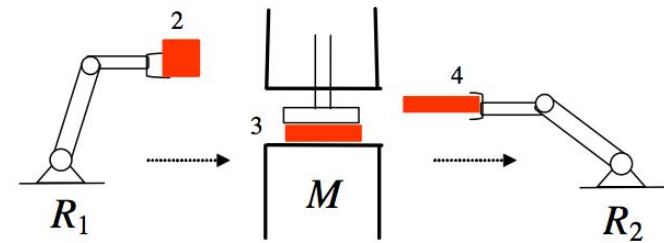


- **occorrono quindi**
 - ➔ *interfaccia eventi-segnali* (e viceversa) per la traduzione delle informazioni
 - ➔ opportuna *sincronizzazione* tra evoluzione del DEDS e quella del calcolatore di controllo
- **implementazione time-driven del controllore DEDS**
 - ➔ le problematiche sono quelle generali relative alla **gestione sincrona di eventi asincroni**
 - ➔ la permanenza minima dei segnali logici negli stati deve essere maggiore di T_c
 - se un segnale logico cambia stato due volte nello stesso periodo di campionamento, l'evento è perso
 - ➔ l'unità di calcolo deve completare sempre le attività in un ciclo $T_c \Rightarrow$ comportamento real-time
 - ➔ il ritardo di controllo complessivo (incluso quelli dei moduli I/O) deve essere tollerabile dal sistema
 - un evento che accade subito dopo l'istante di clock, è rilevato con un ritardo T_c ; se la corrispondente azione di comando è eseguita entro il successivo T_c , il ritardo totale sarà al più **$2T_c$**
 - ➔ più eventi accaduti nello stesso tempo T_c sono considerati come simultanei
 - l'ordine effettivo di tale eventi deve essere indifferente nel determinare la relativa azione di controllo

Modello con automa del supervisore

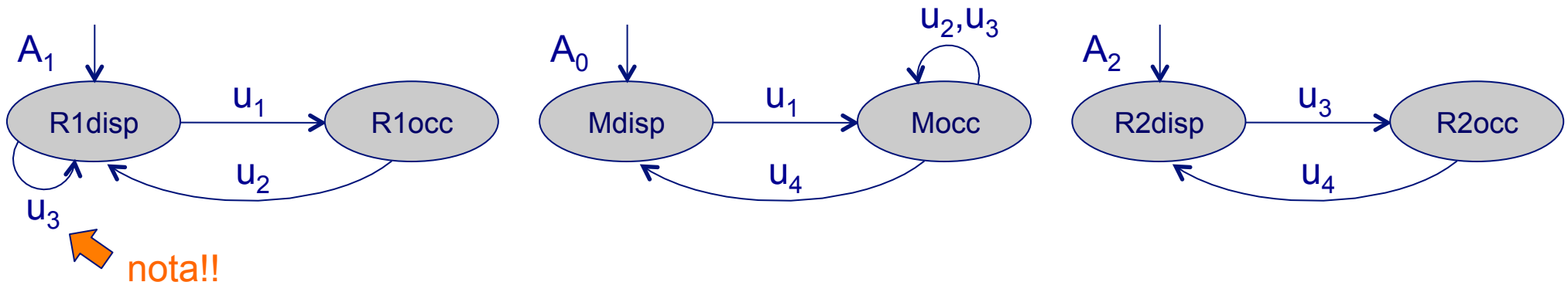
automa **con ingressi e uscite** (di Mealy) per descrivere l'asservimento con due robot di una macchina (**processo**) e il gestore logico del ciclo di lavorazione (**supervisore**)

- per ogni componente fisico (due robot + una macchina) introduciamo due stati
 - ➔ componente occupato (**occ**) in una lavorazione (macchina) o in un trasporto (robot)
 - ➔ componente disponibile (**disp**), in attesa di un nuovo compito
 - ➔ 6 stati per il **processo**
- $X = \{Mocc, Mdisp, R1occ, R1disp, R2occ, R2disp\}$



- gli eventi (in ingresso) sono
 - ➔ u_1 inizio ciclo (fine attesa)
 - ➔ u_2 fine carico
 - ➔ u_3 fine lavorazione
 - ➔ u_4 fine scarico

tre automi A_0 , A_1 e A_2 per descrivere il **processo** (non ci sono temporizzazioni!)



Modello con automa del supervisore

l'automata S che modella il gestore logico del ciclo di lavorazione (**supervisore**) ha 4 stati $X_S = \{x_1, x_2, x_3, x_4\}$ che corrispondono alle condizioni operative in tabella:

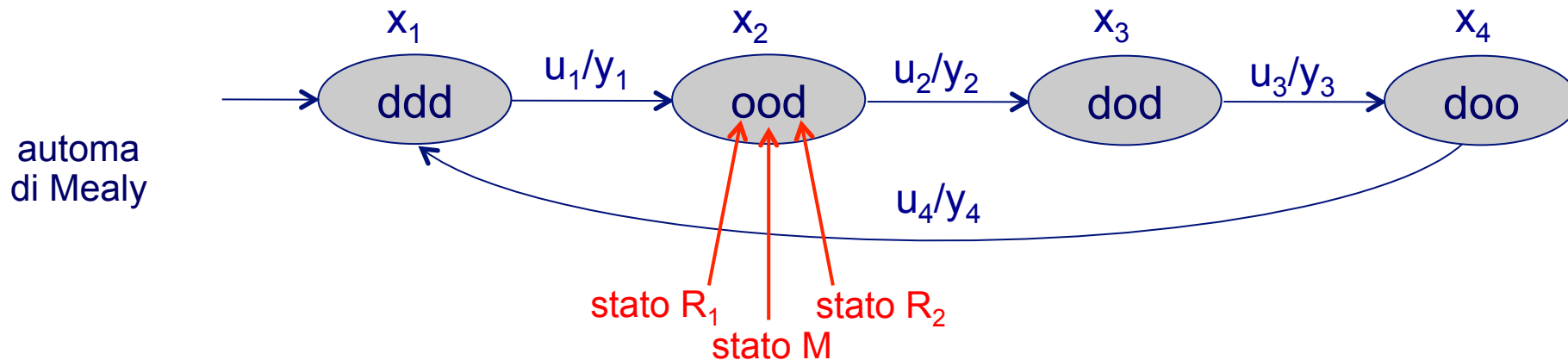
| stato | stato di R ₁ | stato di M | stato di R ₂ | codifica | descrizione |
|-------|-------------------------|------------|-------------------------|----------|----------------------------|
| x_1 | R1disp | Mdisp | R2disp | ddd | M in attesa di lavorazione |
| x_2 | R1occ | Mocc | R2disp | ood | R ₁ carica M |
| x_3 | R1disp | Mocc | R2disp | dod | M lavora |
| x_4 | R1disp | Mocc | R2occ | doo | R ₂ scarica M |

□ evoluzione dell'automata S

- ➔ nello stato iniziale x_1 l'unico evento che può avvenire è u_1 (in A_0 e A_1 , non in A_2 perché è assente nel suo alfabeto): M e R₁ passano in stato occupato, R₂ resta disponibile ($\Rightarrow x_2$)
- ➔ nello stato x_2 l'evento u_2 è l'unico accettabile in A_0 e A_1 (in A_2 è assente): R₁ cambia stato e torna disponibile ($\Rightarrow x_3$)
- ➔ nello stato x_3 l'unico evento accettabile da tutti e tre gli automi è u_3 : R₂ cambia stato e diventa occupato ($\Rightarrow x_4$)
- ➔ nello stato x_4 l'evento u_4 è l'unico accettabile in A_0 e A_2 (in A_1 è assente): anche M e R₂ tornano disponibili ($\Rightarrow x_1$)
- ➔ affinché la composizione sincrona funzioni, è essenziale la presenza dell'evento u_3 nell'alfabeto di A_1 (else u_3 potrebbe scattare prima di u_2) anche se tale evento riguarda un'altra parte del processo (A_2)!

Modello con automa del supervisore

rappresentazione grafica del supervisore S, con l'aggiunta degli eventi in uscita



- gli eventi in ingresso (al supervisore) sono “misure” dal processo
 - ➔ u_1 inizio ciclo (fine attesa)
 - ➔ u_2 fine carico
 - ➔ u_3 fine lavorazione
 - ➔ u_4 fine scarico
- gli eventi in uscita (dal supervisore) sono “comandi” per il processo
 - ➔ y_1 inizio carico
 - ➔ y_2 inizio lavorazione
 - ➔ y_3 inizio scarico
 - ➔ y_4 fine ciclo (inizio attesa)
- una certa simmetria...
↔
- possibile uso inefficiente delle risorse: nello stato x_2 la macchina M risulta occupata durante l'operazione di carico con il robot R_1 ma anche mentre R_1 trasporta il pezzo...
- moltiplicazione degli stati in presenza di operazioni concorrenti: il parallelismo di azioni tra dispositivi rende necessari più stati per rappresentare tutti i possibili sequenziamenti
- scarsa modularità: aggiunta di dettagli complica subito l'automa (e va ridisegnato da capo!)