



SAPIENZA
UNIVERSITÀ DI ROMA

Sistemi dinamici a eventi discreti – DEDS

Automazione

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

- ➔ 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)

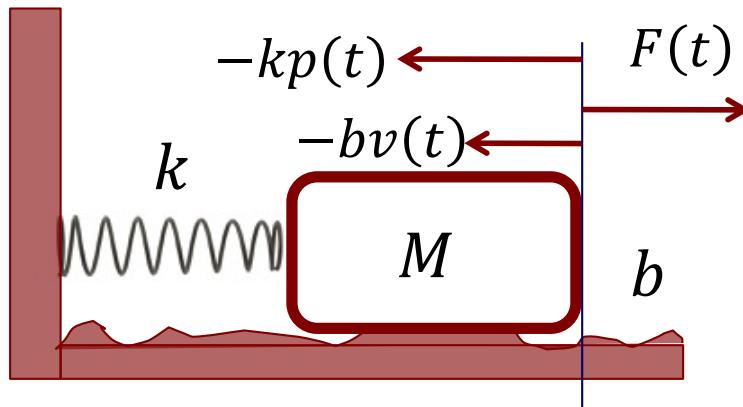
Si possono distinguere due tipi di modelli

- 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)

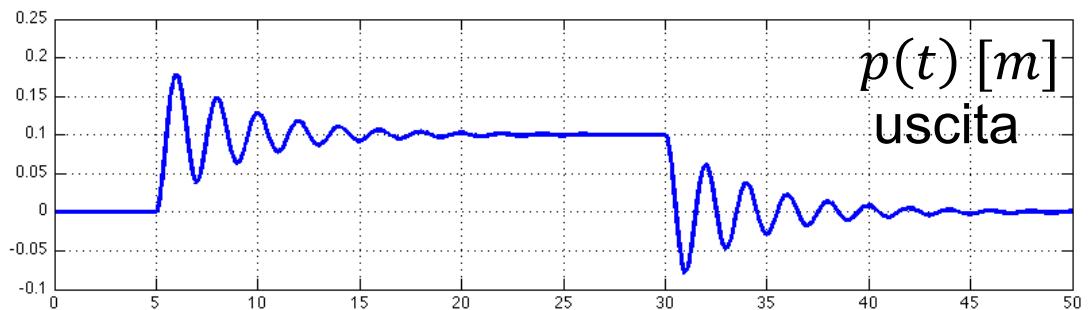
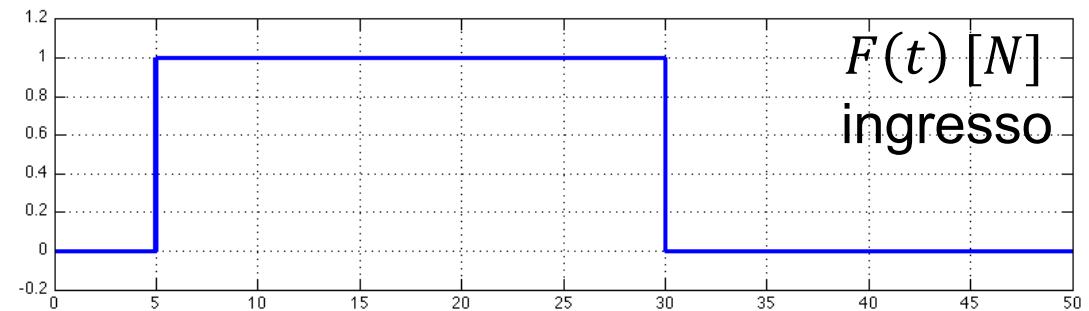
Modelli guidati dal tempo

evoluzioni temporali ($t \in \mathbb{R}$, per il tempo continuo; $k \in \mathbb{Z}$, per i campioni a tempo discreto) con relazioni dinamiche/algebriche tra variabili di ingresso, stato, uscita (approccio “classico” nei corsi di Sistemi Dinamici e Sistemi di Controllo)

esempio: massa-molla-attrito



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

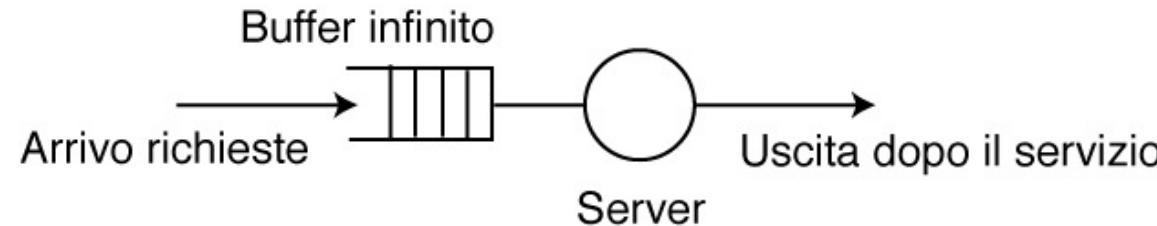


$$\begin{aligned} u(t) &= F(t) \\ x(t) &= [p(t) \quad v(t)]^T \\ y(t) &= p(t) \end{aligned}$$

$$\begin{aligned} \dot{x}(t) &= \begin{bmatrix} 0 & 1 \\ -k(M) & -\frac{b}{M} \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \\ y(t) &= [1 \quad 0] x(t) \end{aligned}$$

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

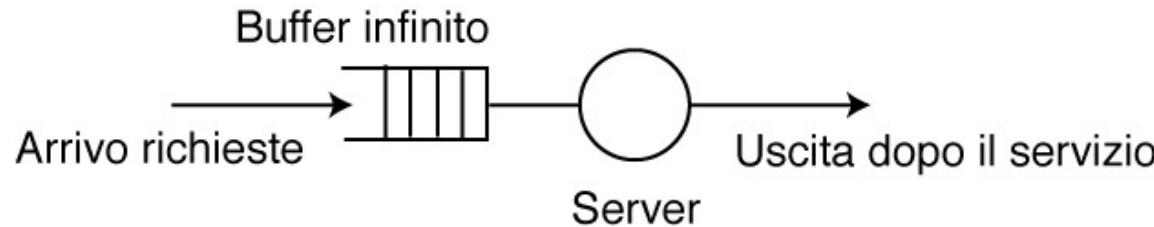
- le variabili possono assumere valori simbolici (es: on/off) in un set finito
- il sistema cambia stato (e le variabili il loro valore) in modo asincrono
- ... in corrispondenza di opportuni eventi

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?

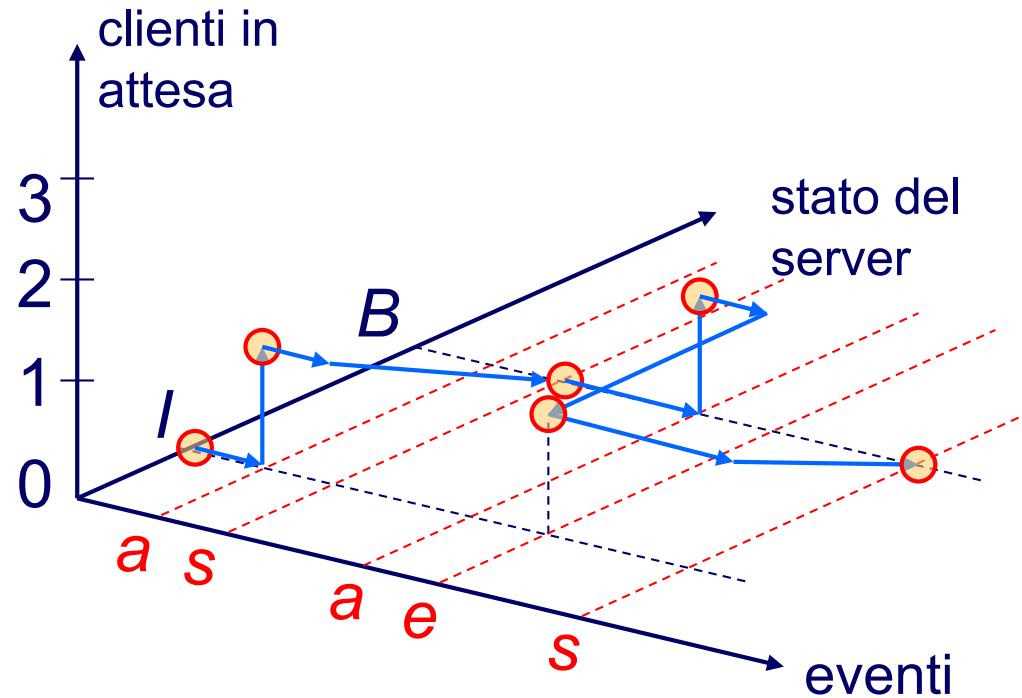
- numero di clienti in attesa nel buffer (numero naturale)
- stato del server (*Idle* = I , *Busy* = B)
- $X = \{I, B\} \times N$

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?

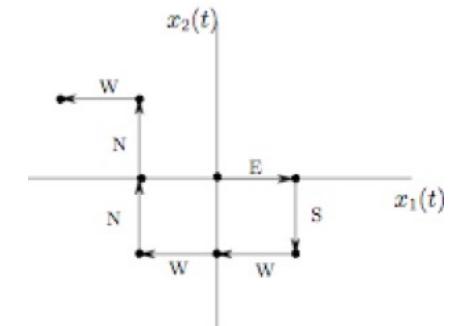


un sistema ad eventi discreti è caratterizzato dalle seguenti proprietà

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

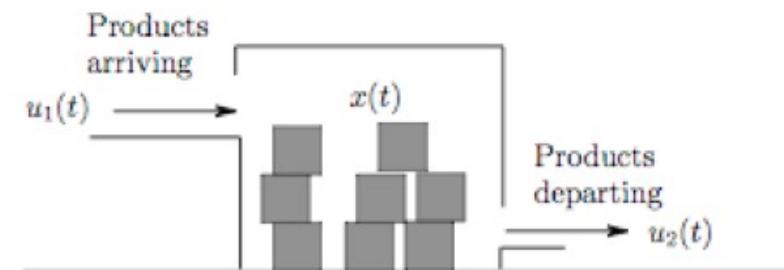
“Random walk” con passi unitari lungo assi coordinati nel piano

- 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”



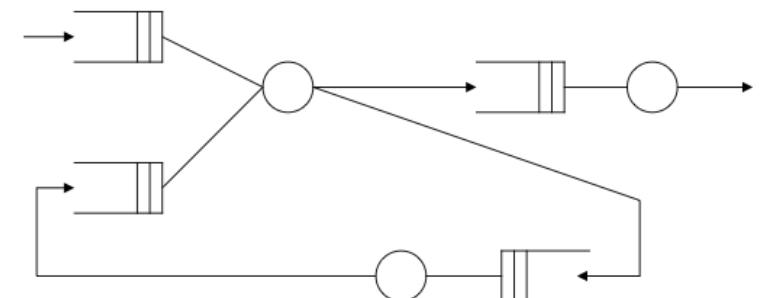
Magazzino di prodotti

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



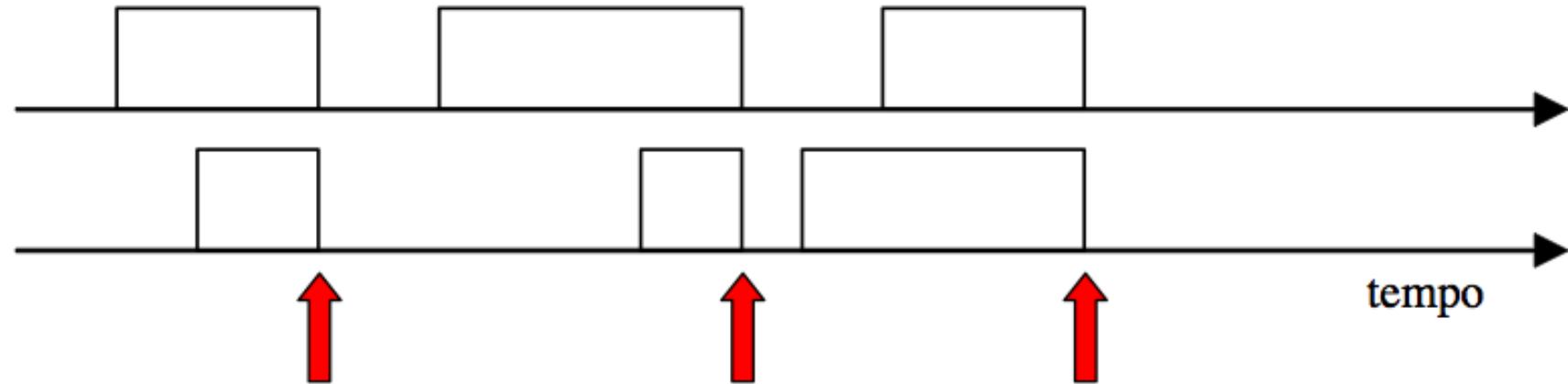
Reti di P code con R server

- $n_i = \#$ clienti nella coda $i \in \mathbb{N}$
- server j idle (I) o busy (B)
- stati $X = \{B, I\}^R \times \mathbb{N}^P$, 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 ...

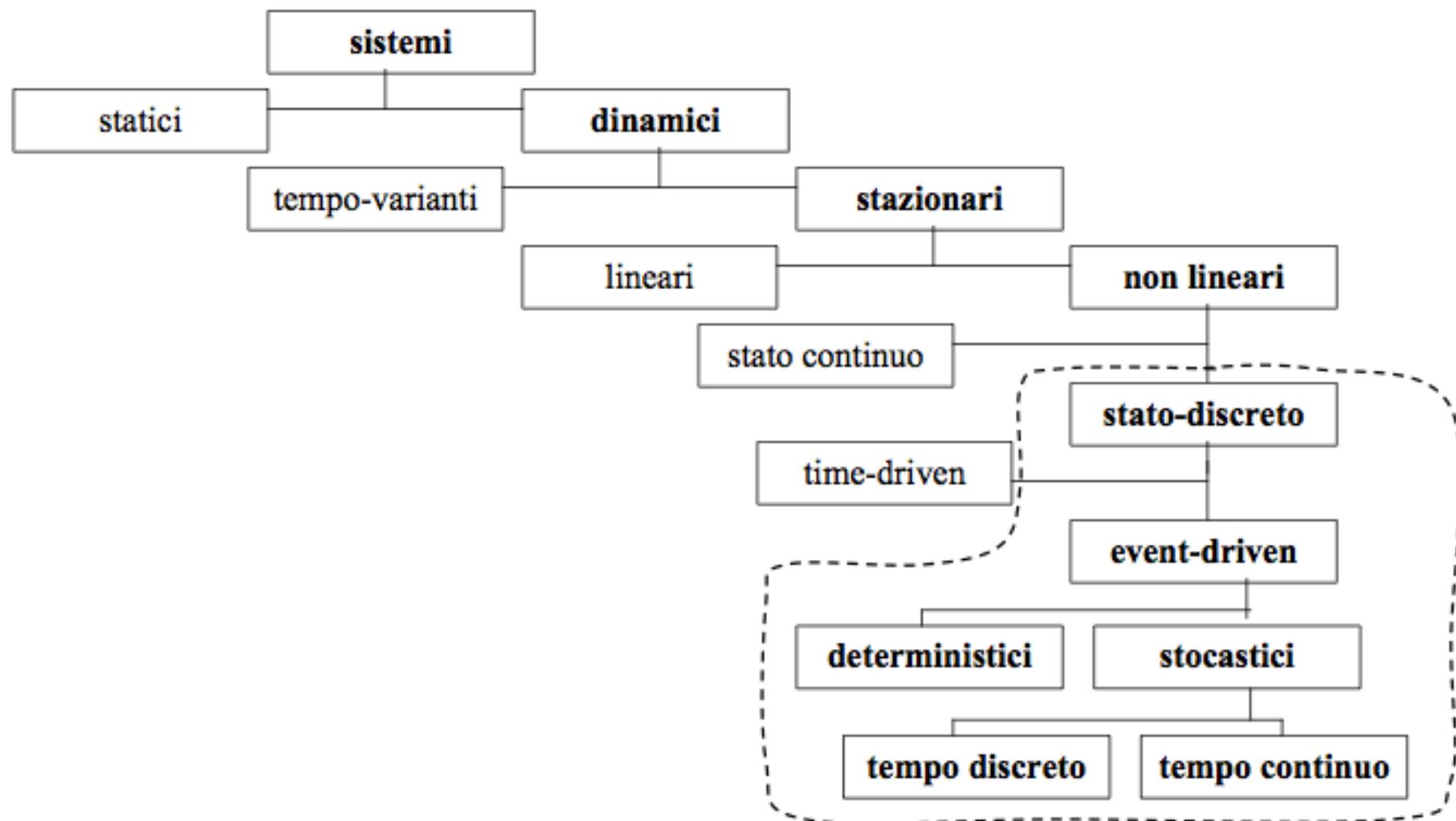


pulsante di comando manuale (oppure risposta del mouse)

- 0 = non premuto, 1 = premuto
- il sistema reagisce **al rilascio** del pulsante (premuto più o meno a lungo)
- nella figura, 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, ...

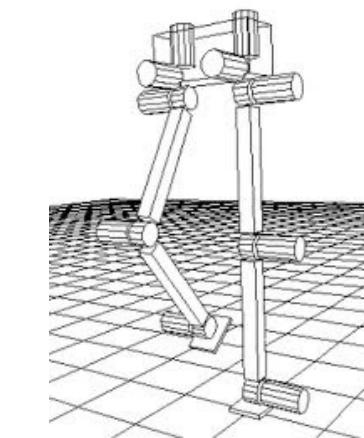
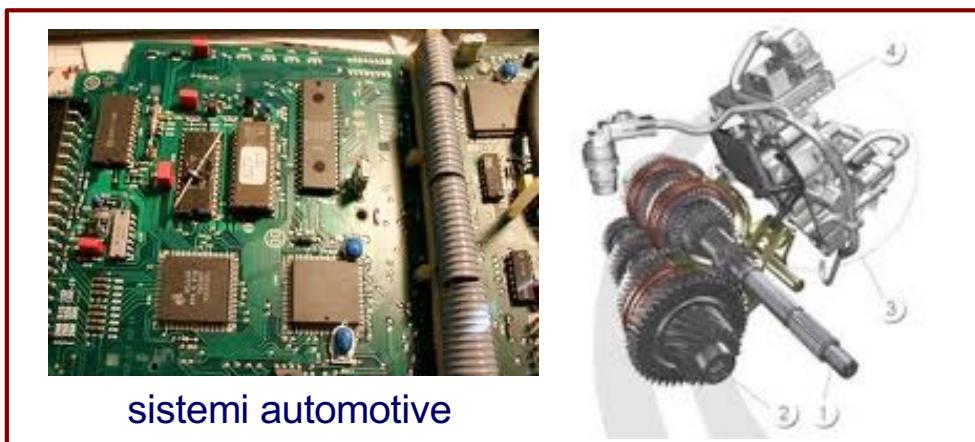
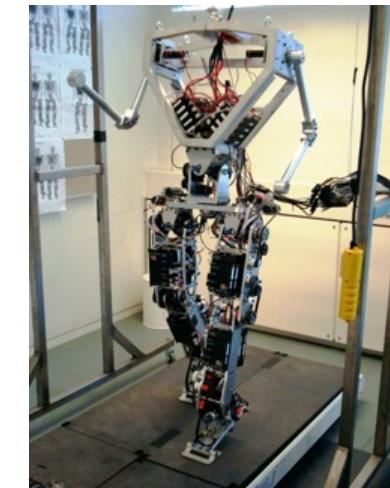
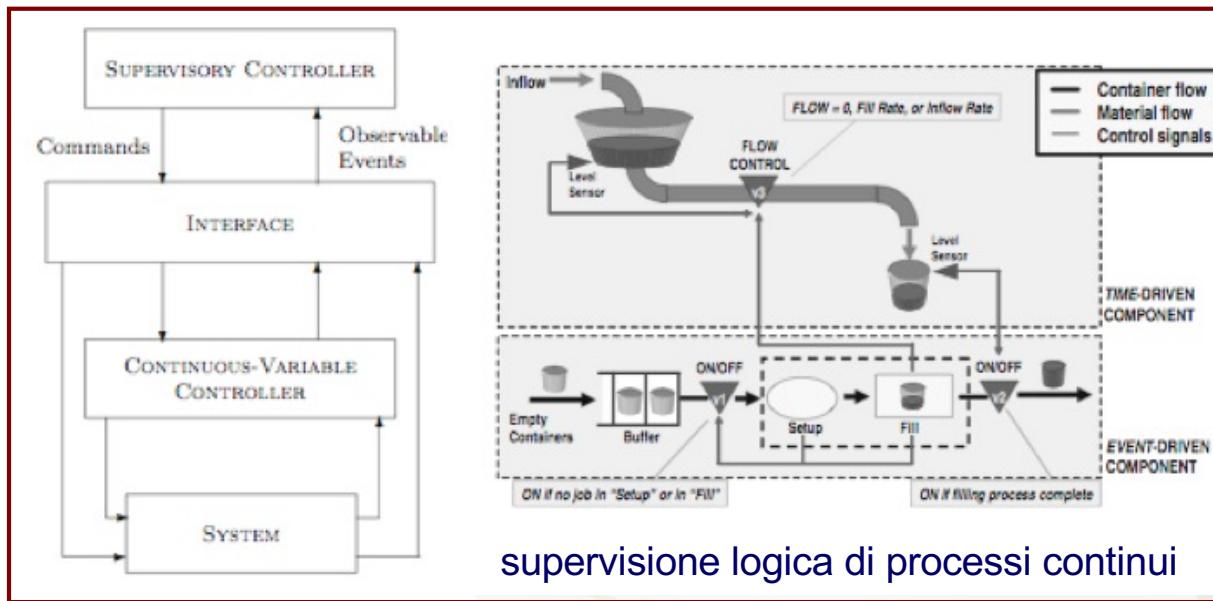
Classificazione dei sistemi



SISTEMI IBRIDI

= combinazioni di sistemi time-driven e event-driven

Esempi di sistemi ibridi

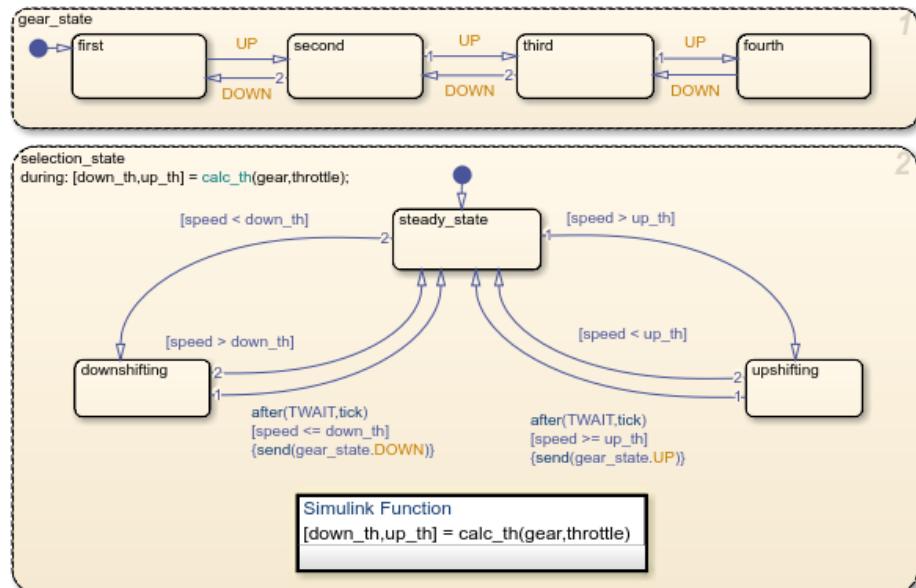


sequenza di supporti
nella locomozione

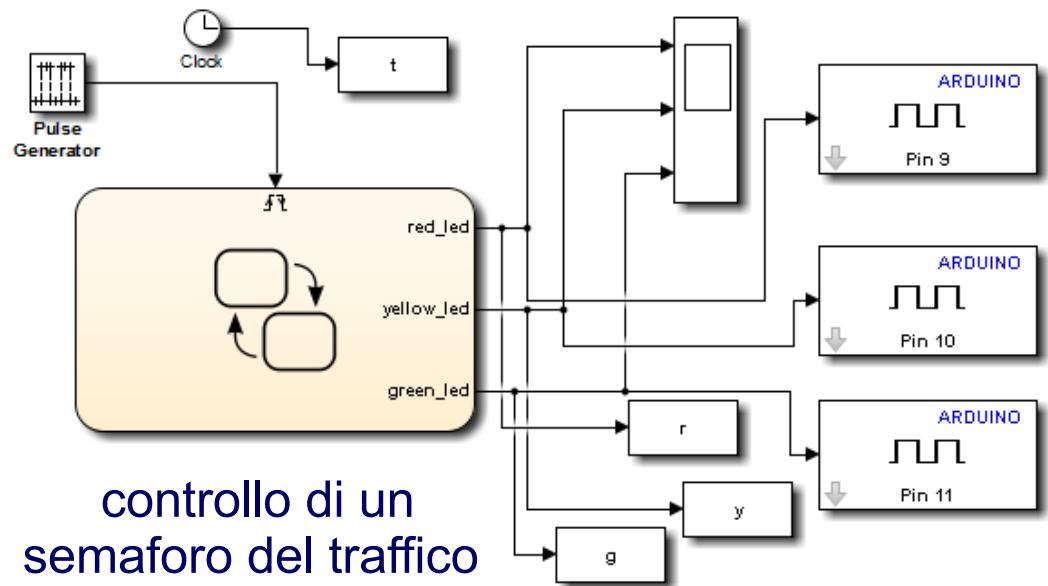
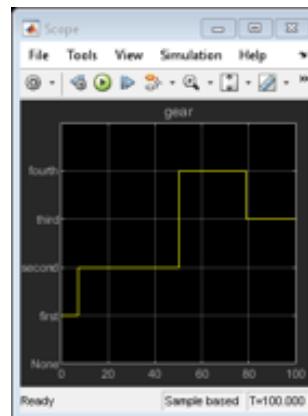
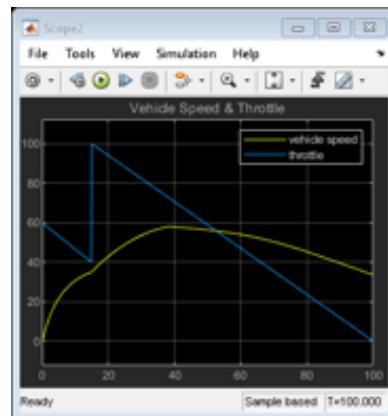
SISTEMI IBRIDI

= combinazioni di sistemi time-driven e event-driven

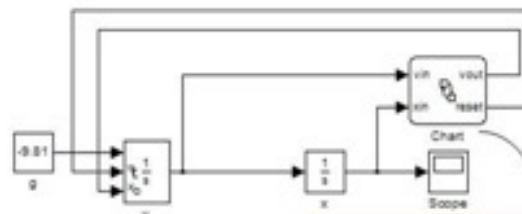
Stateflow = toolbox di simulazione per sistemi ibridi in ambiente Simulink



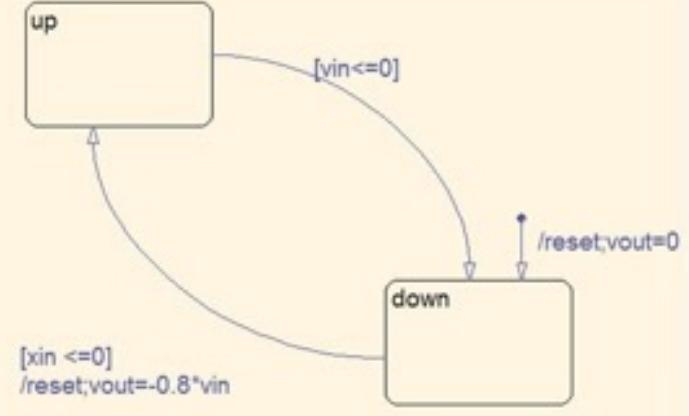
logica di una trasmissione automatica



controllo di un semaforo del traffico



modello di palla che rimbalza....



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 conflitto nella allocazione delle risorse**

Alcune domande a cui si cerca risposta

- quale 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 pianificate?
- cosa bisogna fare (o non fare) per evitare di finire in un blocco critico (**deadlock**)?
- come si devono gestire le 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)
- Grafcet/Sequential Functional Chart (SFC)

} vediamo essenzialmente questi tipi di modelli

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

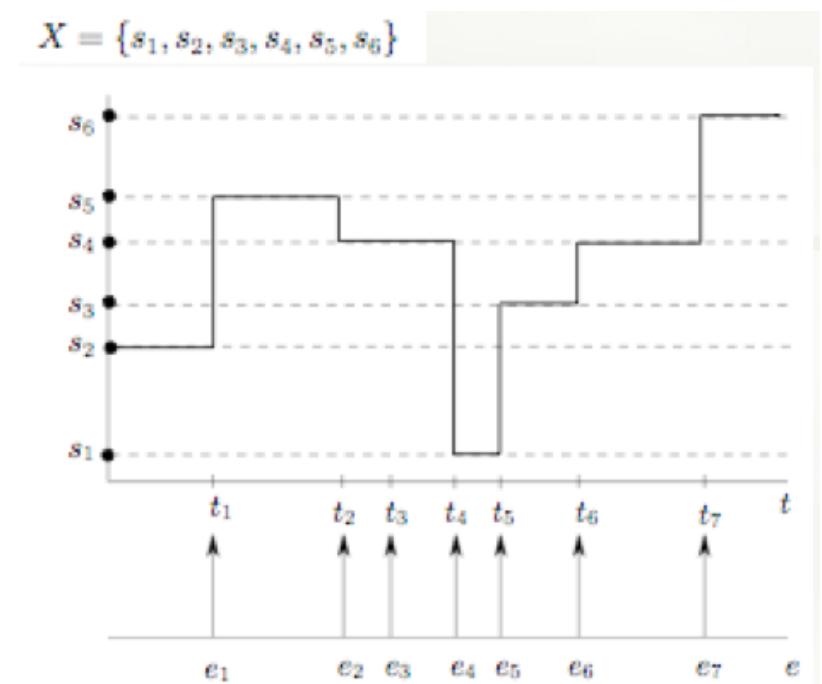
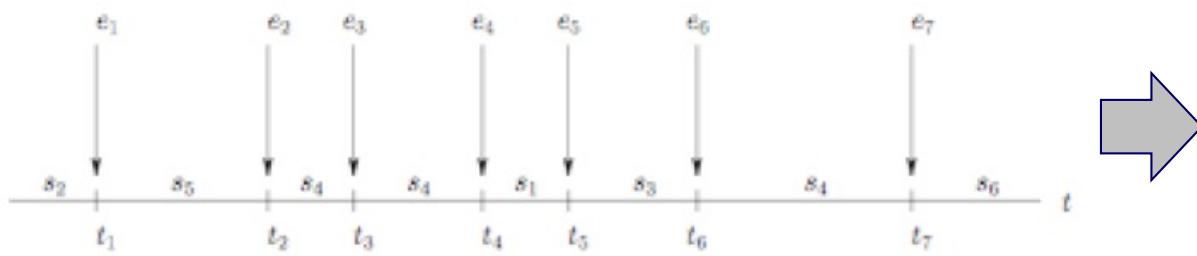
- basati su regole
- basati su equazioni

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

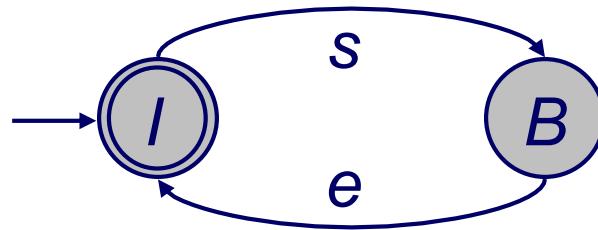
- sequenza di eventi $\{e_1, e_2, e_3, e_4, \dots\}$
- sequenza di eventi temporizzati $\{(t_1, e_1), (t_2, e_2), (t_3, e_3), (t_4, e_4), \dots\}$
- dato lo stato iniziale, una sequenza di eventi definisce in modo *univoco* l'evoluzione dello stato
- uso della “teoria dei linguaggi”: dato un *alfabeto E* di eventi (simboli), tutte le possibili sequenze di eventi ammissibili (*stringhe*) costituiscono un *linguaggio*
- 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'automa 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 $\Gamma(x)$ 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’automa G

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

↑
stringa vuota

tutte le stringhe generabili

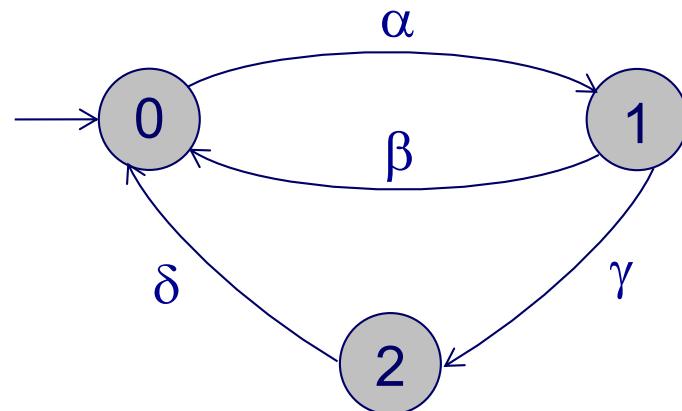
Un automa a stati finiti **con ingressi e uscite** è la sestupla $H = (U, X, Y, f, h, x_0)$ dove

- $U = \{u_1, u_2, u_3, \dots\}$ 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

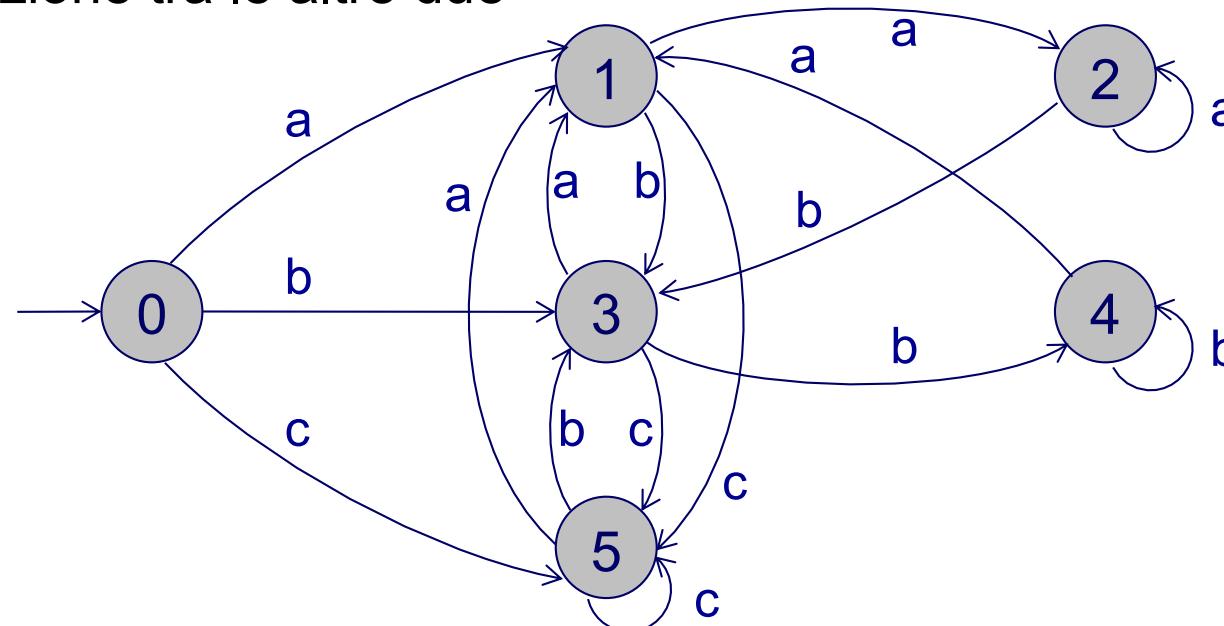
macchina utensile soggetta a guasti



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

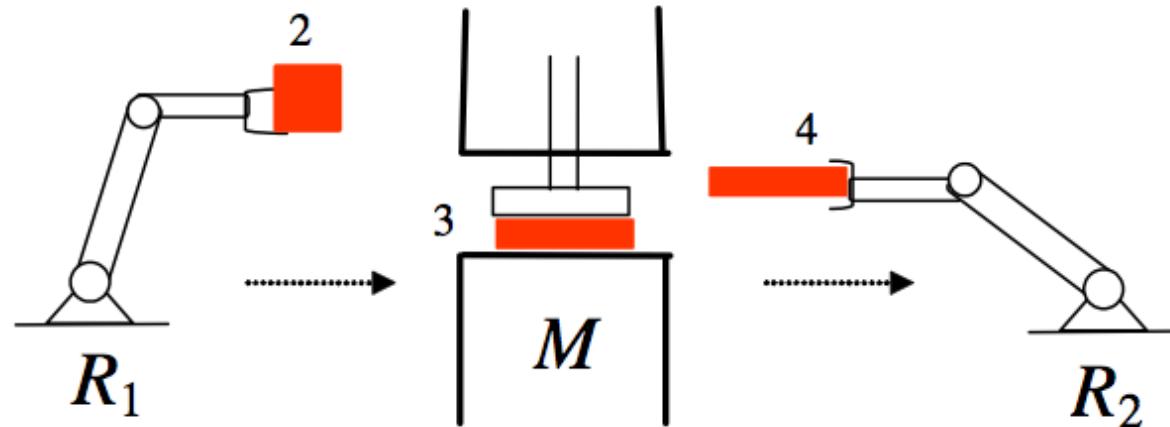
- stato $X = \{0 = \text{macchina libera}, 1 = \text{macchina occupata}, 2 = \text{macchina guasta}\}$
- eventi $E = \{\alpha = \text{inizio lavorazione}, \beta = \text{fine lavorazione}, \gamma = \text{la macchina si guasta}, \delta = \text{la macchina è riparata}\}$
- hp di lavoro
 - ➔ all'inizio del 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)

una macchina utensile esegue tre tipi di operazioni; per vincoli tecnologici, una delle tre operazioni **non** può essere eseguita subito **dopo due** consecutive esecuzioni di una 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 }

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



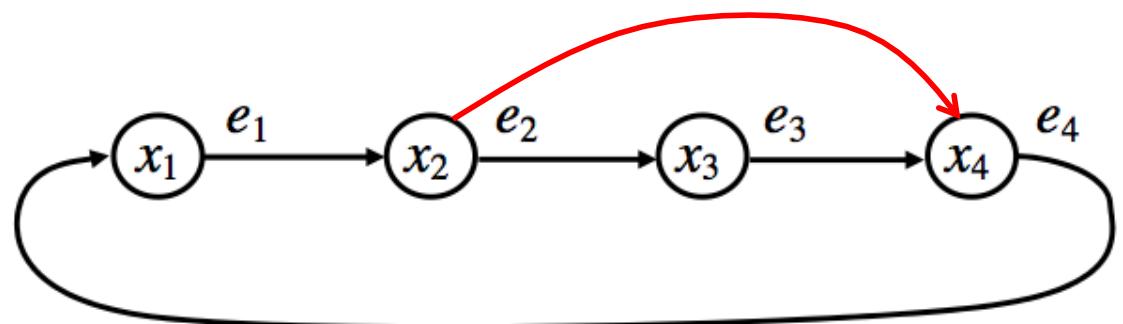
□ stati

- ➔ 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

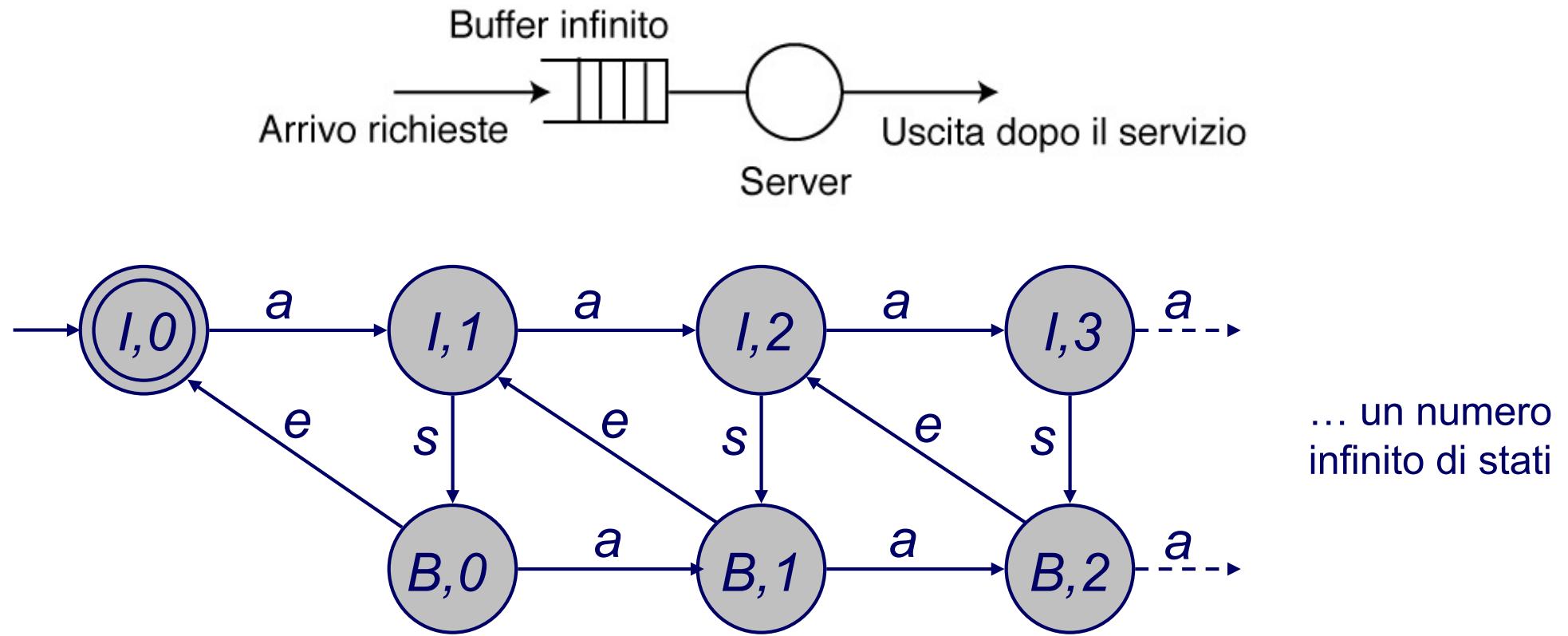
evento che vorremmo non accadesse ...
come "controllare" tale situazione?

□ eventi (fasi del ciclo produttivo)

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



modello del sistema client/server (con buffer illimitato)



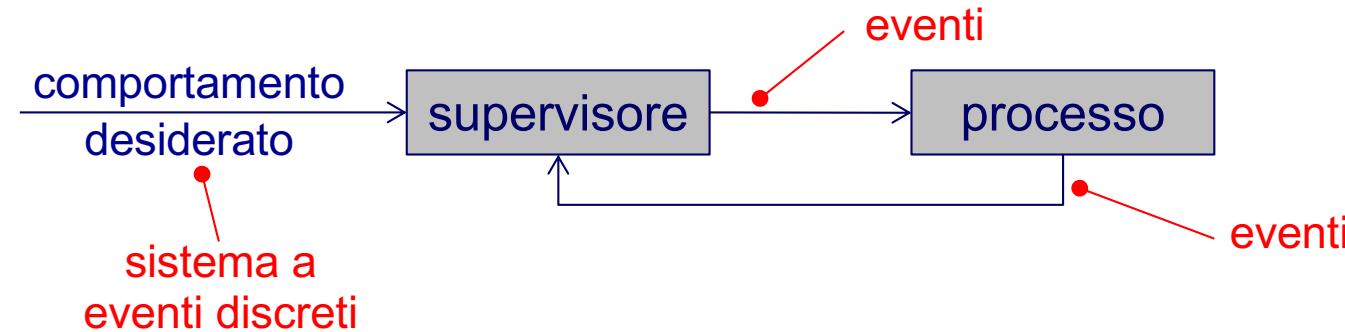
problema: gli automi **non** sono in grado di rappresentare con un numero finito di stati
alcuni “linguaggi” a cardinalità infinita

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 quello 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 intervallo di tempo T_c sono considerati come simultanei
 - l'ordine effettivo di tale eventi deve essere indifferente nel determinare la relativa azione di controllo

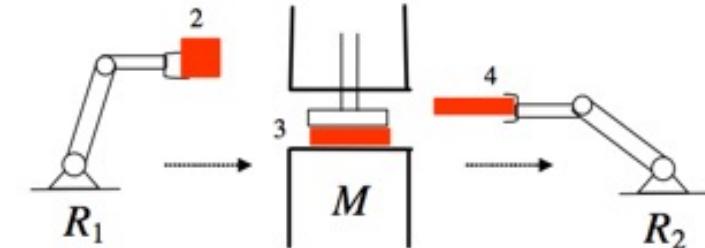
Modello con automa del supervisore

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 S**)

□ 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** (con tre moduli/automi)

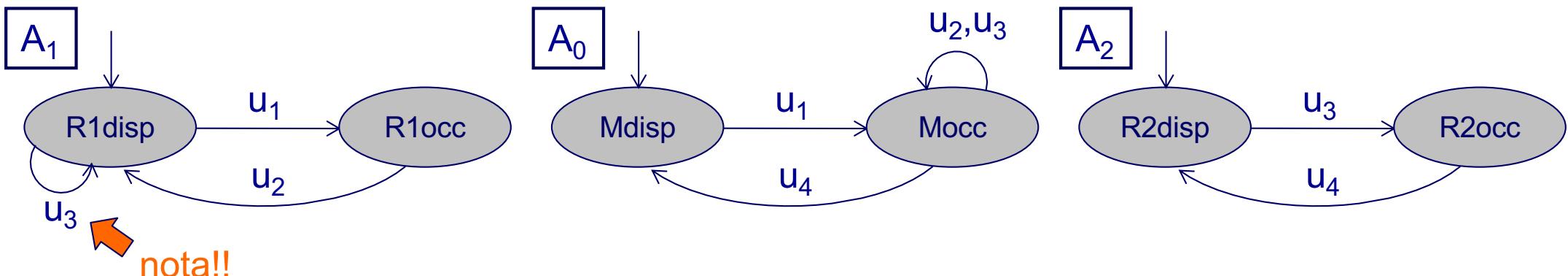
$$X = \{\text{Mocc}, \text{Mdisp}, \text{R1occ}, \text{R1disp}, \text{R2occ}, \text{R2disp}\}$$



□ gli eventi (in ingresso a S) 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'automa S che modellizza 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

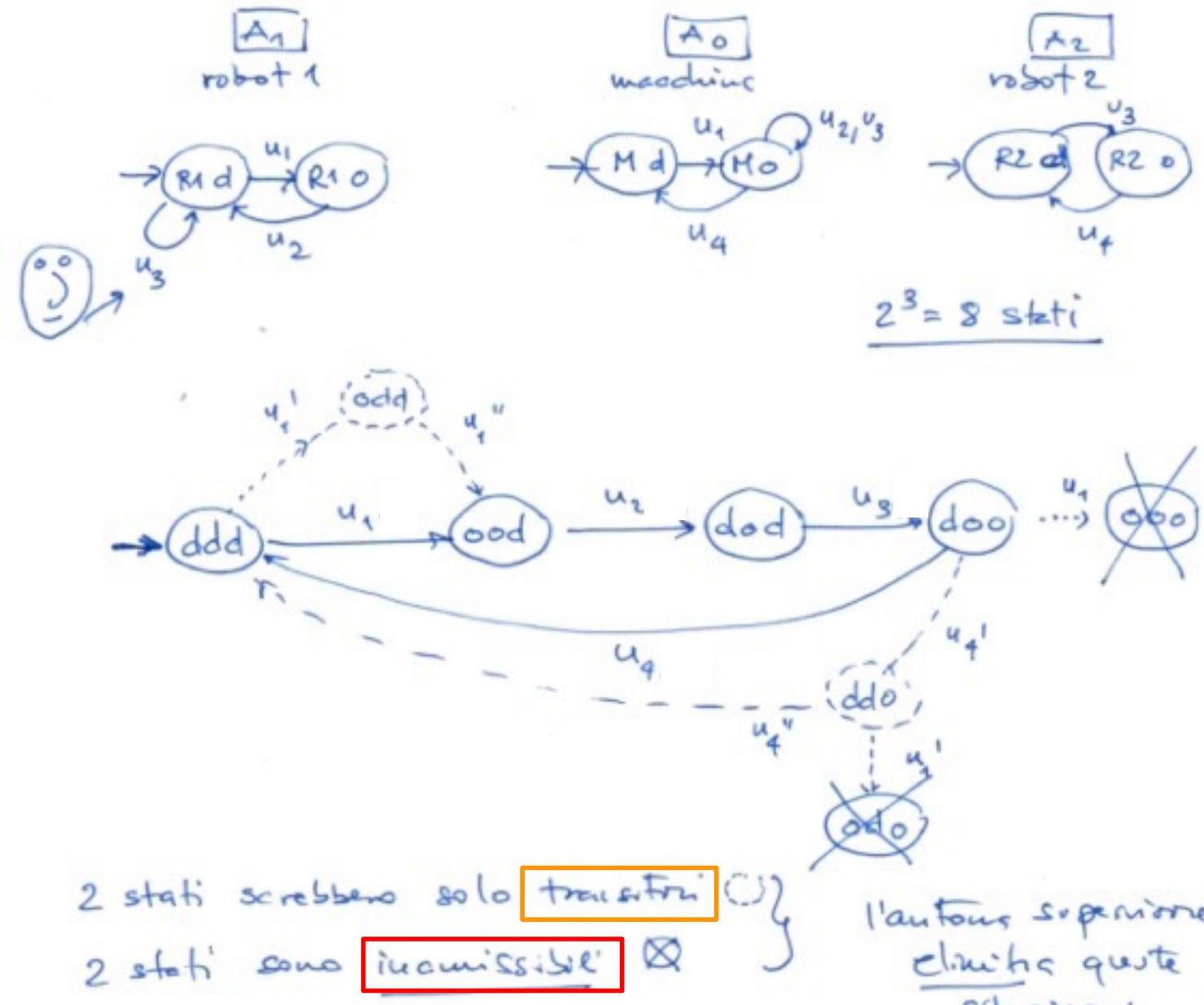
evoluzione dell'automa S

- nello stato iniziale x₁ l'unico evento che può avvenire è u₁ (in A₀ e A₁, non in A₂ perché è assente nel suo alfabeto): M e R₁ passano in stato occupato, R₂ resta disponibile ($\Rightarrow x_2$)
- nello stato x₂ l'evento u₂ è l'unico accettabile in A₀ e A₁ (in A₂ è assente): R₁ cambia stato e torna disponibile ($\Rightarrow x_3$)
- nello stato x₃ l'unico evento accettabile da tutti e tre gli automi è u₃: R₂ cambia stato e diventa occupato ($\Rightarrow x_4$)
- nello stato x₄ l'evento u₄ è l'unico accettabile in A₀ e A₂ (in A₁ è assente): anche M e R₂ tornano disponibili ($\Rightarrow x_1$)

affinché la composizione sincrona funzioni, è essenziale la presenza dell'evento u₃ nell'alfabeto di A₁ (else u₁ potrebbe scattare prima di u₃), anche se tale evento riguarda un'altra parte del processo!

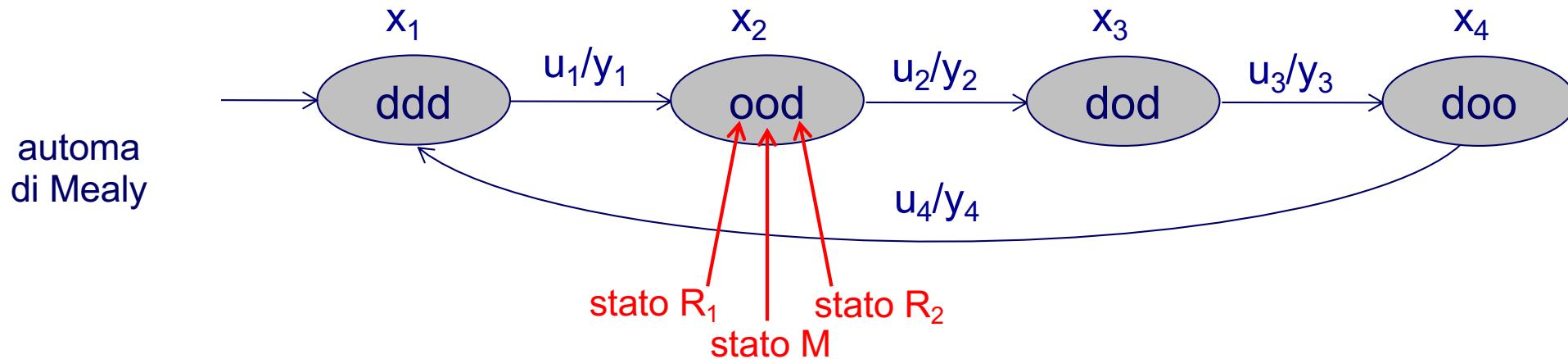
Modello con automa del supervisore

dove sono finiti
i restanti “possibili” stati
a seguito dell’integrazione
dei tre automi elementari?



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
 - 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 tutte le possibili sequenze di operazioni
 - scarsa modularità: aggiunta di dettagli complica subito l’automa
- una certa simmetria...

Modello con automa del supervisore

consideriamo un modello più dettagliato del processo, introducendo più stati per ciascun dispositivo:

componente del processo	stati
macchina M	Mdisp, Mcar, Mlav, Msca
robot R1	R1disp, R1prel, R1car, R1rit
robot R2	R2disp, R2sca, R2rit

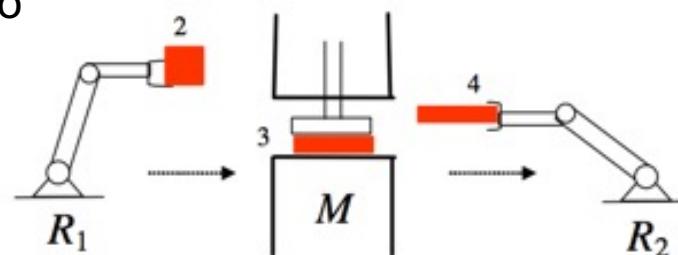
□ evoluzioni aggiuntive della macchina e dei due robot

- ➔ lo stato di “occupato” di M, si spezza nelle fasi di carico (Mcar), effettiva lavorazione (Mlav) e scarico del pezzo lavorato (Msca)
- ➔ il robot R₁ preleva un pezzo (R1prel), lo carica sulla macchina M (R1car) e ritorna nella posizione di stand-by (R1rit)
 - solo nello stato R1car è necessario sincronismo tra R₁ e M
 - negli altri stati, il funzionamento di R₁ è indipendente da quello degli altri dispositivi
- ➔ Il robot R₂ è in attesa di un pezzo lavorato su M da prelevare; quando ce n’è uno disponibile, R₂ scarica M (R2sca), lo porta alla coda di uscita e lo rilascia; ritorna infine in posizione di stand-by (R2rit).
 - solo nello stato R2sca è necessario sincronismo tra R₂ e M

Modello con automa del supervisore

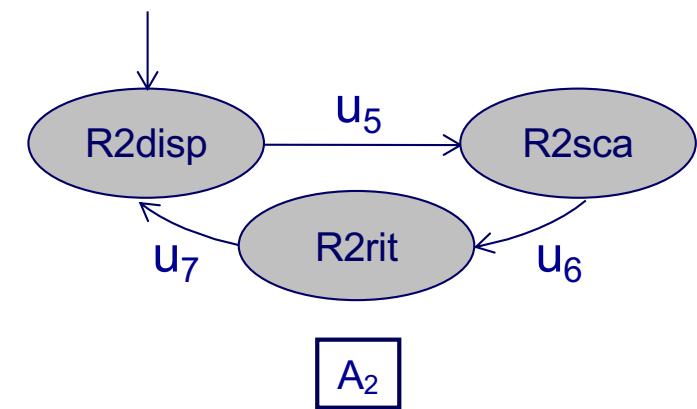
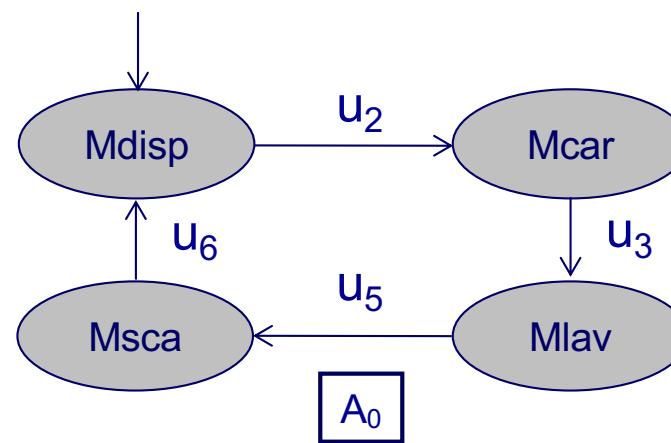
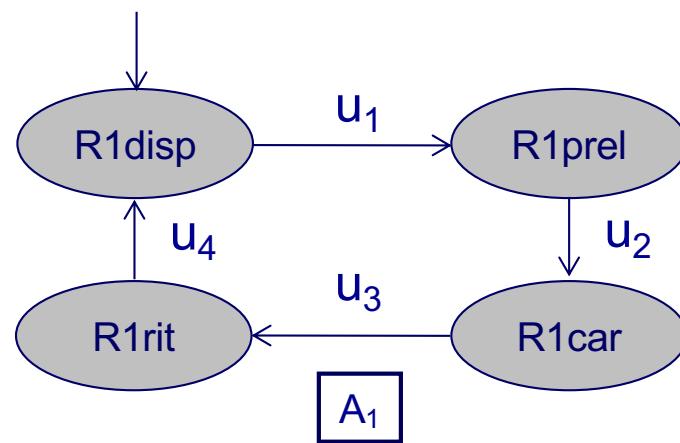
- gli eventi per gestire gli stati aggiuntivi del processo crescono di conseguenza

- u_1 pezzo disponibile al prelievo
- u_2 fine prelievo pezzo con R_1
- u_3 fine carico di M con R_1
- u_4 fine ritorno di R_1
- u_5 fine lavorazione di M
- u_6 fine scarico di M con R_2
- u_7 fine ritorno di R_2



- gli eventi che caratterizzano le nuove azioni di controllo del supervisore sono

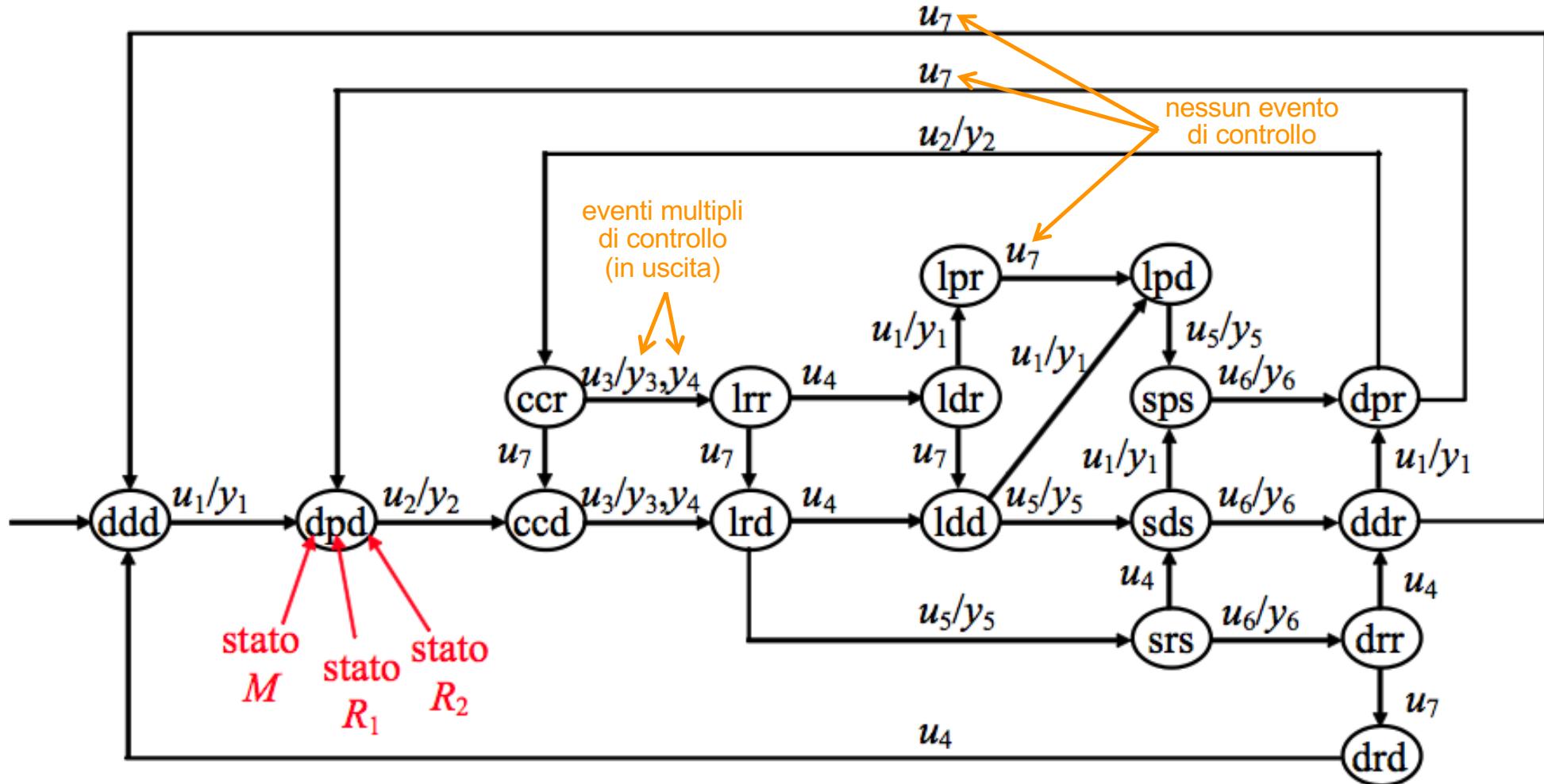
- y_1 inizio prelievo pezzo con R_1
- y_2 inizio carico di M con R_1
- y_3 inizio ritorno di R_1
- y_4 inizio lavorazione di M
- y_5 inizio scarico di M con R_2
- y_6 inizio ritorno di R_2



- componendo i 3 sotto-automi dettagliati, il modello si complica notevolmente
- gli stati sono dell'ordine di $4 \times 4 \times 3 = 48$ (quelli effettivamente raggiungibili sono 17)
- non c'è alcun legame con il modello precedente: occorre rigenerare l'automa da zero!

Modello con automa del supervisore

automa **con ingressi e uscite** (di Mealy) per descrivere in modo più dettagliato il processo (2 robot, 1 macchina) con il supervisore (gestore logico della lavorazione)



17 stati $x_i = \{ijk\}$, con $i \in \{d,c,l,s\}$, $j \in \{d,p,c,r\}$, $k \in \{d,s,r\}$, associati rispettivamente a stati di M , R_1 , R_2

Modello con automa del supervisore

tabella riassuntiva degli stati dell'automa

stato	stato M	stato R ₁	stato R ₂	code	descrizione
x ₁	Mdisp	R1disp	R2disp	ddd	M in attesa di lavorazione
x ₂	Mdisp	R1prel	R2disp	dpd	R ₁ preleva il pezzo
x ₃	Mcar	R1car	R2disp	ccd	R ₁ carica il pezzo su M
x ₄	Mlav	R1rit	R2disp	lrd	M lavora, R ₁ ritorna in posizione di riposo
x ₅	Mlav	R1disp	R2disp	ldd	M lavora
x ₆	Msca	R1disp	R2sca	sds	R ₂ scarica il pezzo da M
x ₇	Mdisp	R1disp	R2rit	ddr	R ₂ trasporta in uscita il pezzo e ritorna in posizione di riposo
x ₈	Msca	R1rit	R2sca	srs	R ₁ ritorna in posizione di riposo, R ₂ scarica il pezzo da M
x ₉	Mdisp	R1rit	R2rit	drr	R ₁ ritorna a riposo, R ₂ trasporta il pezzo e ritorna a riposo
x ₁₀	Mdisp	R1rit	R2disp	drd	R ₁ ritorna in posizione di riposo
x ₁₁	Msca	R1prel	R2sca	sps	R ₁ preleva il (nuovo) pezzo, R ₂ scarica il pezzo da M
x ₁₂	Mdisp	R1prel	R2rit	dpr	R ₁ preleva il (nuovo) pezzo, R ₂ trasporta pezzo e ritorna a riposo
x ₁₃	Mcar	R1car	R2rit	ccr	R ₁ carica il (nuovo) pezzo su M, R ₂ trasporta pezzo e ritorna a riposo
x ₁₄	Mlav	R1rit	R2rit	lrr	M lavora, R ₁ ritorna a riposo, R ₂ trasporta pezzo e ritorna a riposo
x ₁₅	Mlav	R1disp	R2rit	ldr	M lavora, R ₂ trasporta il pezzo e ritorna in posizione di riposo
x ₁₆	Mlav	R1prel	R2rit	lpr	M lavora, R ₁ preleva il (nuovo) pezzo, R ₂ trasporta e ritorna a riposo
x ₁₇	Mlav	R1prel	R2disp	lpd	M lavora, R ₁ preleva il pezzo