



## **Automazione industriale dispense del corso**

### **4. Sistemi dinamici ad eventi discreti**

Luigi Piroddi  
[piroddi@elet.polimi.it](mailto:piroddi@elet.polimi.it)

## Necessità modelli e approccio formale

Possiamo identificare due possibili approcci per il progetto di sistemi di automazione:

- 1 implementazione diretta
- 2 modellizzazione e approccio formale

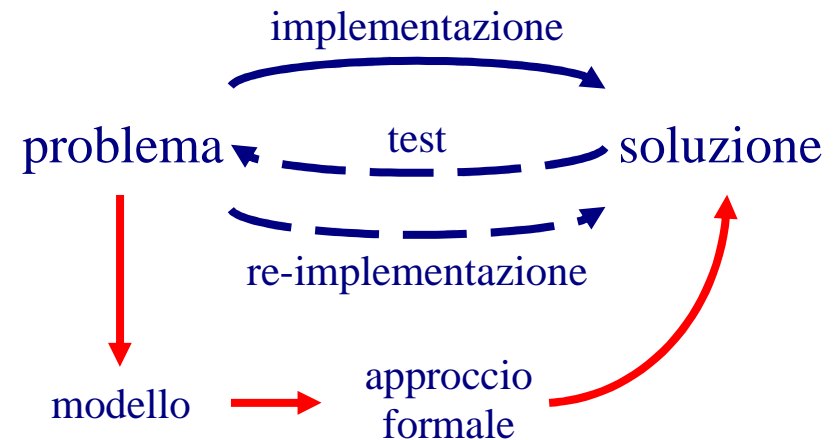
Nel primo approccio, si implementa direttamente il sistema di controllo e lo si valida mediante test sul campo.

Il test può consistere semplicemente nel far funzionare il sistema per un tempo sufficientemente lungo!

Con un approccio del genere, fare il debugging del sistema è lungo e laborioso e se cambiano le specifiche occorre rifare tutto il progetto.

Con il secondo approccio è possibile fornire garanzie sul rispetto delle specifiche di controllo, ed è molto più facile introdurre modifiche.

Al crescere della complessità del problema il secondo approccio risulta l'unico praticabile.





## Modelli matematici per l'automazione

Ci serve descrivere i processi ad un livello di astrazione elevato, in cui si evidenzino le *sequenze di operazioni*, con i relativi problemi di sincronizzazione, parallelismo, allocazione delle risorse, blocchi critici (deadlock), ecc.

In questa ottica, ci interessano le *caratteristiche macroscopiche di un sistema*, descrivibili con *condizioni logiche di funzionamento discrete*, come ad es. “macchina pronta per la lavorazione”, “macchina in attesa”, “macchina guasta”.

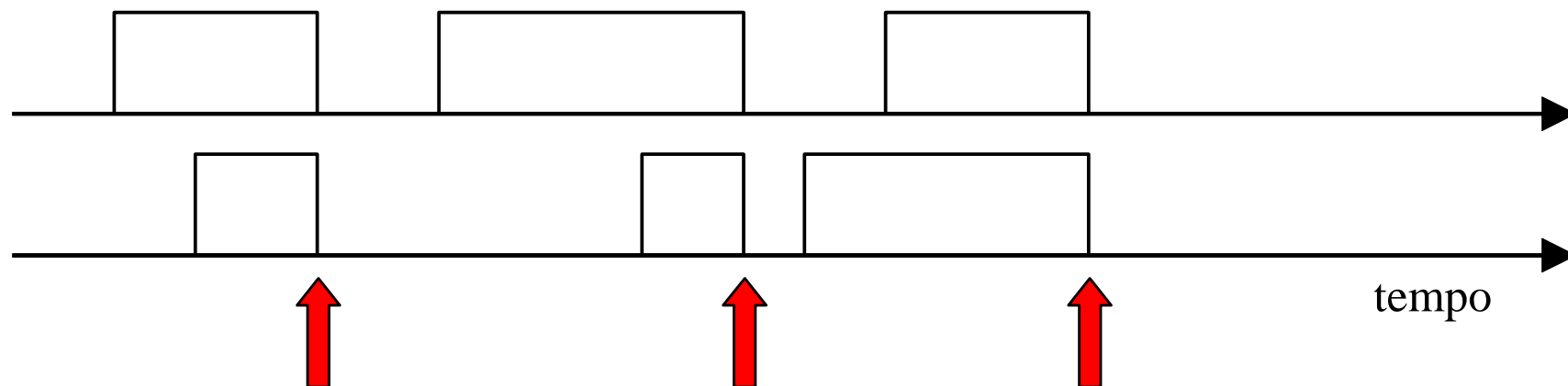
Tali condizioni logiche individuano lo *stato* del sistema, che può assumere un numero finito di valori (*stato discreto*), e può essere descritto numericamente (mediante una quantità finita o numerabile di valori) o in termini simbolici (mediante parole, stringhe, ecc.).

Lo stato del sistema cambia solo in certi istanti, con transizioni istantanee da un valore all'altro (p.es. da “macchina occupata” a “macchina libera”), in corrispondenza dell'occorrenza di *eventi*, quali ad es.

- ▶ un *comando* “accendi la macchina”, oppure 
- ▶ un *segnale* di “fine corsa raggiunto” 

Si immagini di rappresentare con un segnale binario lo stato di un pulsante di attivazione (0 = non premuto, 1 = premuto) e che il sistema reagisca al rilascio del pulsante.

In queste ipotesi, i due *segnali* raffigurati, che sono ovviamente diversi istante per istante, sono tuttavia caratterizzati dalla stessa sequenza di *eventi*.



Partendo dal medesimo stato iniziale tali ingressi produrranno lo stesso andamento dello stato!

Anche gli eventi sono discreti e possono essere descritti in termini non numerici.

Normalmente, non è noto a priori né quale sia il nuovo valore dello stato, né l'istante temporale in cui avviene il cambiamento.

L'evoluzione di tali sistemi può allora essere caratterizzata in termini di cambiamento delle condizioni logiche di funzionamento discrete (stato), per effetto di sequenze di eventi (comandi/segnali).

Lo strumento che ci serve è una qualche forma di *sistema dinamico*, perché solo conoscendo la “storia” del sistema (conoscenza dello *stato iniziale*) posso determinare qual è la prossima operazione da compiere.

Sistemi con queste caratteristiche prendono il nome di *sistemi (dinamici) ad eventi discreti* (in inglese *Discrete Event (Dynamic) Systems (DE(D)S*).

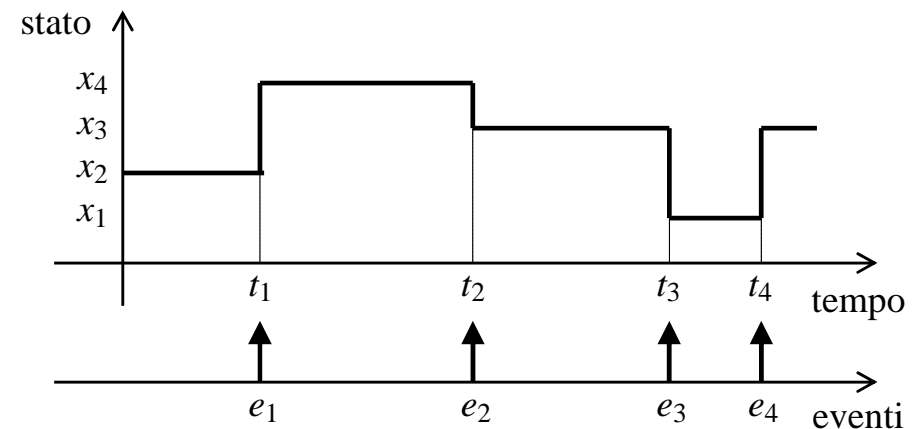
Problematiche relative alla modellizzazione con sistemi ad eventi discreti:

- ▶ non c'è convergenza su *metodi e modelli di rappresentazione*
- ▶ anche nell'ambito dello stesso formalismo (v. reti di Petri) esistono tipologie di modelli e tecniche modellistiche molto varie
- ▶ i sistemi ad eventi discreti servono per modellizzare sistemi ad un livello di astrazione elevato, al quale non esistono dei chiari riferimenti modellistici come i “principi primi” fisici di conservazione di massa, energia e quantità di moto

## Sistemi ad eventi discreti

Un sistema ad eventi discreti può essere studiato analizzando i suoi movimenti (dello stato).

Nel caso raffigurato, lo stato assume solo i quattro valori  $x_1$ ,  $x_2$ ,  $x_3$  e  $x_4$ , e cambia solo in alcuni istanti ( $t_1$ ,  $t_2$ ,  $t_3$ ,  $t_4$ ), in corrispondenza degli eventi  $e_1$ ,  $e_2$ ,  $e_3$ ,  $e_4$ .



Assumendo che lo stato iniziale sia noto e che il sistema sia deterministico (ovvero, che la legge che determina lo stato successivo in corrispondenza di un evento sia univoca), l'informazione aggiuntiva necessaria per calcolare il movimento dello stato è fornita dalla sequenza (temporizzata) di eventi:

$(t_1, e_1) (t_2, e_2) (t_3, e_3) (t_4, e_4)$

Dal punto di vista del comportamento logico del sistema, non ci interessa tanto *quando* il sistema entra in un determinato stato o *quanto a lungo* il sistema rimane nel medesimo stato, ma piuttosto l'ordinamento delle transizioni di stato.

Al fine di modellizzare il *comportamento logico* del sistema, ci interessa perciò la *sequenza* di eventi (che determinano la sequenza di transizioni di stato):

$e_1 e_2 e_3 e_4$

Se interpretiamo l'insieme di tutti gli eventi che possono occorrere al sistema ad eventi discreti come un *alfabeto*, la sequenza  $e_1 e_2 e_3 e_4$  è una *stringa*, che fa parte del *linguaggio* associato al sistema:

- ▶ alfabeto:  $E = \{e_1, e_2, e_3, e_4, \dots\}$
- ▶ stringa:  $s = e_2 e_1 e_2 e_4 e_3$
- ▶ linguaggio:  $\mathcal{L}$  = insieme di stringhe compatibili con il sistema  
(che ne determinano tutte le possibili evoluzioni)

Il linguaggio:

- ▶ è una descrizione esplicita e completa del comportamento del sistema  
(non necessariamente la più agevole da usare in pratica)
- ▶ se l'informazione temporale non può essere omessa si parla di *linguaggio temporale*
- ▶ se gli istanti di occorrenza degli eventi non sono definiti in modo deterministico, ma sono variabili aleatorie, si parla di *linguaggio temporale stocastico*

Un sistema ad eventi discreti può essere descritto con un *modello operativo*, ovvero un modello con una struttura di transizione che descriva esattamente cosa succede nel sistema (p.es. dallo stato  $x_1$  si va allo stato  $x_2$  se accade l'evento  $e_1$ ):

- ▶ automi
- ▶ reti di Petri
- ▶ Grafcet/SFC

In particolare, le reti di Petri (e in parte il Grafcet/SFC) risultano particolarmente utili per comprendere e rappresentare esplicitamente i meccanismi interni di funzionamento del sistema, come parallelismo, condivisione di risorse, scelte, asincronia, ecc., che determinano l'evoluzione dello stato.

In alternativa, si possono utilizzare *modelli dichiarativi*, ovvero modelli in cui vengono dichiarati i comportamenti del sistema (p.es. le uscite  $y_1$  e  $y_2$  non possono essere contemporaneamente al valore logico alto):

- ▶ modelli basati su regole
- ▶ modelli basati su equazioni



In linea di principio, per rappresentare sistemi ad eventi discreti, si potrebbe utilizzare direttamente un linguaggio di programmazione, che ha le seguenti importanti caratteristiche:

- ▶ è un modello formale (e analizzabile in modo formale)
- ▶ è eseguibile (su una determinata macchina)
- ▶ esistono supporti di sviluppo (debug, case, ecc.)
- ▶ non ha problemi di traduzione: è già pronto per l'implementazione del controllo!

Ma la descrizione e lo studio di un sistema ad eventi discreti tramite linguaggio di programmazione:

- ▶ è un lavoro complesso e troppo dettagliato
- ▶ non favorisce l'astrazione dei concetti principali
- ▶ non ha strutture modellistiche standard
- ▶ i relativi “modelli” non sono portabili da un sistema ad un altro

## Modelli logici e temporizzati

I modelli logici servono a capire se e come il sistema evolve (stati di blocco, ecc.) per il loro studio ci sono strumenti di analisi formale (ad esempio si può stabilire a priori se sono possibili dei blocchi critici). Un modello logico consente di:

- ▶ discriminare le sequenze di eventi compatibili con delle specifiche di comportamento
- ▶ verificare se un determinato stato è raggiungibile, e con quale sequenza di eventi
- ▶ verificare se il sistema si blocca in uno stato

I modelli temporizzati servono ad analizzare le prestazioni (tempi di lavorazione, pezzi prodotti nell'unità di tempo, ecc.).

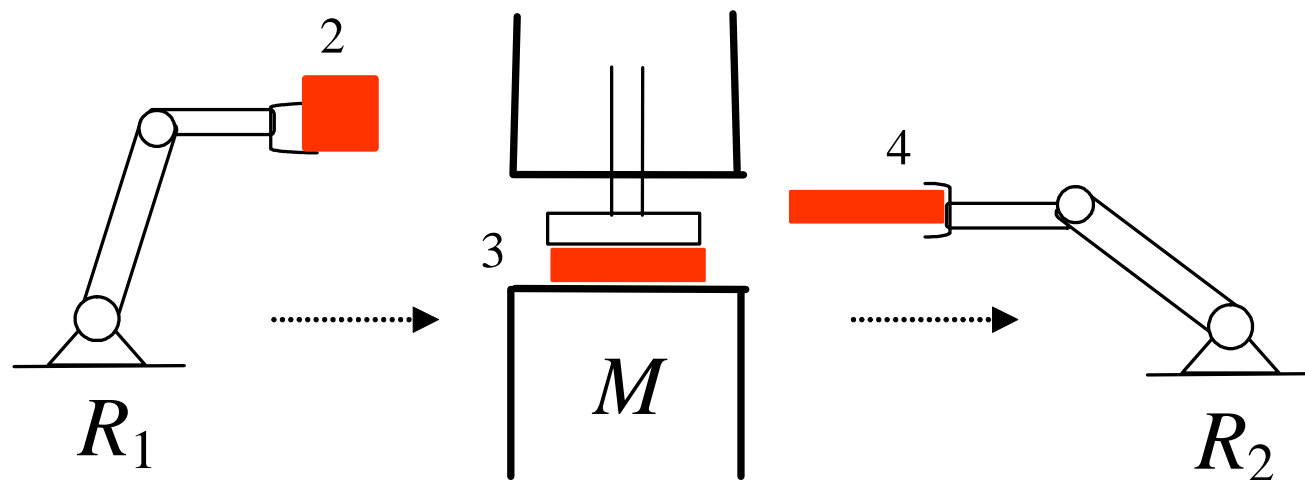
Non ci sono strumenti di analisi potenti come per i modelli logici, e spesso bisogna ricorrere alla simulazione.

Un modello temporizzato consente di rispondere a domande come le seguenti:

- ▶ quanto tempo spende il sistema in un determinato stato?
- ▶ qual è il tempo minimo in cui uno stato può essere raggiunto?
- ▶ può una sequenza di eventi essere completata in un tempo assegnato?
- ▶ quanto dura il tempo di ciclo di una sequenza di lavorazioni?

## Esempio: modellizzazione di una macchina

Consideriamo una sezione di un sistema manifatturiero in cui una macchina  $M$  venga caricata con un pezzo grezzo da lavorare mediante un robot manipolatore  $R_1$ , effettui una lavorazione specifica, e al termine della lavorazione un altro robot ( $R_2$ ) prelevi il prodotto finito.



Noi siamo interessati ad un modello del funzionamento della macchina che ne spieghi la sequenza di operazioni (modello logico). Dobbiamo schematizzare:

- ▶ gli *stati logici*
- ▶ gli *eventi* associati alle *transizioni* da uno stato all'altro

Per esempio, possiamo definire 4 stati:

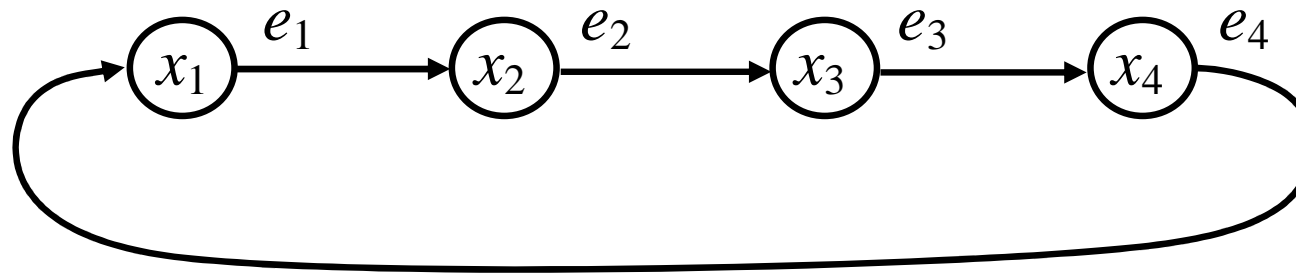
- $x_1$ ) macchina  $M$  in attesa, disponibile per la lavorazione
- $x_2$ ) carico del pezzo su  $M$  da parte di  $R_1$
- $x_3$ ) lavorazione
- $x_4$ ) scarico del pezzo lavorato da  $M$  tramite  $R_2$

In questo caso, sono 4 anche le possibili transizioni di stato, da ciascun stato al successivo (si tratta infatti delle 4 fasi di un ciclo produttivo).

Ogni transizione avviene in corrispondenza dell'occorrenza di un evento, corrispondente al comando di attivazione dell'operazione successiva:

- $e_1$ ) inizio carico
- $e_2$ ) inizio lavorazione
- $e_3$ ) inizio scarico
- $e_4$ ) inizio attesa

Il funzionamento logico della macchina può essere rappresentato in forma di grafo orientato in cui i nodi rappresentano gli stati e gli archi le transizioni.



Abbiamo così ottenuto quello che si chiama un *automa a stati finiti*, nella sua rappresentazione grafica.

## Controllo di sistemi ad eventi discreti

Nel contesto del controllo, non possiamo più fare riferimento a semplici modelli isolati, dato che occorre operare con più sistemi

- ▶ il controllore
- ▶ il processo da controllare

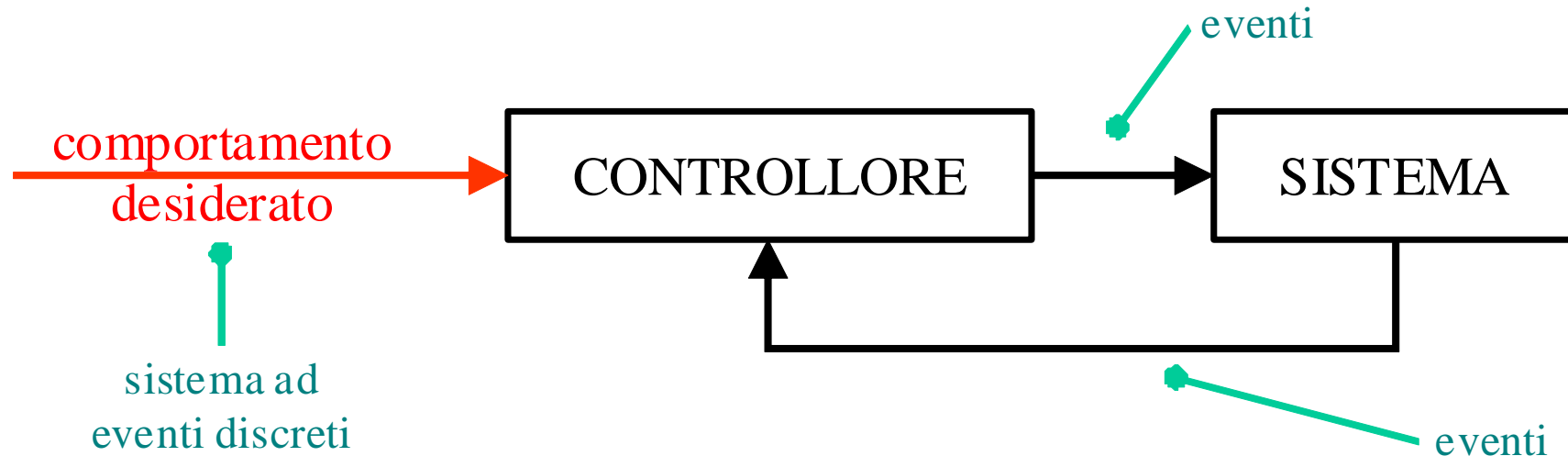
che interagiscono e quindi si scambiano informazioni.

In particolare:

- ▶ il controllore *riceve informazioni* dal processo (*misure*)
- ▶ il controllore *invia ordini* al processo (*comandi*)
- ▶ il sistema complessivo deve avere un *comportamento desiderato*

Occorre perciò usare dei *modelli non autonomi*, ovvero con variabili di ingresso e uscita, che siano in grado, tramite tali variabili, di interagire tra di loro.

Come facilmente prevedibile, l'interazione tra processo e controllore avviene nei due sensi, configurando un anello di *retroazione*.



Tuttavia, il problema del controllo di sistemi ad eventi discreti si differenzia in modo consistente da quello del controllo di sistemi continui.

E ciò richiede un approccio mentale e concettuale e degli strumenti di analisi e di sintesi significativamente diversi da quelli visti per l'analisi e il controllo di sistemi continui.

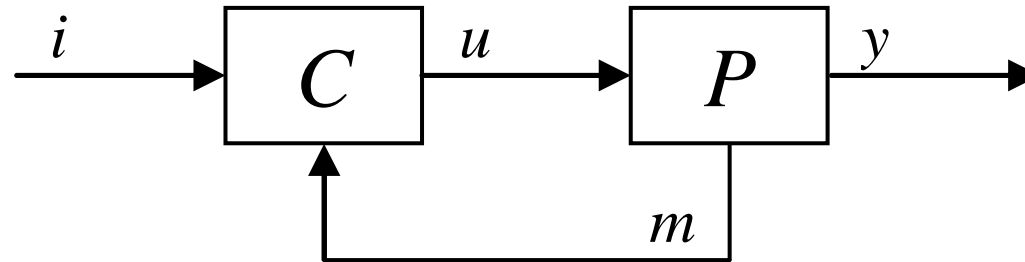
## Principali differenze rispetto al controllo di sistemi continui:

- ❶ Le informazioni scambiate tra sistema e controllore non sono segnali, ma *eventi*.  
Tipicamente, la conclusione di un'operazione sul sistema si traduce in un evento trasmesso dal sistema al controllore, che a sua volta genera degli eventi di comando per eseguire delle operazioni sul sistema.
- ❷ Non esiste l'analogo del segnale di riferimento che determini l'obiettivo del sistema di controllo. E non esistono indici che consentano di quantificare le prestazioni (come il margine di fase). La specifica di comportamento si definisce attraverso:
  - ▼ *vincoli da non violare*  
come ad esempio la mutua esclusione nell'uso di una macchina, il non bloccaggio del sistema (deadlock = blocco critico)
  - ▼ *modello del comportamento desiderato*  
ovvero un sistema ad eventi discreti che rappresenta *l'evoluzione logica desiderata* del sistema, nel rispetto dei requisiti del progetto; è strutturalmente omogeneo al modello del sistema, ma alcune transizioni di stato non desiderate sono inibite sulla base dello stato corrente e del comportamento desiderato: esso rappresenta quindi una *“restrizione” del comportamento dinamico del sistema da controllare*.



## Modelli di sistemi ad eventi discreti retroazionati

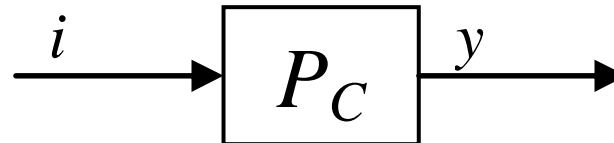
Il generico sistema ad eventi retroazionato è schematizzato nel modo seguente:



dove:

- ▶  $C$  = modello a eventi discreti del controllore  
(da implementare poi con un *programma*)
- ▶  $P$  = modello a eventi discreti del processo da controllare
- ▶  $i$  = variabili di ingresso
- ▶  $u$  = variabili di controllo
- ▶  $m$  = variabili misurate
- ▶  $y$  = variabili di uscita

Complessivamente, il sistema controllato è a sua volta un sistema ad eventi non autonomo:



che sperabilmente coincide con il sistema ad eventi ( $Sp$ ) che descrive le specifiche del comportamento desiderato.

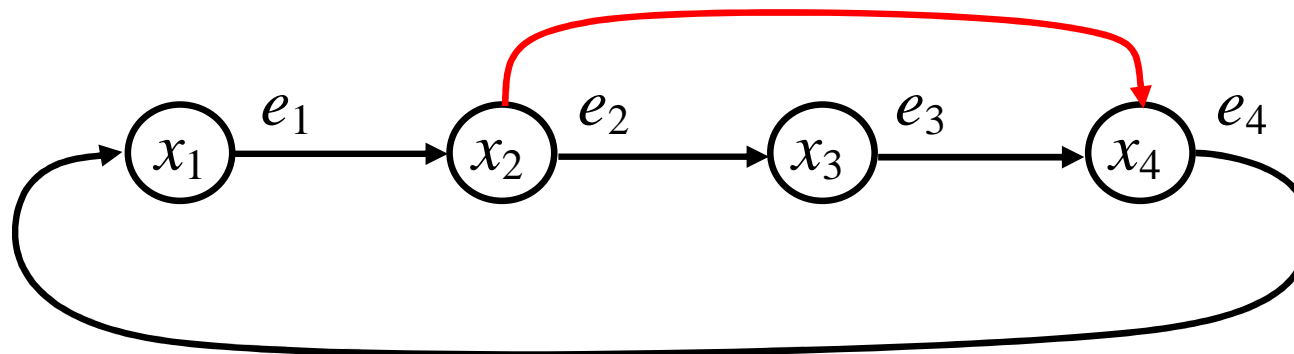
Ci sono alcune importanti differenze rispetto ai classici schemi di retroazione di sistemi continui:

- ▶ le variabili di ingresso non sono variabili di riferimento, variabili da inseguire, ma generici ingressi esogeni utilizzati per modellizzare le impostazioni dell'utente (o di un livello superiore della gerarchia di controllo), come per esempio il comando di attivazione di una ricetta di lavorazione
- ▶ il comportamento desiderato non è l'andamento delle variabili  $i$ , ma il modello a eventi discreti  $Sp$

## Esempio: modellizzazione logica di una macchina (continuazione)

L'automa a 4 stati visto in precedenza è in realtà il *modello del comportamento desiderato* del sistema macchina, che in assenza di controllo potrebbe avere molti più stati e transizioni, rappresentativi di altri comportamenti ammissibili.

Si immagini, per esempio, cosa succederebbe se dallo stato  $x_2$  si passasse direttamente allo stato  $x_4$ .



Il pezzo transiterebbe sulla macchina, ma senza essere sottoposto alla lavorazione.

Un comportamento, in linea di principio, perfettamente compatibile con il funzionamento dell'impianto descritto, ma certo non voluto dal progettista!

Il primo automa a 4 stati specifica, tra le evoluzioni possibili del sistema macchina, quella che *noi* consideriamo corretta.

Il controllore dovrà monitorare lo stato del sistema per consentire solo le transizioni di stato descritte dall'automa: in quanto tale lo possiamo identificare con l'automa stesso del comportamento desiderato.

***controllore = sistema ad eventi discreti del comportamento desiderato***

Oltre a dare i comandi giusti, il controllore dovrà assicurare che un'operazione venga attivata solo quando la precedente si è conclusa.

In realtà, quindi, alle transizioni dell'automa dovranno essere associati anche gli eventi trasmessi dall'impianto al controllore, che segnalano la conclusione di un'operazione (altrimenti non si chiude l'anello di retroazione).

Ad esempio, se il sistema si trova nello stato  $x_2$ , quando l'operazione di carico termina il controllore genera un evento di comando che trasmette alla macchina l'ordine di inizio lavorazione.

Contestualmente, lo stato del controllore commuta da  $x_2$  a  $x_3$  coerentemente con quello che succede nella macchina.

## Controllo e supervisione

Ci sono due scuole di pensiero diverse sull'interpretazione da dare alla realizzazione dell'azione di controllo:

① *controllo* (controllore come *boss*)

- ▼ il controllore definisce le sequenze di lavorazione per i dispositivi che compongono l'impianto, inoltrando al momento opportuno gli opportuni comandi

② *supervisione* (controllore come *vigile*)

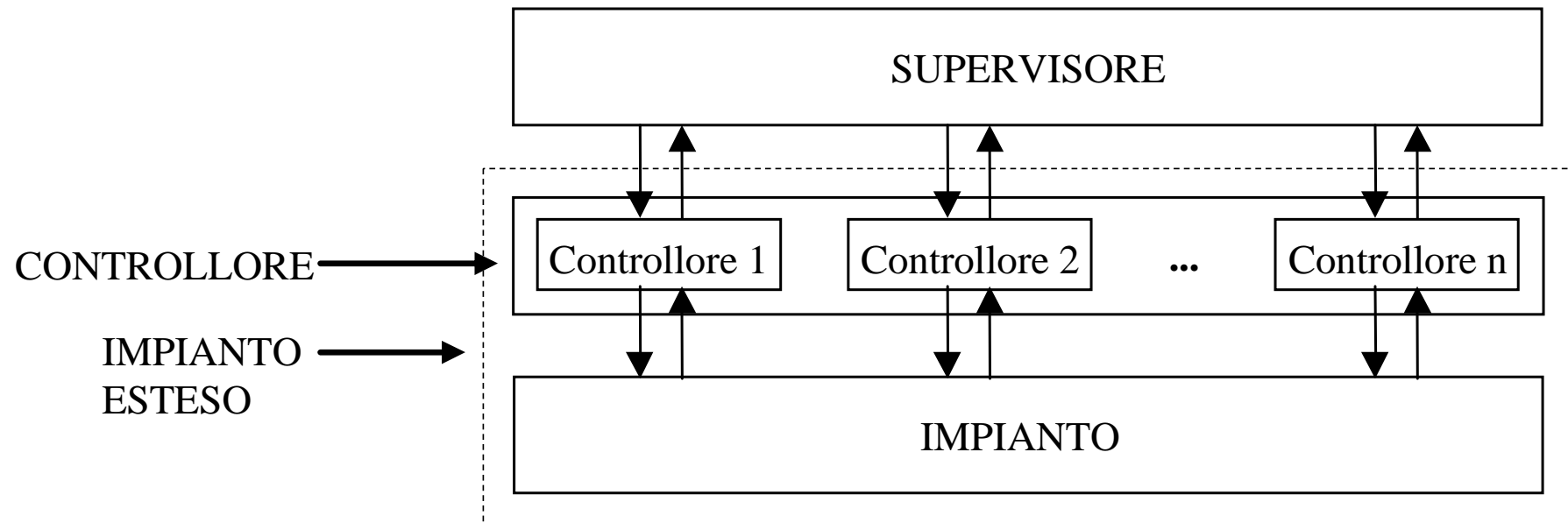
- ▼ il controllore interviene sul sistema da controllare *inibendo* alcuni eventi sulla base dello stato osservato
- ▼ in quanto tale, il controllore non decide l'esatta sequenza degli eventi, ma si limita ad imporre il rispetto dei vincoli
- ▼ il supervisore non impone un comportamento preciso al sistema da controllare, che viene lasciato libero di evolvere secondo le sue modalità di funzionamento
- ▼ esso interviene solamente al fine di evitare che il sistema finisca in stati di blocco parziale o totale

La differenza concettuale è molto sfumata: emettere un comando equivale a non inibire l'evento corrispondente e inibire tutti gli altri eventi.

Dal punto di vista operativo le cose sono molto diverse:

- ▶ l’approccio supervisivo risulta naturale quando il sistema da controllare è già dotato di alcuni automatismi (ovvero di *controllori*) ed è quindi in grado di evolvere autonomamente, e si vogliono imporre dei vincoli specifici al comportamento del sistema
- ▶ in una configurazione di controllo di un sistema complesso, possono avere spazio entrambe le funzioni, quella di controllo vero e proprio e quella di supervisione
- ▶ ad esempio, in un sistema manifatturiero,
  - ▼ le macchine operatrici sono tipicamente dotate di controllori che definiscono le sequenze principali di lavorazione (gestendole come se fossero indipendenti l’una dall’altra), mentre
  - ▼ un supervisore verifica, tipicamente sulla base dello stato complessivo del sistema, che l’esecuzione simultanea e concorrente delle varie sequenze non crei blocchi critici (deadlock) nel sistema, qualunque sia l’ordine di accadimento degli eventi

## Gerarchia del sistema di supervisione e controllo



Si noti che il supervisore non vede l'impianto reale, ma un *impianto esteso*, cioè l'impianto parzialmente controllato.

## Osservazioni:

- ▶ l'impianto esteso ha un numero di ingressi e uscite sensibilmente inferiore rispetto all'impianto reale, in modo tale da limitare la complessità e la dimensione del supervisore
- ▶ infatti, mentre i modelli dei singoli controllori sono strutturalmente semplici (tipicamente sequenze), il modello del supervisore può essere assai intricato, poiché deve servire per l'allocazione delle risorse a tutte le possibili sequenze di produzione
- ▶ se le operazioni sono rappresentate in maniera sufficientemente sintetica nel supervisore il modello risulta facile da analizzare per l'individuazione di stati indesiderati
- ▶ la gerarchia individuata permette di ottenere un sistema modulare (dal punto di vista del supervisore il comportamento dettagliato delle macchine è “incapsulato” nel livello sottostante dei controllori)

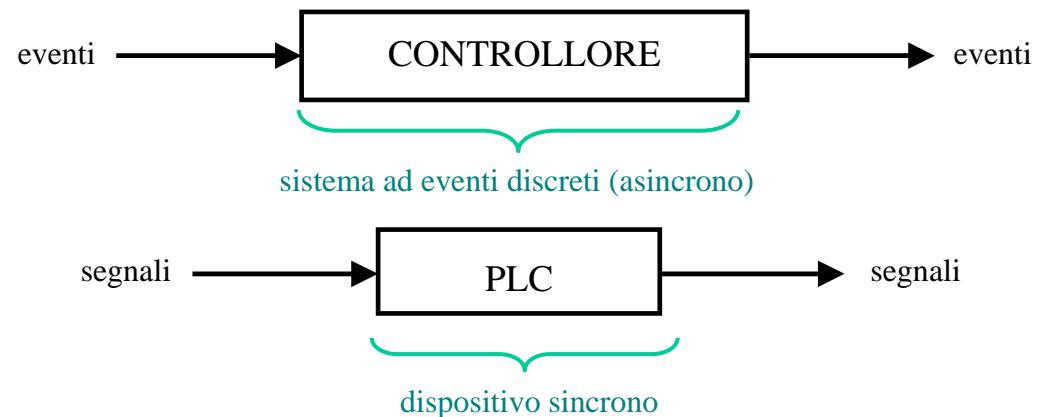


## Implementazione del controllore

Il controllore è esso stesso un sistema ad eventi discreti, che evolve in modo asincrono e interagisce con il sistema da controllare attraverso eventi.

Nasce allora il problema di come implementare un controllore siffatto in un calcolatore dedicato come il PLC, che per sua natura

- ▶ è un dispositivo *sincrono* che esegue ciclicamente un insieme di istruzioni
- ▶ interagisce con il processo scambiando dei *segnali*



Occorre quindi risolvere i seguenti problemi concettuali e pratici:

- ❶ bisogna progettare un'*interfaccia eventi-segnali* per tradurre le informazioni del campo in una forma compatibile con il modello (e viceversa)
- ❷ bisogna *sincronizzare* in modo opportuno l'evoluzione del sistema ad eventi discreti con quella del calcolatore che lo realizza

## Comportamento real-time

Problematiche relative alla gestione sincrona di eventi asincroni (legata all'implementazione time-driven del controllore):

- ▶ nell'intervallo di tempo tra due rilevazioni successive degli ingressi il sistema è cieco; se un segnale logico cambia stato due volte all'interno dello stesso periodo di campionamento, l'evento è perso; occorre pertanto fare l'ipotesi che la permanenza minima di un segnale logico in uno stato sia maggiore del tempo di ciclo
- ▶ nel caso peggiore, un evento accade appena dopo l'istante di clock e viene quindi rilevato con un ritardo di un passo; se entro il passo successivo viene eseguita l'azione di risposta relativa, il ritardo sarà stato al più di 2 periodi di campionamento (tempo di risposta massimo del sistema, cui in realtà vanno sommati i ritardi dei moduli I/O); occorre perciò fare l'ipotesi che tale ritardo del controllo sia tollerabile dal sistema; il comportamento real-time del sistema è garantito a patto che l'unità di elaborazione riesca sempre a completare le attività in esecuzione in un tempo di ciclo
- ▶ più eventi riscontrati nello stesso tempo di ciclo vengono interpretati come simultanei: il loro ordine di accadimento è mascherato dal rilevamento sincrono; occorre pertanto fare l'ipotesi che il loro ordine sia indifferente nel determinare l'azione di controllo corrispondente

## Progetto del controllore

Formulazione del problema del progetto del controllore:

*“Dato un modello  $P$  di un impianto e una specifica  $Sp$  di comportamento desiderato, progettare un controllore  $C$  tale che, posto in retroazione con  $P$ , dia luogo ad un sistema il più possibile prossimo ad  $Sp$ .”*

Osservazioni:

- ▶ il comportamento di  $Sp$  deve essere una *restrizione* di quello di  $P$  (che rappresenta il sistema non controllato)
- ▶ controllare un sistema vuol dire vincolarlo ad assumere, tra tutti i comportamenti possibili, uno il più vicino possibile a quello voluto, e comunque a non assumere mai certi comportamenti indesiderati
- ▶ non è detto che si riesca ad ottenere esattamente  $Sp$  se le specifiche sono mal formulate o irrealizzabili (ovvero se chiedono all'impianto più di quanto può fare)

## Approcci possibili al progetto del controllore

### Approccio *diretto*:

- ▶ si progetta il controllore direttamente a partire dalle specifiche
- ▶ si costruisce direttamente un modello formale (con proprietà fondamentali verificabili) del comportamento desiderato del sistema, opportunamente vincolato e ristretto rispetto al sistema da controllare
- ▶ il controllore si ricava poi facilmente dal modello delle specifiche, con poche e semplici modifiche concettuali

### Approccio *indiretto*:

- ▶ il controllore è il risultato di un processo di *sintesi*, a partire dal modello dell'impianto  $P$  e dal modello delle specifiche  $S_p$
- ▶ l'approccio indiretto è simile al paradigma classico del progetto di un sistema di controllo: si descrive il modello di un sistema da controllare e poi si progetta un controllore che ne vincoli il comportamento, in modo che siano soddisfatte le specifiche

## Approccio diretto

L'approccio diretto prevede che si costruisca *direttamente* un modello delle specifiche (senza passare attraverso un modello del sistema da controllare).

Si passa direttamente da  $Sp$  a  $C$ .

Procedimento tipico:

- ❶ specifiche espresse in modo informale (p.es. a parole in linguaggio naturale)
- ❷ costruzione di un modello formale del comportamento desiderato dell'intero sistema
- ❸ costruzione del controllore a partire dal modello del comportamento desiderato, con poche modifiche concettuali ( $C = Sp + I/O$ )

La stesura delle specifiche è un problema complesso, per due motivi:

- ▶ occorre definire nel dettaglio tutti i funzionamenti ammissibili del sistema, definendo le “ricette” di produzione e specificando sequenze di lavorazione, tipi di prodotti, macchine necessarie per i vari prodotti, ecc.
- ▶ il modello delle specifiche deve essere definito con metodi formali, in modo da assicurare determinate proprietà strutturali al termine del progetto

## Osservazioni:

- ▶ per la costruzione del modello delle specifiche si possono utilizzare varie tecniche di manipolazione dei modelli (v. metodi *top-down*, *bottom-up* e *ibridi*)
- ▶ il passo ③ è relativamente semplice:
  - ▼ il modello  $Sp$  contiene la concatenazione di eventi che devono accadere nel sistema  $P$
  - ▼ il controllore, in più, deve distinguere tra
    - gli eventi che *arrivano* al controllore dall'impianto (eventi incontrollabili o *misure*), per i quali il controllore deve porsi “in attesa/ascolto” (il controllore non può forzare una misura ad accadere), e
    - gli eventi che *vanno* dal controllore all'impianto (eventi controllabili o *comandi*), che devono essere emessi non appena possibile.
- ▶ non viene utilizzato un modello dell'impianto, per stabilire se una specifica sia realizzabile (ciò può essere anche motivato dal fatto che il modello potrebbe dare luogo ad un numero di stati molto più grande del modello delle specifiche)
- ▶ la bontà del controllore finale può essere giudicata solo in base a modelli informali del comportamento dell'impianto o ad eventuali modelli di simulazione realizzati *ad hoc*

## Approccio indiretto

Il controllore è il risultato di un processo formale di sintesi, a partire dal modello delle specifiche  $Sp$  e anche dal modello dell'impianto  $P$ .

Come tale, il controllore può avere delle proprietà in più rispetto ad un controllore progettato con un metodo diretto:

- ▶ per costruzione, è garantita la compatibilità del comportamento in anello chiuso con le specifiche (tipicamente espresse sotto forma di vincoli)
- ▶ si possono garantire le proprietà fondamentali del sistema complessivo

Teoria del *controllo supervisivo*:

- ① costruzione del modello  $P$  dell'impianto
- ② costruzione del modello  $Sp$  delle specifiche
- ③ sintesi del controllore  $C$

## Osservazioni:

- ▶ rispetto alla stesura delle specifiche, la modellizzazione dell'impianto non controllato è un problema più semplice
  - ▼ per esempio, un modello non controllato di un impianto manifatturiero può essere una collezione di sottomodelli indipendenti, ciascuno dedicato a descrivere gli stati di ogni singolo dispositivo come se fosse indipendente
  - ▼ dal punto di vista del numero degli stati, generalmente è un modello molto più grande di quello del comportamento desiderato
- ▶ il modello del comportamento desiderato, oltre a questo, specifica le sequenze di operazioni, la condivisione di risorse, ecc.