

# Reti di Petri, LD e SFC

1

## Reti di Petri, LD e SFC - introduzione -

Abbiamo più volte detto che è sensato

- esprimere le specifiche di un problema di controllo logico in parte come vincoli e in parte come sequenze desiderate di eventi,
- progettare il controllo logico usando un modello logico formale (una rete di Petri o anche un automa, etc.) e con uso coordinato dei metodi studiati (indiretti, diretti top-down e bottom-up),
- trattare nel modello il tempo "inserendolo nella definizione delle attività",
- fare, se del caso, le analisi e le verifiche sul modello del sistema controllato,
- implementare il controllore con uno o più linguaggi IEC1131-3, usando tendenzialmente SFC per il controllo (sequenze) e LD per la supervisione (vincoli).

E' ora il momento di vedere come, da un unico modello logico del controllo (in senso lato), si ottengono le parti di supervisione e controllo (in senso stretto) e le si implementa con un uso coordinato di LD e SFC.

Come in altri casi, si tratta in sostanza d'imparare una tecnica del tutto standard (del resto, se così non fosse, non staremmo implementando ma ancora progettando).

Tuttavia a noi interessa in primis l'aspetto concettuale della questione, per cui non entriamo in tutti i dettagli.

2

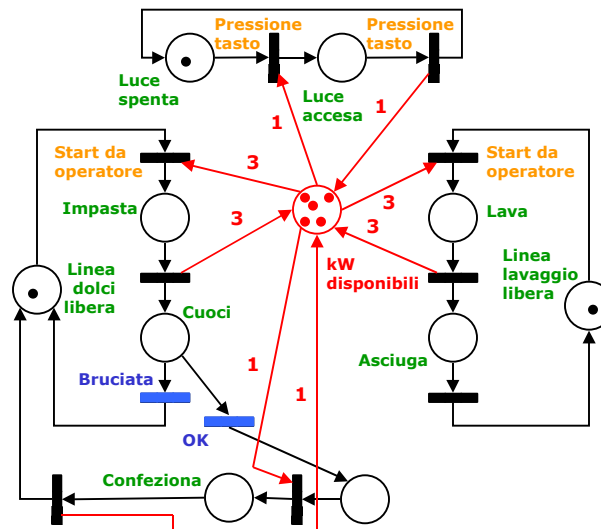
## Reti di Petri, LD e SFC - posizione del problema -

Appoggeremo la spiegazione su di un esempio.

Consideriamo un sistema di controllo logico semplice ma con tutti gli elementi di nostro interesse, ovvero

- **sequenze,**
- **stati (attività),**
- **eventi, di cui alcuni incontrollabili,**
- **vincoli (risorse condivise).**

Naturalmente, si ribadisce, usiamo questo esempio soltanto per spiegare come mettere insieme reti di Petri, LD e SFC per realizzare la coppia supervisione/controllo (in senso stretto): i dettagli "fisici" del processo qui non c'interessano.



3

## Reti di Petri, LD e SFC - posizione del problema -

**Ruolo delle reti di Petri come strumento di specifica del codice.**

Già sappiamo che le reti di Petri sono il formalismo per scrivere ed analizzare il modello del sistema di controllo. Qui non richiamiamo tutti i dettagli, ma facciamo alcune osservazioni concettuali, che partendo dal mondo dei soli modelli conducono a capire come le reti di Petri specificano anche il codice di controllo.

- La rete potrebbe esser il prodotto della fusione di tre sottoreti (linea dolci, linea lavaggio e luci) "messe insieme" dal controllo superviso che impone il vincolo sulla potenza disponibile, e potrebbe essere successivamente complicata affinando le attività nelle loro sequenze operative di dettaglio. *Si dovrebbe capire come in generale una rete siffatta è dunque il risultato dell'uso coordinato di tutti i metodi di progetto visti.*
- La rete potrebbe essere analizzata per vedere se può bloccarsi, se le risorse e i conflitti sono rappresentati correttamente, e così via. Qui non lo facciamo per brevità ma si vede subito che tutto funziona correttamente: la rete non si blocca e quando ci sono conflitti effettivi il supervisore può risolverli. Non sempre ha totale libertà, però: ad esempio, se non si sta impastando il supervisore è libero di scegliere se consentire l'accensione della luce o il lavaggio, se si sta impastando no. Infatti la rete non è a scelta libera, e non lo è per via dell'esistenza di un vincolo sulla potenza e del valore della massima potenza disponibile. *Si dovrebbe ulteriormente apprezzare il ruolo dell'analisi, che va fatta sul modello.*

4

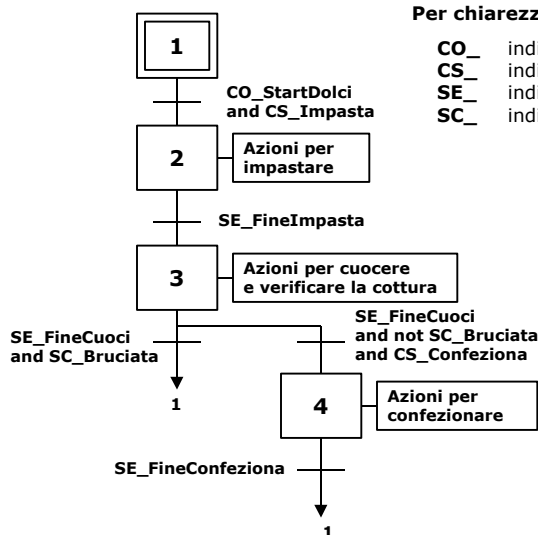
## Reti di Petri, LD e SFC - posizione del problema -

- Il supervisore vede un impianto esteso, nel senso che per realizzare il supervisore non serve affinare tutte le attività, *basta che si vedano gli eventi che esso deve osservare e/o controllare*.
- Gli affinamenti di massimo dettaglio (quelli finali) si possono anche fare direttamente sul codice SFC (già sappiamo che le sequenze si gestiscono in quella sede, *ed ecco che compare il codice*).
- Il supervisore dovrà risolvere i conflitti dando o negando i consensi all'inizio delle attività (e questo si farà in LD).
- Risolvere i conflitti significa, a livello d'implementazione, rendere o meno superabili certe transizioni (abilitate) del codice SFC. *Nel contesto del codice si possono anche rendere superabili tutte le transizioni abilitate* (ad esempio, quando è concesso dal vincolo, si può consentire insieme l'inizio del lavaggio e l'accensione della luce). Questo differenzia il codice del PLC (sincrono) dal suo modello (asincrono), nel senso che il primo non è il secondo ma bensì una sua implementazione. Tuttavia, quando si è discusso dell'evoluzione delle reti di Petri, si è visto che lo scatto contemporaneo di tutte le transizioni abilitate di una rete non è in contraddizione con la scelta casuale purché esse non siano in conflitto (ovvero purché i conflitti siano correttamente risolti, come dev'essere - e sul modello lo si verifica - nel sistema controllato).

5

## Reti di Petri, LD e SFC - codice SFC per il controllo -

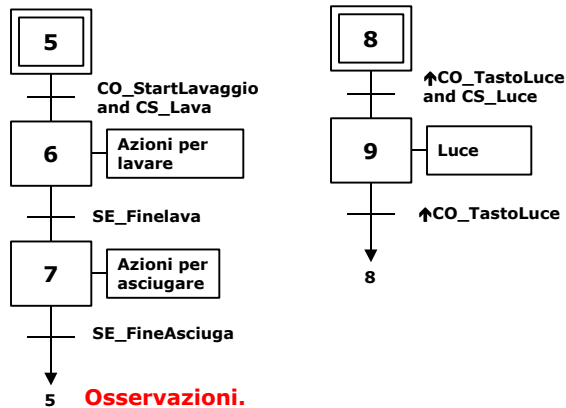
### Controllo della linea dolci.



6

## Reti di Petri, LD e SFC - codice SFC per il controllo -

### Controllo della linea lavaggio e della luce.



#### Osservazioni.

I tre SFC del controllo corrispondono alle tre ipotetiche sottoreti unite dal supervisore a formare quella complessiva (il modello da analizzare).

Coerentemente, in questi tre SFC il supervisore non si vede ma si deve vedere la connessione con esso (eventi segnalati e consensi ricevuti). 7

## Reti di Petri, LD e SFC - codice LD per la supervisione -

Ricordiamo preliminarmente che, per eseguire il programma SFC ad ogni ciclo del PLC quest'ultimo "esplora" lo schema SFC e fa quanto segue:

- esegue le azioni corrispondenti alle fasi attive,
- determina le transizioni superabili,
- aggiorna lo stato, ovvero l'insieme delle fasi attive.

Ricordiamo anche che il PLC opera per copia massiva, e quindi l'effetto di quanto si è calcolato in un ciclo del PLC si trasmette alle uscite soltanto alla fine del ciclo stesso, mentre gli ingressi si leggono una sola volta all'inizio del ciclo.

Ciò premesso, il codice LD del supervisore (che gira appunto una volta ad ogni ciclo del PLC, e noi supporremo preceda l'esplorazione del codice SFC) dovrà

- aggiornare il suo stato (ovvero la marcatura dei posti di controllo) in base agli eventi osservati (che temporalmente sono avvenuti nel ciclo prima),
- determinare quali sono i consensi da dare o negare in base allo stato suo e del resto del sistema (controllo e impianto),
- decidere quali consensi dare risolvendo gli eventuali conflitti in essere,
- aggiornare nuovamente il suo stato in base ai consensi dati (e che produrranno effetto nella corrente esplorazione del codice SFC, stanti le regole di evoluzione di quest'ultimo).

8

## Reti di Petri, LD e SFC - codice LD per la supervisione -

### Applichiamo quanto detto all'esempio.

Per comodità, prima di scrivere il LD vero e proprio, lo esprimeremo come pseudocodice (se si scrivesse il supervisore in ST, cosa possibile, questo sarebbe praticamente già il codice).

1. Aggiornamento dello stato del supervisore  
in funzione degli eventi osservati (la  
variabile kWdisp è la marcatura del posto  
di controllo e si assume inizializzata a 5).

```
if SE_FineImpasta
    kWdisp = kWdisp+3
endif
if SE_FineConfeziona
    kWdisp = kWdisp+1
endif
if SE_FineLava
    kWdisp = kWdisp+3
endif
if X9 and ^CO_TastoLuce
    kWdisp = kWdisp+1
endif
```

2. Determinazione dei consensi richiesti in funzione  
dello stato complessivo (RC\_ è il prefisso che  
useremo per le richieste di consenso).

```
RC_Impasta    = X1 and CO_StartDolci
RC_Confeziona = X3 and SE_FineCuoci and not SC_Bruciata
RC_Lava       = X5 and CO_StartLavaggio
RC_Luce       = X8 and ^CO_TastoLuce
```

9

## Reti di Petri, LD e SFC - codice LD per la supervisione -

3. Determinazione dei consensi da dare con risoluzione dei conflitti e  
aggiornamento dello stato del supervisore, assumendo che i consensi dati  
producano effetto nel ciclo PLC in corso (qui usiamo un meccanismo di priorità  
banale e lo implementiamo con l'ordine delle istruzioni, come poi si fa in LD)

```
CS_Impasta    = 0
CS_Confeziona = 0
CS_Lava       = 0
CS_Luce       = 0

if RC_Impasta and kWdisp>=3
    CS_Impasta = 1
    kWdisp = kWdisp-3
endif
if RC_Lava and kWdisp>=3
    CS_Lava = 1
    kWdisp = kWdisp-3
endif
if RC_Confeziona and kWdisp>=1
    CS_Confeziona = 1
    kWdisp = kWdisp-1
endif
if RC_Luce and kWdisp>=1
    CS_Luce = 1
    kWdisp = kWdisp-1
endif
```

Da questo (pseudo)codice  
ora otterremo il programma LD

10

## Reti di Petri, LD e SFC - codice LD per la supervisione -

SE_FineImpasta	+	TRUE	CS_Lava
ADD			( )
kWdisp			
3		TRUE	CS_Luce
kWdisp			( )
+	RC_Impasta	+	CS_Impasta
SE_FineConfeziona	+	GEQ	( )
ADD		kWdisp	
kWdisp		3	SUB
1			
kWdisp			
+	+	+	+
SE_FineLava	+	RC_Lava	CS_Lava
ADD			( )
kWdisp		GEQ	
3		kWdisp	
kWdisp		3	SUB
+	+	+	
X9 CO_TastoLuce	+	+	
ADD			
kWdisp			
1			
kWdisp		RC_Confeziona	CS_Confeziona
+			( )
X1 CO_StartDolci	RC_Impasta		
		kWdisp	
		1	SUB
X3 SE_FineCuoci SC_Bruciata RC_Confeziona			
X5 CO_StartLavaggio	RC_Lava		
		RC_Luce	CS_Luce
X8 CO_TastoLuce	RC_Luce		( )
		GEQ	
		kWdisp	
		1	SUB
TRUE	CS_Impasta		
		kWdisp	
		3	
TRUE	CS_Confeziona		
		kWdisp	
		3	

### NOTE

Non si sono implementate inizializzazioni.

TRUE è una costante (dal significato ovvio).

11

## Reti di Petri, LD e SFC - considerazioni generali -

Astraiamo ora dall'esempio quanto vi è di generale.

Come osservare nel supervisore un evento generato dal processo o dal controllo:

Evento Ep, generato dal processo e segnalato dal sensore SE\_Ep

Posto del supervisore, la cui marcatura è rappresentata dalla variabile intera mPs

SE_Ep	+	ADD	
	mPs		
	p		
	mPs		
+	+	+	

Nel caso più generale, oppure

SE_Ep	mPs	
	( L )	

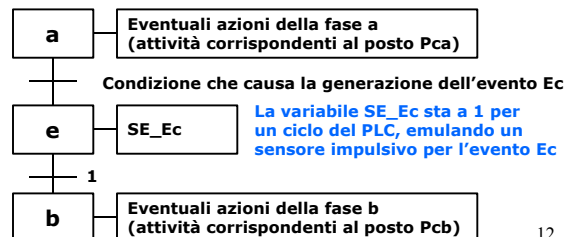
Se si è certi che la rete controllata è binaria (e ovviamente p=1)

Come generare un evento nel controllo (come li generi il processo è ovvio):

Posto Pca del controllore

Evento Ec, generato dal controllo e per il quale non c'è ovviamente alcun sensore

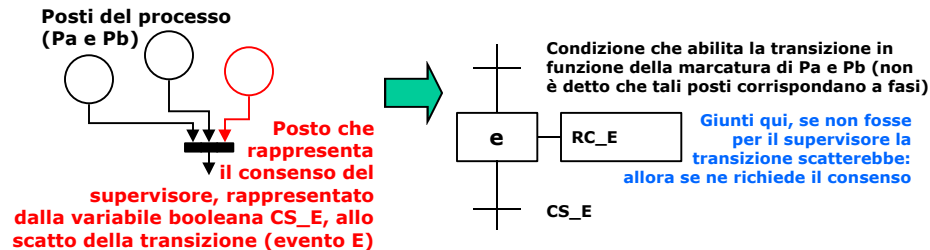
Posto Pcb del controllore



12

## Reti di Petri, LD e SFC - considerazioni generali -

Come subordinare il superamento di una transizione nel controllo ad un consenso (cioè alla marcatura e al meccanismo di risoluzione dei conflitti) del supervisore:



### Osservazioni.

In questi tre casi non ci si è soffermati sui pesi degli archi nelle reti di Petri, perché si è visto dall'esempio come essi si considerano nello scrivere il codice.

La generazione della richiesta di consenso nello schema SFC consente di ridurre la necessità di replicare le condizioni logiche nel controllo (SFC) e nel supervisore (LD), cosa su cui nell'esempio non ci si è diffusi per brevità.

### NOTA

Se **Pa** e **Pb** possono smarcarsi prima del consenso, occorre prevedere una transizione che disattivi la fase **e** (se il controllo è ben progettato ciò avviene di rado).

13

## Reti di Petri, LD e SFC - considerazioni generali -

### Osservazioni conclusive.

Non sempre è necessario usare il metodo nella sua completezza: a volte un supervisore semplice si può realizzare con poche variabili gestite dalle stesse fasi del controllo scritto in SFC.

Tuttavia, se si fa così è difficile operare con variabili del supervisore non booleane e soprattutto in presenza di potenziali conflitti da risolvere.

Il metodo proposto produce un codice LD non particolarmente leggibile e a volte causa l'aggiunta di fasi e transizioni al codice SFC (per le richieste di consenso).

Tuttavia il metodo è sistematico, consente di tradurre in codice un modello senza ambiguità, e consente anche di preservare la separazione tra supervisione e controllo (nell'esempio, se si vuol togliere il vincolo sulla potenza basta fissare a 1 i consensi nel codice LD e gli schemi SFC non cambiano).

Da ultimo, la struttura del codice prodotto con questo metodo non corrisponde soltanto al modello formale, ma anche alla struttura di programma adottata da diversi ambienti di sviluppo IEC1131-3 (ad esempio in IsAGRAF è naturale mettere il codice LD del supervisore nella sezione "begin" e il codice SFC del controllore nella sezione "sequential"). Questo aiuta a comprendere l'importanza sia modellistica che implementativa non soltanto del metodo proposto, ma di tutto l'approccio alla sintesi del controllo logico che ad esso ha condotto.

14

# Riepilogo generale del corso

15

## Riepilogo generale del corso - concetti fondamentali -

### **Problema**

Controllo logico e sue relazioni col controllo modulante per formare il tipico sistema di controllo industriale.

### **Formalismo descrittivo**

DES, con le loro caratteristiche distintive.

### **Formalismo modellistico**

Vari, tra cui abbiamo scelto di usare prevalentemente le reti di Petri.

### **Metodi per il progetto del controllo**

Indiretti e diretti, coerentemente alla specifica come mix di vincoli e sequenze (supervisione e controllo in senso stretto), da usarsi in modo coordinato.

### **Tecnologia per l'implementazione del controllo**

PLC e concetti connessi, IEC1131-3 con particolare riferimento a LD e SFC, relazioni tra modello del controllo e sua implementazione, metodi per implementazione e traduzione senza ambiguità di un modello di controllo.

16



## Riepilogo generale del corso - conclusioni -

L'obiettivo didattico che il corso persegue non è quello d'insegnare in dettaglio la tecnologia del controllo logico, che può essere appresa nella pratica professionale (qualunque uso si debba farne) ed è comunque in rapida evoluzione.

L'obiettivo è invece insegnare i principi sottostanti a tale tecnologia, portando l'allievo a comprendere che l'uso efficiente (e spesso la stessa comprensione) della tecnologia si ottiene soltanto a patto che nell'affrontare un problema di controllo logico si abbia un metodo di lavoro sistematico.

Lo scopo ultimo del corso, in fondo, è proprio fornire un metodo siffatto, e tale scopo si può ritenere aggiunto se l'allievo, oltre a conoscere concetti modellistici e metodi per operare sui modelli al fine di ottenerne il controllo, trae da tutto ciò la decisa convinzione che - a meno dei casi banali - di fronte ad un problema di controllo di un sistema ad eventi discreti è molto più produttivo affrontarlo in modo formale piuttosto che "provare a buttar giù una logica e poi metterla a posto man mano".