



## **Automazione industriale dispense del corso**

### **21. Implementazione di automi e reti di Petri**

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

## Problemi generali di implementazione

Come per l'SFC, anche per gli automi e le reti di Petri (modelli astratti, asincroni e, nel secondo caso anche paralleli) si pone il problema dell'implementazione su macchine generiche per l'elaborazione dell'informazione, che operano in modo sequenziale e sincrono, come il PLC.

- ▶ l'implementazione del controllore dovrà essere sufficientemente più rapida del sistema da controllare, in modo da poter reagire prontamente agli ingressi che arrivano da esso (non si devono perdere eventi)
- ▶ se dovessero scattare due eventi nello stesso tempo di ciclo, come si comporta il PLC? E' in grado di processarli entrambi?

In generale, i modelli devono essere congegnati in modo tale che:

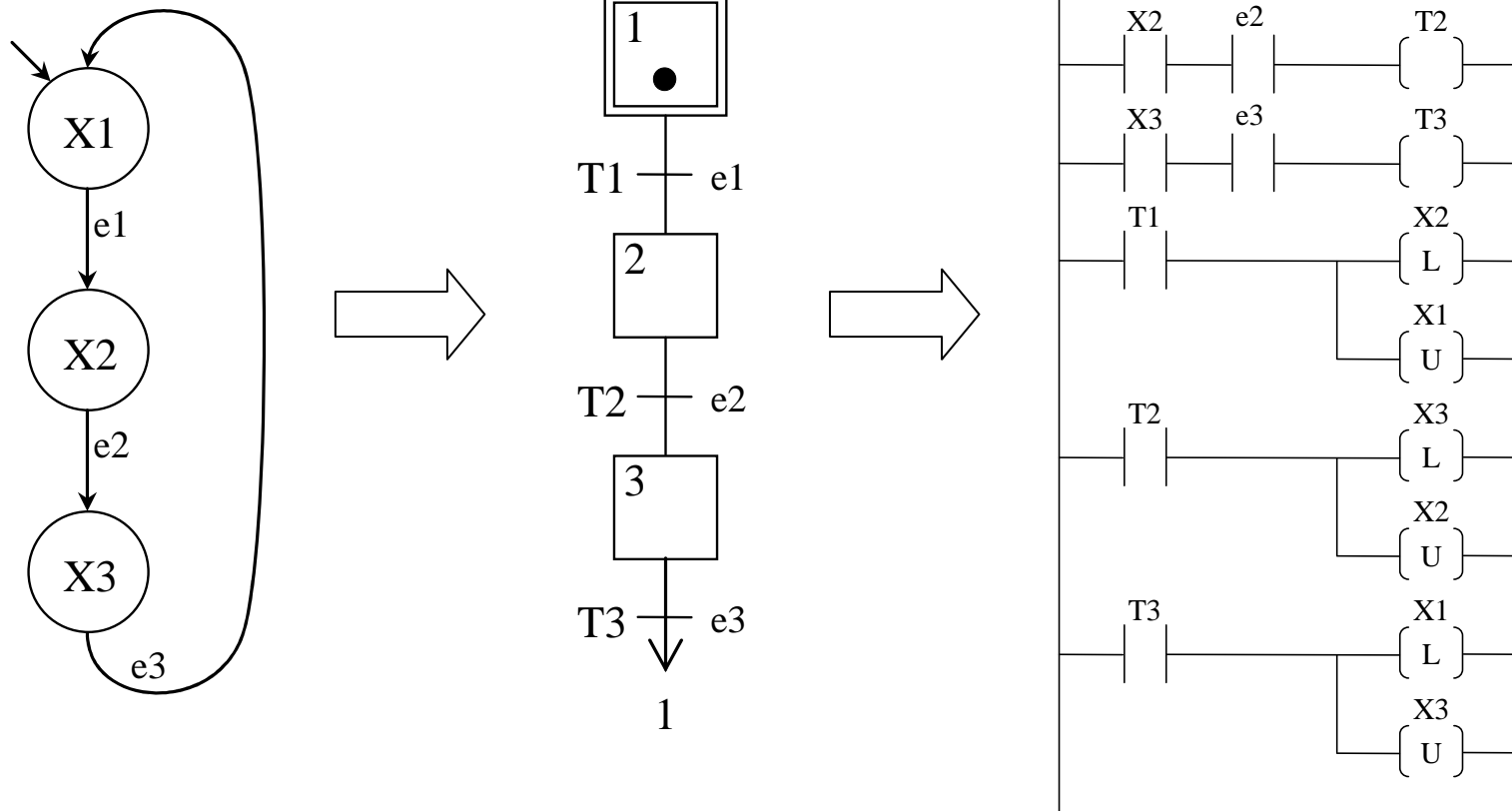
- ▶ ogni evento generato dal processo sia accettabile dal PLC, quando accade (la transizione di stato corrispondente deve poter avvenire al momento giusto)
- ▶ se accadono più eventi in un tempo di ciclo devono essere tutti accettabili dal PLC e processabili correttamente nello stesso tempo di ciclo

Rispetto agli schemi SFC, gli automi e le reti di Petri sono modelli più astratti (sono pensati per un “interprete ideale”, in grado di farli evolvere in modo infinitamente rapido, quando richiesto) e quindi la traduzione in linguaggi di basso livello, come il LD, è meno immediata:

- ▶ gli automi e le reti di Petri sono in generale modelli non deterministici
  - ▼ in un automa, uno stato può avere più nodi successori, di modo che sono possibili diverse transizioni di stato
  - ▼ in una rete di Petri, più transizioni possono essere abilitate in uno stato
  - ▼ al contrario, un controllore deve reagire in modo deterministico
  - ▼ la traduzione di un automa o una rete di Petri dovrà quindi essere compatibile con una delle possibili evoluzioni del modello ed evitare di “tralasciare” qualche transizione di stato
- ▶ il tempo non è modellizzato
  - ▼ non sappiamo *quando* avviene una transizione di stato
  - ▼ un controllore deve invece reagire in tempi prefissati

## Implementazione di automi

Il modo più semplice di tradurre un automa è quello di riformularlo come SFC, e poi (se necessario) tradurre quest'ultimo in LD.



## Implementazione di reti di Petri

In primo luogo, occorre estendere la rete di Petri, associandovi ingressi e uscite:

- ▶ tipicamente, ingressi e uscite sono eventi associati alle transizioni (v. modello a 2 eventi)
- ▶ ciò non esclude però altre soluzioni, come l’associazione di segnali di comando ai posti e delle sole misure alle transizioni (v. SFC)

L’implementazione della rete dovrà quindi:

- ▶ accettare le misure dell’impianto controllato
  - ▼ quando l’impianto genera un evento, il controllore deve essere pronto ad accettarlo
- ▶ emettere i comandi prima possibile
  - ▼ se dal modello risulta che un comando può essere emesso, non ci sono motivi per dilazionarlo

## Algoritmo di evoluzione

Anche qui si deve utilizzare un algoritmo di evoluzione che interpreti correttamente le regole di evoluzione della rete di Petri.

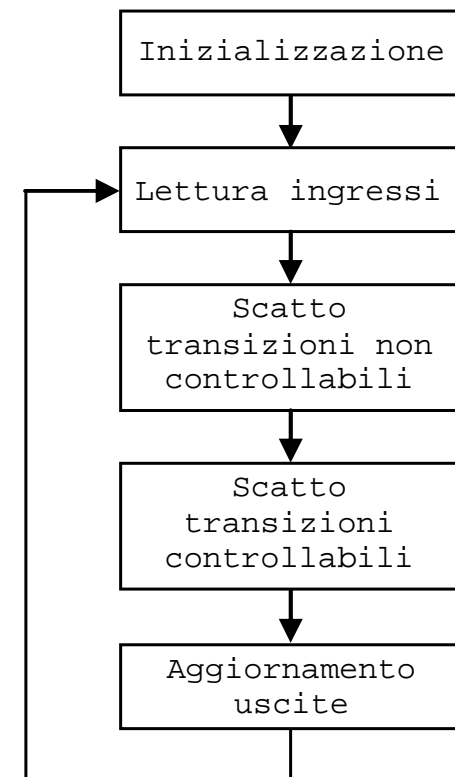
L'idea base è che *prima* il controllore accetta informazioni dall'impianto e *quindi*, sulla base del nuovo stato, prende decisioni aggiornate.

L'algoritmo di evoluzione è diviso in due porzioni:

- ❶ accettazione degli eventi non controllabili (*misure*)
- ❷ produzione degli eventi controllabili (*comandi*)

Ad ogni ciclo vengono compiuti i seguenti quattro passi:

- ❶ lettura ingressi
- ❷ determinazione e scatto delle transizioni non controllabili
- ❸ determinazione e scatto delle transizioni controllabili
- ❹ aggiornamento segnali di uscita



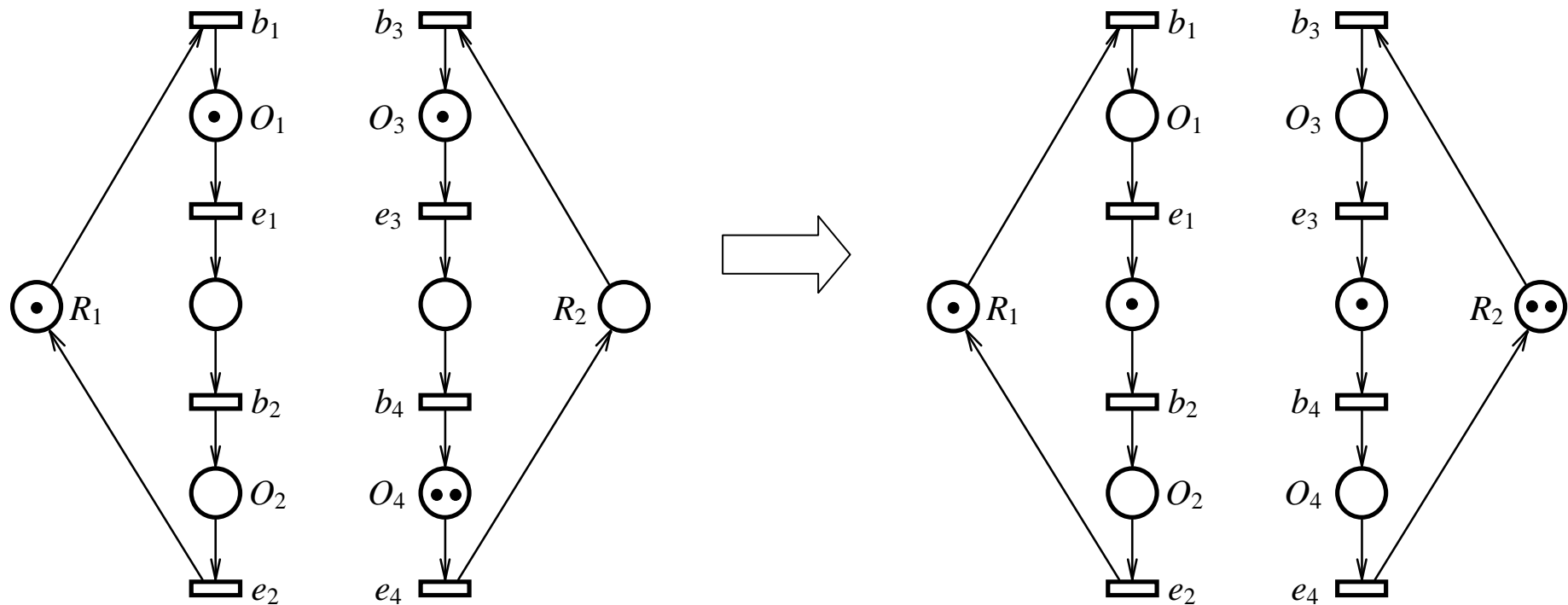
## Osservazioni:

- ▶ cosa accade se ci sono più transizioni abilitate nelle porzioni non controllabili o controllabili?
- ▶ quale ordine di scatto va imposto?

Faremo l'ipotesi che non esista un ordine di scatto migliore di un altro.

Più precisamente, assumeremo che lo stato assunto dalla rete di Petri sia indipendente dall'ordine con cui essa esegue una sequenza di eventi non controllabili.

## Esempio 1



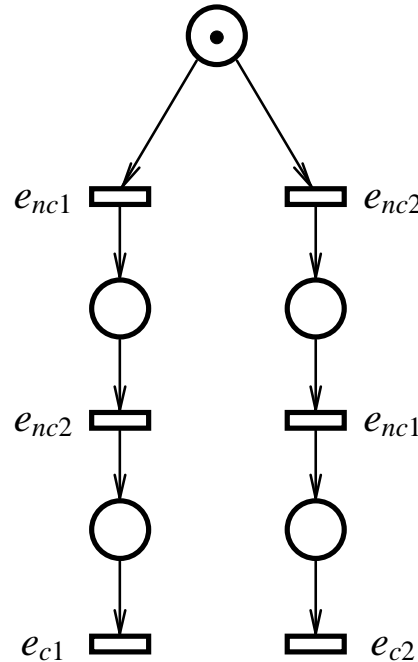
Nello stato rappresentato nella figura a sinistra, il sistema è in grado di accettare gli eventi non controllabili  $e_1$ ,  $e_3$  ed  $e_4$  (2 volte), rispettivamente associati alle terminazioni delle operazioni  $O_1$ ,  $O_3$  ed  $O_4$ .

Qualunque sia l'ordine di scatto delle transizioni associate, lo stato finale sarà sempre lo stesso (figura a destra).



## Esempio 2

Un esempio di rete che non soddisfa l’ipotesi è il seguente:

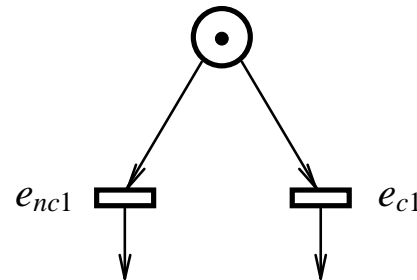


Gli eventi  $e_{nc1}$  e  $e_{nc2}$  sono non controllabili, mentre  $e_{c1}$  e  $e_{c2}$  sono controllabili.

Se  $e_{nc1}$  e  $e_{nc2}$  sono letti nello stesso tempo di ciclo, l’ordine con cui vengono interpretati dipende dal codice e a seconda dell’implementazione scelta verrà emesso  $e_{c1}$  oppure  $e_{c2}$ .

## Esempio 3

Un'altra situazione non accettabile è quella riportata in figura:



In effetti, questa è una struttura ambigua:

- ▶ occorre aspettare l'emissione di  $e_{nc1}$  oppure si deve emettere  $e_{c1}$ ?
- ▶ e se si stabilisce di attendere  $e_{nc1}$ , per quanto tempo deve durare l'attesa?

Faremo perciò anche una seconda ipotesi, e cioè che non esistano conflitti tra eventi controllabili e eventi non controllabili.

## Traduzione di reti di Petri in LD

Per semplicità supporremo inizialmente che la rete sia binaria.

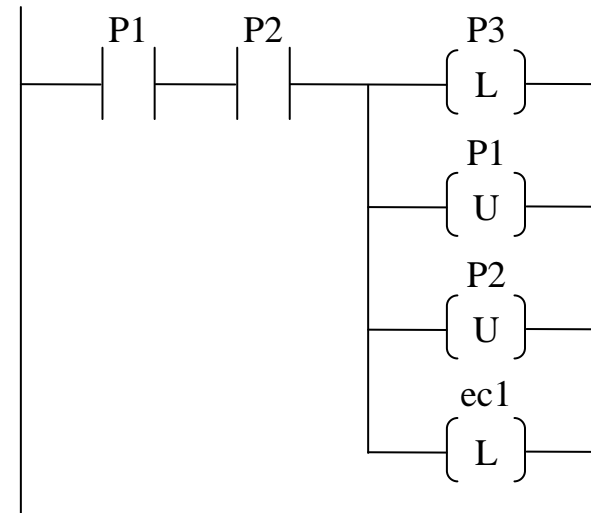
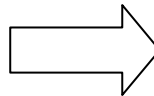
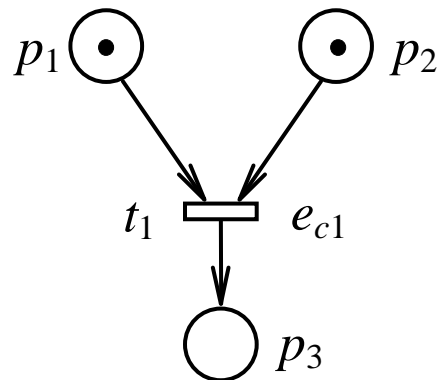
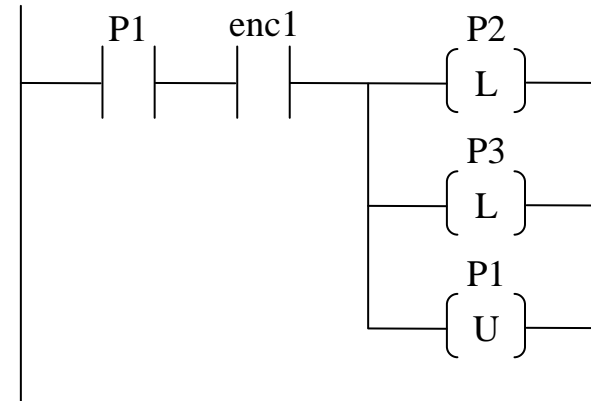
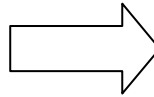
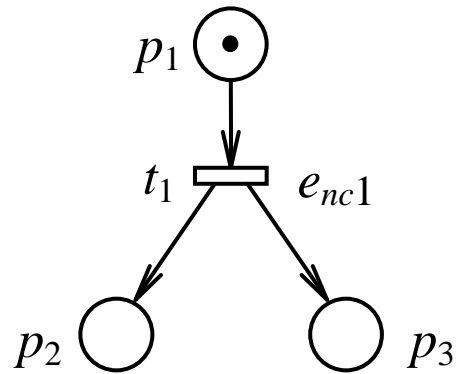
- ▶ altrimenti bisogna introdurre opportuni contatori per gestire le marcature dei posti e operatori di confronto per valutare l'ammissibilità di transizioni

Associazioni:

- ▶ ad ogni posto si associa un bit di memoria che ne rappresenta lo *stato*
- ▶ ad ogni transizione si associa un'istruzione (un rung) che ne rappresenta lo *scatto*
- ▶ gli eventi controllabili sono associati a bobine
- ▶ gli eventi non controllabili sono associati a contatti

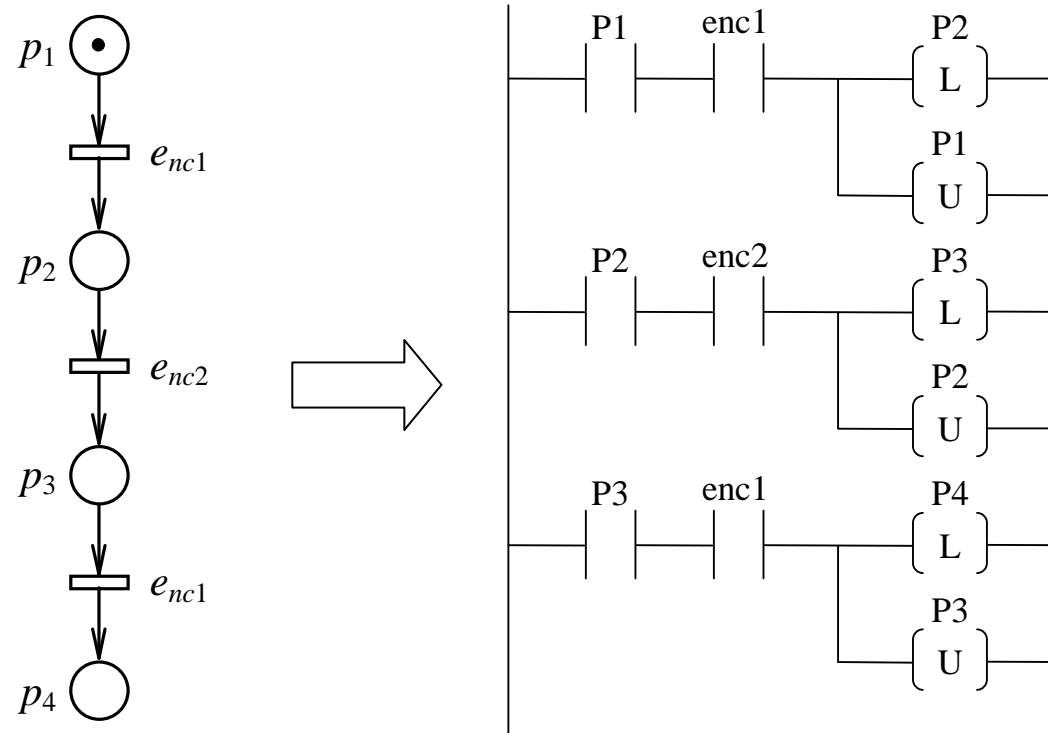
Entrambi i tipi di eventi sono modellizzati come impulsi di durata pari ad un tempo di ciclo.

## Idea base:



Supponiamo di avere una rete di Petri contenente una sequenza di transizioni tra cui alcune associate allo stesso evento non controllabile.

Se accade l'evento non controllabile e il programma LD lo consuma più volte, come se ne fossero accaduti più di uno contemporaneamente (nello stesso tempo di ciclo), allora si dice che ha avuto luogo l'*effetto valanga*, che rappresenta un comportamento scorretto, non coerente con la rete originaria.



Nell'esempio, per marcare  $p_4$  devono accadere gli eventi  $e_{nc1}$ ,  $e_{nc2}$  ed  $e_{nc1}$ , esattamente in quest'ordine.

Invece, se si verificassero solo  $e_{nc1}$  ed  $e_{nc2}$  si dovrebbe raggiungere la marcatura in cui c'è un gettone in  $p_3$ .

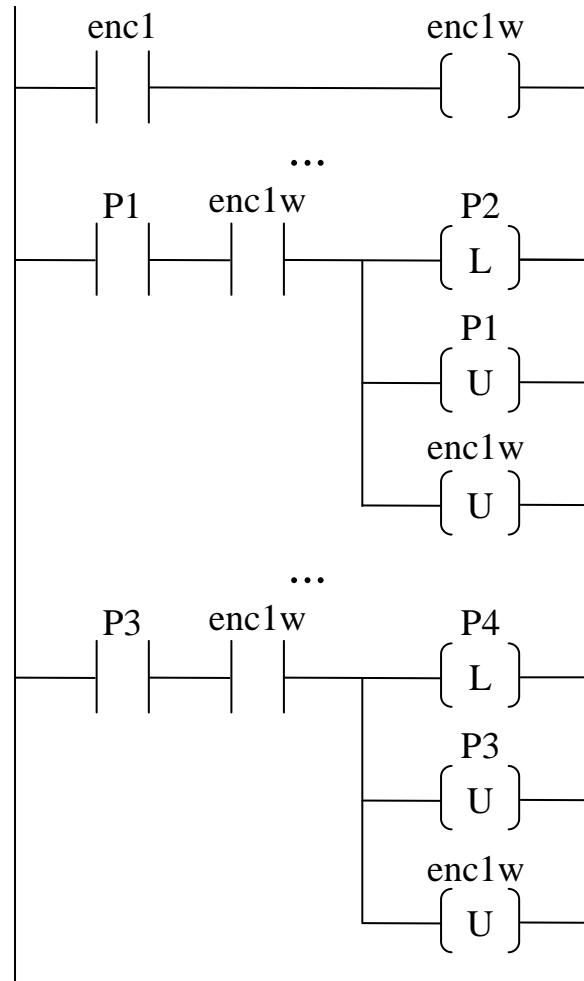
Se si implementa la rete come nello schema LD a destra e gli eventi  $e_{nc1}$  ed  $e_{nc2}$  capitano nello stesso tempo di ciclo, con  $P1 = 1$ , c'è continuità logica in tutti e 3 i rung e alla fine il bit associato a  $p_4$  risulta settato a 1.

L'evento  $e_{nc1}$  è stato “consumato” 2 volte!

Rimedi:

- ▶ cambiare l'ordine dei pioli (per esempio invertire l'ordine rispetto alla sequenza della rete di Petri); non è detto che esista sempre un ordine che risolva tutti i possibili problemi di questa natura
- ▶ copiare l'ingresso associato ad un possibile effetto valanga in una variabile temporanea, da usare in luogo dell'ingresso vero, così da poterla resettare appena dopo averla consumata (in modo che possa essere consumata una volta sola)

## Implementazione con le variabili temporanee



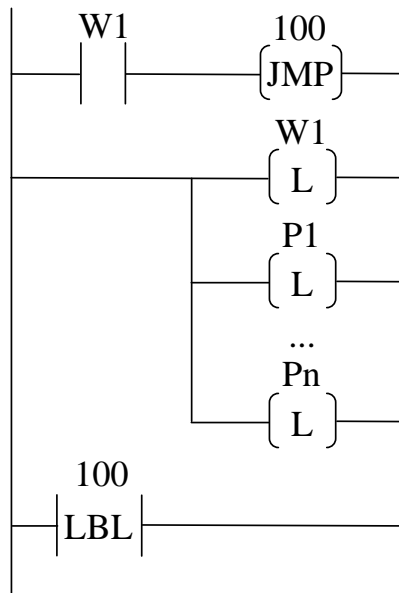
## Algoritmo generale di traduzione

Il codice LD sarà costituito dalle seguenti porzioni consecutive.

- ❶ inizializzazione
- ❷ creazione delle copie degli eventi soggetti all'effetto valanga
- ❸ accettazione eventi non controllabili
- ❹ reset delle variabili di comando
- ❺ generazione delle variabili di comando

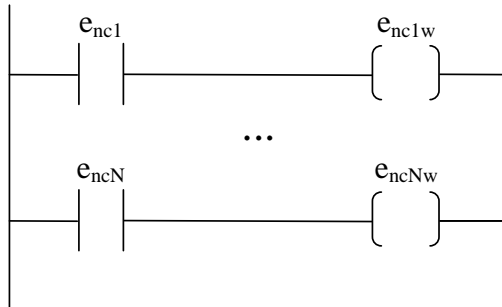


## ① inizializzazione

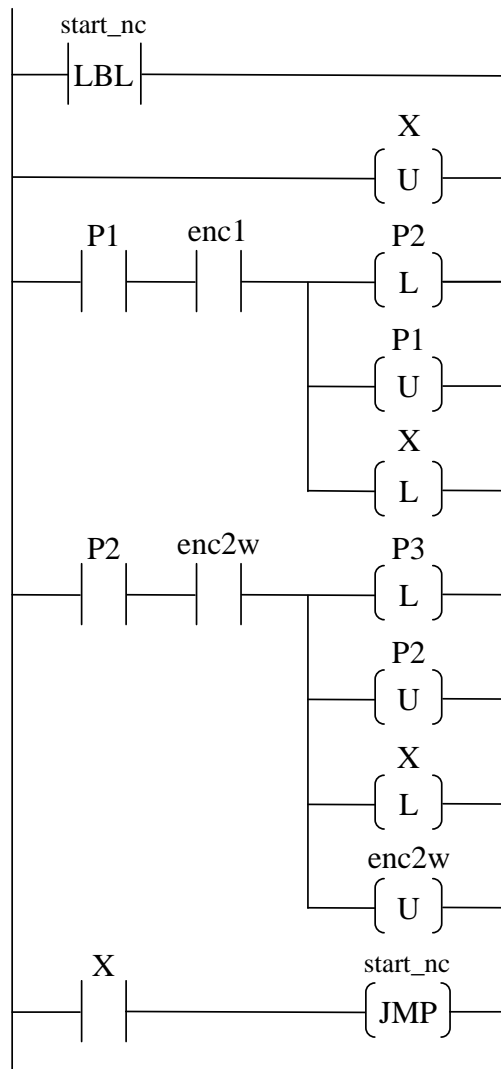


Vengono settati a 1 i bit associati ai posti inizialmente marcati.

## ② creazione delle copie degli eventi soggetti all'effetto valanga



### ③ accettazione eventi non controllabili



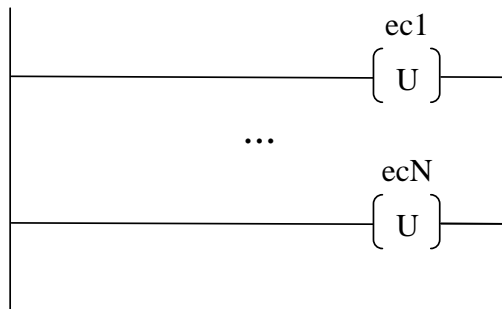
La variabile temporanea X registra l'eventuale scatto di una transizione associata ad un evento non controllabile (ogni piolo che rappresenta una transizione setta  $X = 1$ ).

Se alla fine delle istruzioni della sezione  $X = 1$ , si riesegue l'intera sezione.

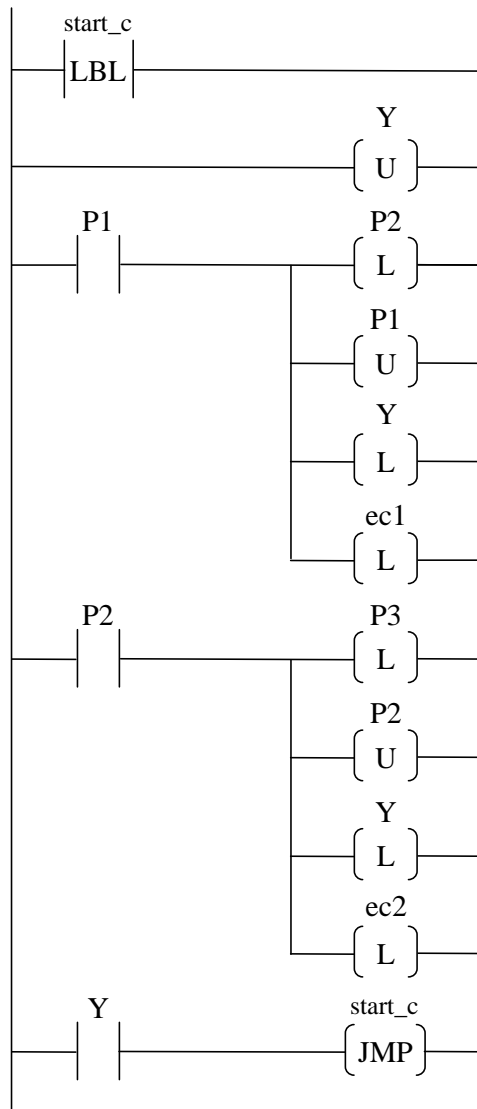
L'algoritmo continua a consumare eventi non controllabili finché ce ne sono.

In questo modo, se ci sono due transizioni in sequenza associate a due eventi non controllabili che capitano entrambi nello stesso tempo di ciclo, scattano entrambe, indipendentemente dall'ordine in cui sono eseguiti i due rung corrispondenti.

#### ④ reset delle variabili di comando



## 5 generazione delle variabili di comando



Anche in questo caso, si usa una variabile temporanea Y per registrare l'eventuale scatto di una transizione associata ad un evento controllabile (ogni piolo che rappresenta una transizione setta  $Y = 1$ ).

Se alla fine delle istruzioni della sezione  $Y = 1$ , si riesegue l'intera sezione.

L'algoritmo continua a generare eventi controllabili finché è possibile.

Agli eventi controllabili sono associate bobine a ritenzione, altrimenti se lo stesso evento comparisse in due rung di cui uno attivo e uno no, la sua effettiva esecuzione dipenderebbe dall'ordine relativo dei due rung!

Usando bobine a ritenzione, il rung inattivo non ha alcun effetto sulla bobina associata all'evento, ed esso viene sicuramente eseguito per effetto del rung attivo.

## Osservazioni

La rete di Petri comunica con l'esterno attraverso eventi, mentre il sistema controllato trasmette e riceve segnali.

Occorre quindi un'interfaccia per consentire lo scambio di informazioni tra i due elementi.

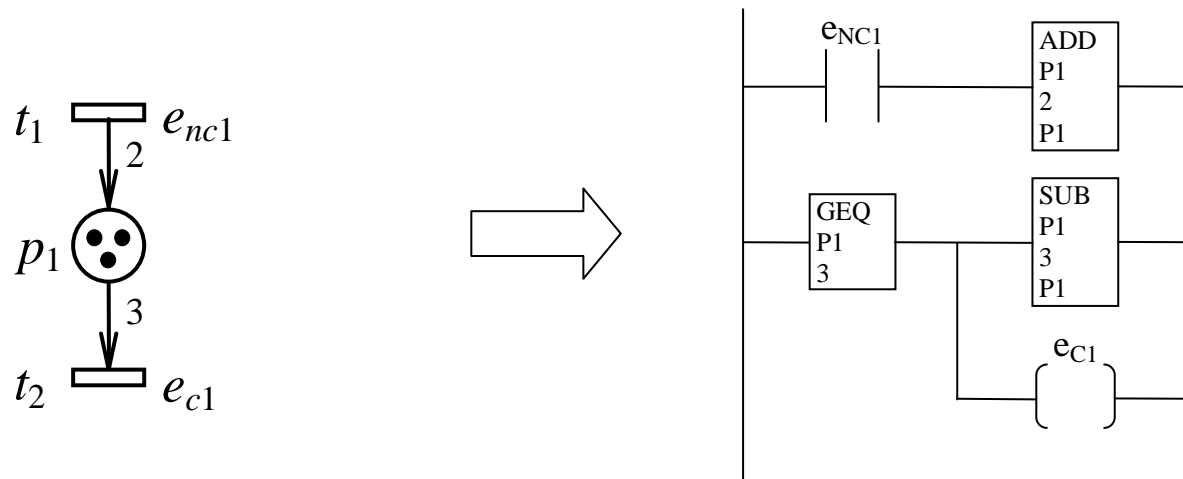
- ▶ con il metodo illustrato vengono generati impulsi di durata pari ad un tempo di ciclo per gli eventi controllabili
  - ▼ occorrerà allora un'ulteriore sezione del codice LD che trasformi tali impulsi in segnali adatti a pilotare effettivamente gli attuatori
- ▶ analogamente, anche gli eventi non controllabili sono impulsi di durata pari ad un tempo di ciclo
  - ▼ occorrerà allora un'ulteriore sezione del codice LD che rilevi i fronti di salita/discesa sui segnali di ingresso

L'effetto valanga non si presenta se tutti gli eventi non controllabili sono distinti tra loro, o se sono alternati ad eventi controllabili come accade seguendo il paradigma di rappresentazione delle attività con modelli a 2 eventi.

## Traduzione di reti di Petri non binarie

Se la rete non è binaria, occorre implementare la marcatura di un posto con dei contatori e le condizioni di scatto con comparatori (marcatura  $\geq$  peso arco).

L'idea generale è la seguente:



Le transizioni temporizzate si implementano con semplici timer.