Capitolo 5

Controllo logico

Per approfondimenti su questa sezione è possibile fare riferimento al libro:
Pasquale Chiacchio ***PLC e automazione industriale*** ed. McGraw Hill

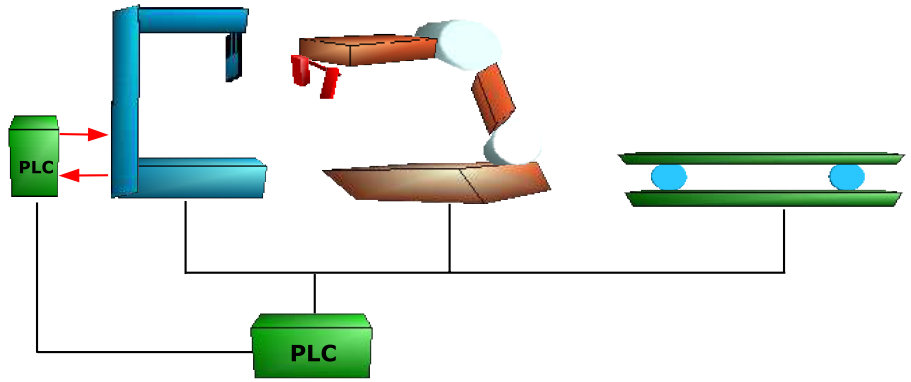

Figura 5.1: Schema di un impiano industriale

Sono rappresentati in figura 5.1 un generico impianto industriale e in figura 5.2 i livelli gerarchici dell'architettura di controllo di un sistema di automazione; il mezzo di collegamento tra i vari componenti va scelto in base alle prestazioni richieste al sistema; in generale una soluzione con bus di campo è meno costosa rispetto a collegamenti con cavi multipli in parallelo, ma più lenta.

Si richiamano alcune definizioni di base:

* **Attuatori**: attuano un comando trasformandolo in un'azione di controllo;
* **Controllo numerico**: sulla base di coordinate spaziali fornite da sensori applica un'azione di controllo per far seguire una traiettoria calcolata;
* **Controllo logico**: acquisisce variabili booleane, si addice a sistemi caratterizzati da eventi discreti.

Per poter realizzare un processo le macchine devono essere tra loro coordinate: questa **funzione di integrazione** è svolta dai **PLC** (*programmable logic controller*).

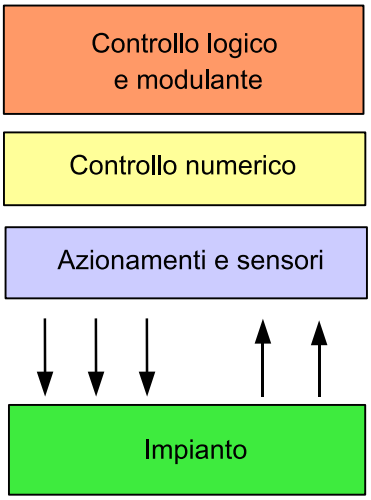

Figura 5.2: Livelli gerarchici dell'architettura di controllo in un sistema di automazione

Macchine utensili e robot complessi hanno in dotazione un PLC a bordo macchina, che acquisisce da essi variabili logiche e fornisce azioni di controllo; in tali casi il PLC d'impianto si interfaccia ai PLC-macchina.

In figura 5.4 è riportata la struttura di un PLC, composto da più ***rack*** entro i quali sono disposti vari moduli; in particolare:

- i moduli di Input/Output possono essere di tipo analogico o digitale;
- nella CPU vengono implementati gli algoritmi di controllo;
- HMI è il modulo *human machine interface*.

Si può prevedere su un singolo modulo una struttura ridondante con la presenza di 3 CPU: se una di esse fornisce dati non consistenti viene considerata in stato di errore; qualora questo avvenga anche per una delle CPU restanti, l'intero modulo va in emergenza. Questa struttura viene triplicata su 3 moduli, con la possibilità di avere quindi 9 processori che operano in modo parallelo.

PLC	I/O	MEMORIA
Micro	64	1-2 kbyte
Piccoli	512	4 kbyte
Medi	2048	~10^1 kbyte
Grandi	~10^3	~10^2 kbyte

Figura 5.3: Caratteristiche di PLC con diverse dimensioni

Si rappresenta in figura 5.3 una schematizzazione delle caratteristiche di dispositivi PLC di diverse dimensioni

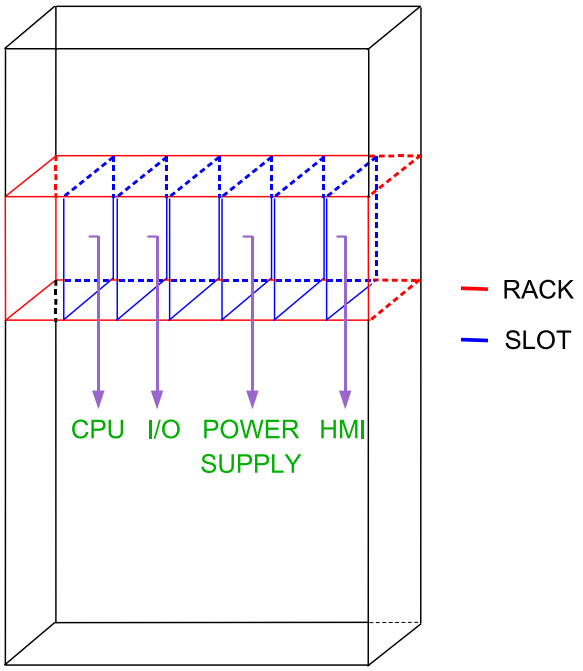

Figura 5.4: Struttura di un PLC

Riguardo alla programmazione di dispositivi PLC, si fa riferimento allo *standard IEC61131-3*, in cui, nella parte 3 di 8, vengono riconosciuti *5 linguaggi di programmazione* (1,2,3 sono di tipo grafico, 4 e 5 di tipo testuale):

1 **SFC**: *Sequential Functional Chart*;

2 **Ladder Diagram** o *linguaggio a contatti*, tra i più diffusi, adotta una logica simile a quella elettrotecnica, basata su contatti e bobine;

3 FBD *Functional Block Diagram*;

4 **Instruction List**: lista sequenziale di istruzioni;

5 **ST**: *Structured Test.*

Un PLC opera eseguendo in modo ciclico una sequenza di istruzioni; la sua velocità di esecuzione dipende dal numero di ingressi e di uscite, dal clock di macchina e dal numero di istruzioni.

Sono largamente utilizzate architetture di tipo ***host-target***: il programma di controllo viene scritto su PC e scaricato, sotto forma di linguaggio a basso livello, su uno o più dispositivi target, tipicamente PLC; questi ultimi gestiscono variabili in ingresso e uscita con *vincoli real-time*. Il comportamento del sistema deve essere deterministico, deve quindi garantire l'esecuzione delle operazioni entro un dato tempo.

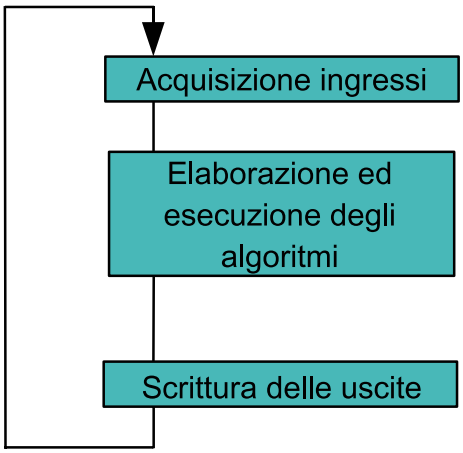

Figura 5.5: Ciclo di programma di un PLC

5.1 Il linguaggio LADDER DIAGRAM

Noto anche come diagramma a scala, è un *linguaggio a contatti*: tra due montanti si costituiscono linee (*rung*) composte da vari elementi, in genere almeno un *contatto* ed una *bobina*. A ciascun elemento viene associata una variabile, il cui valore è conservato in un bit di memoria. Se i valori associati agli elementi realizzano una continuità elettrica, la linea viene percorsa dal flusso di esecuzione da destra a sinistra.

- **Contatti**: ad essi è associata una variabile di ingresso; sono sempre connessi al montante di sinistra.

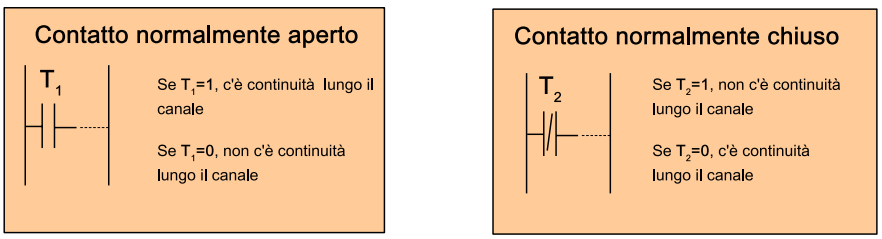

Figura 5.6: Tipi di contatto

- **Bobina**: elemento terminale di un'istruzione, è sempre collegata al montante di destra.

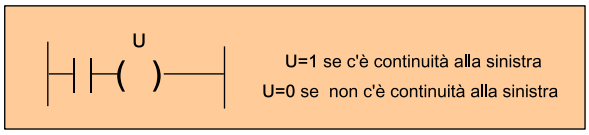

Figura 5.7: Bobina

Il programma scorre ciciclamente la sequenza di pioli dall'alto verso il basso e l'esecuzione di ciascun piolo avviene da sinistra verso destra.

Si realizzano attraverso gli elementi introdotti alcune funzioni logiche come in figura 5.8. In alcuni casi si sfruttano i teoremi di De Morgan:

$$\overline{A \cdot B} = \overline{A} + \overline{B} \tag{5.1}$$

$$\overline{A + B} = \overline{A} \cdot \overline{B} \tag{5.2}$$

Ad esempio per la realizzazione della porta NAND, si sfrutta l'equazione 5.1

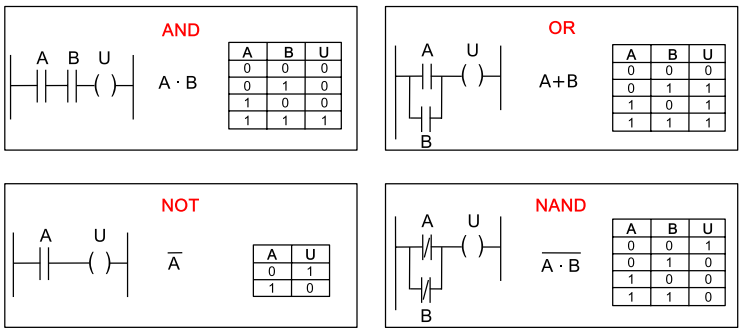

Figura 5.8: Realizzazione di funzioni logiche

5.1.1 Esempio

Si vuol regolare tramite PLC l'accensione di una lampadina, comandata da due interruttori; rispetto al PLC, si definiscono *swl* ed *swr* come variabili d'ingresso, L è la variabile in uscita. Il funzionamento prevede che la lampadina si accenda solo se viene premuto l'interruttore di destra o quello di sinistra; se entrambi vengono premuti a partire dallo stato di lampadina spenta, la lampadina non si accende.

Poiché passare direttamente alla soluzione in Ladder Diagram può non essere intuitivo, è possibile realizzare un passaggio intermedio, ad esempio con una tabella di verità come in figura 5.10.

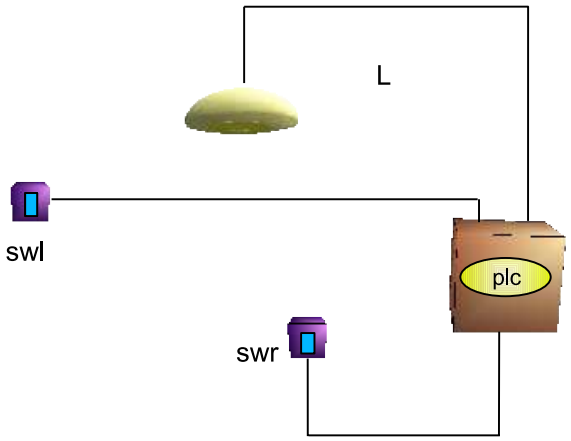

Figura 5.9: Controllo dell'accensione di una lampadina

swl	swr	L
0	0	0
0	1	1
1	0	1
1	1	0

Figura 5.10: Tabella di verità

La funzione descritta dalla tabella è di tipo exor, con espressione

$$L = (\overline{swl} \cdot swr) + (swl \cdot \overline{swr})$$

Si può dunque rappresentare la soluzione in ladder diagram:

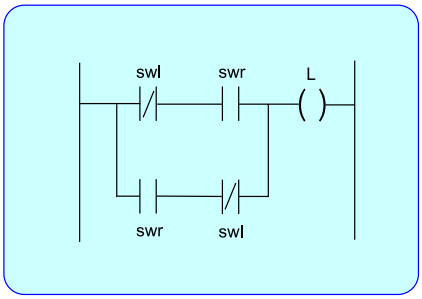

Figura 5.11: Soluzione in ladder diagram

5.1.2 Bobina latch e bobina unlatch

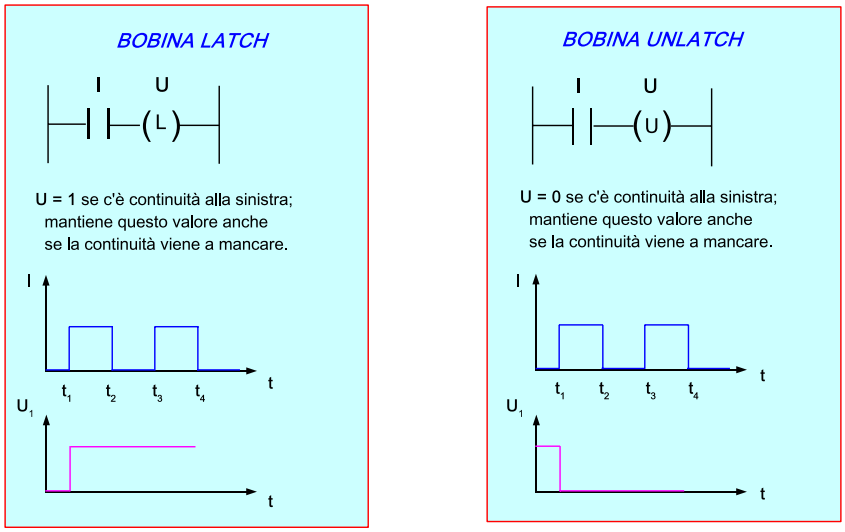

Figura 5.12: Bobina latch e unlatch

La ***bobina latch*** presenta il funzionamento in figura 5.12: quando all'istante t_1 la variabile I in ingresso alla bobina U assume il valore logico alto, la bobina assume il valore logico alto e lo mantiene anche se la variabile che pilota la bobina torna al livello logico basso (negli istanti t_2 e t_4).

La ***bobina unlatch*** presenta un funzionamento duale alla bobina latch: quando all'istante t_1 la variabile I in ingresso alla bobina U assume il valore logico alto, la bobina assume il valore logico basso e lo mantiene anche se la variabile che pilota la bobina torna al livello logico basso (negli istanti t_2 e t_4).

5.1.3 Esempio del carrello

Si prenda in esame un carrello come in figura 5.13 che viene messo in moto dall'abbassamento di una leva g, e dopo esser giunto al punto finale FD (finecorsa destro) di un dato percorso, viene caricato mediante il ribaltamento R di un serbatoio; dopo che il ribaltamento è stato completato FR (fine ribaltamento), il carrello può compiere il suo percorso a ritroso, tornando al punto iniziale FS (finecorsa sinistro).

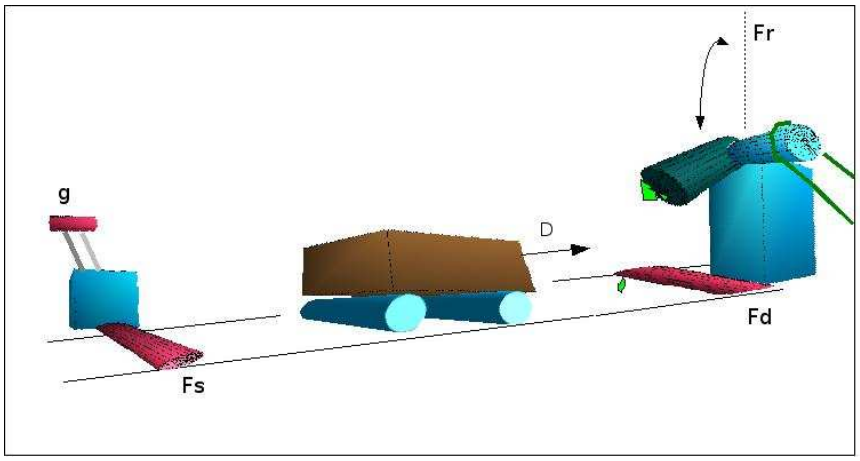

Figura 5.13: Esempio del carrello

Indicati con D il movimento del carrello verso destra ed S il movimento verso sinistra, dal punto di vista del PLC le variabili FS, FD, FR e g sono ingressi, mentre S, D e R sono variabili in uscita (figura 5.14). Si ricorda che queste variabili sono di tipo ***booleano***; in particolare $FS = 1$ significa che il carrello è in posizione di partenza, $FD = 1$ significa che il carrello è presso il fine corsa destro, $FR = 1$ significa che il serbatoio è stato svuotato.

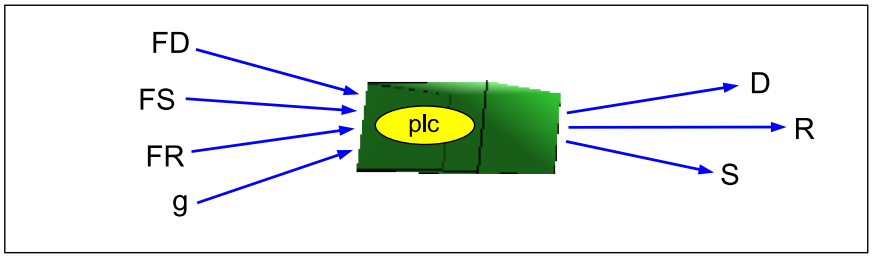

Figura 5.14: I/O nell'esempio del carrello

Fasi del programma:

- ⋆ D (che attiva lo spostamento a destra del carrello) viene posta a 1 tramite bobina latch dopo l'abbassamento della leva g (piolo 1);
- ⋆ finchè non si giunge al finecorsa destro, il valore dell'uscita D è 1, perciò il carrello prosegue in movimento verso destra. Giunti al finecorsa destro (attivazione di FD), si disattiva la variabile D e il carrello si arresta (piolo 2);
- ⋆ quando il carrello è giunto al finecorsa destro inizia il ribaltamento del serbatoio (R), che dura finchè non termina il ribaltamento (quando si attiva variabile di fine ribaltamento FR) (piolo 3);
- ⋆ nel momento in cui termina il ribaltamento si attiva lo spostamento del carrello verso sinistra (S) che resta attivo finchè il carrello non giunge la finecorsa sinistro (piolo 4);
- ⋆ quando il carrello giunge al finecorsa sinistro (attivazione di FS) si arresta lo spostamento verso sinistra (unlatch della bobina S) e il programma riprende il ciclo.

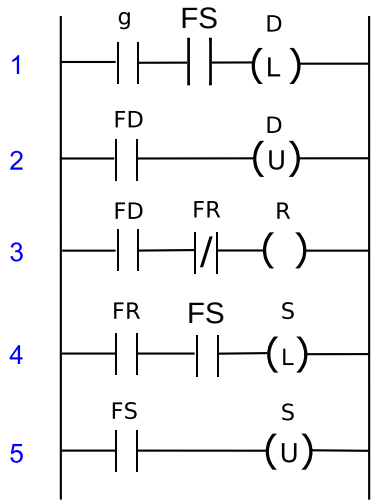

Figura 5.15: Ladder Diagram dell'esempio del carrello

5.1.4 Temporizzatori e contatori

Oltre ai temporizzatori, come illustrato in figura 5.16, esistono anche temporizzatori a ritenuta.

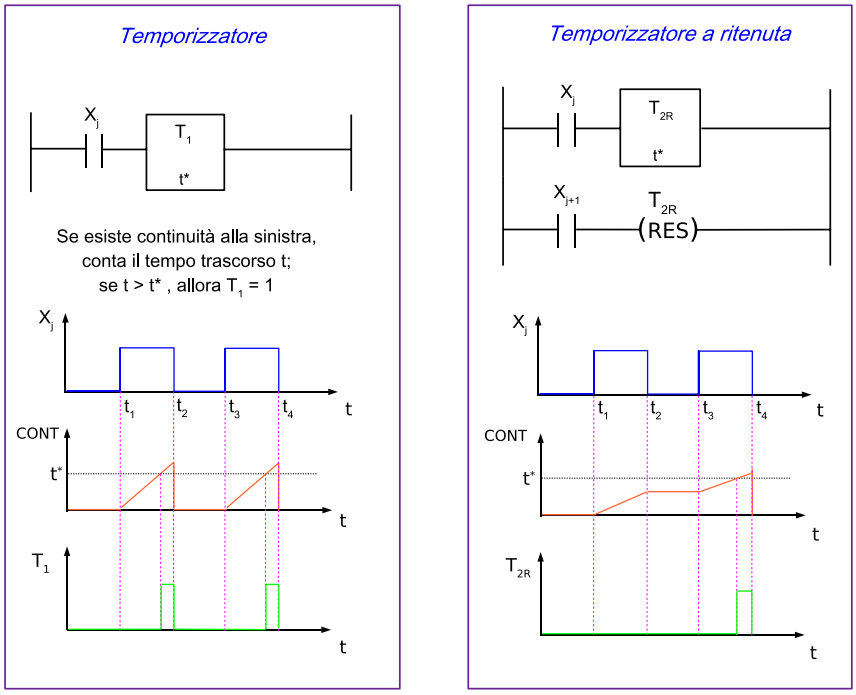

Figura 5.16: Temporizzatore e temporizzatore a ritenuta

Il ***temporizzatore*** attiva la variabile ad esso associata (in figura 5.16 la variabile T_1) dopo che l'ingresso che lo pilota assume il valore logico alto per un tempo almeno pari a t^*. Se la variabile in ingresso al temporizzatore (X_j) torna al livello logico basso (t_2 e t_4), il conteggio si annulla e riprende da 0 nel momento in cui la variabile X_j torna al livello logico alto.

Il ***temporizzatore a ritenuta***, a differenza di quello normale, conserva il conteggio anche se la variabile in ingresso torna a livello logico basso (si veda l'istante t_3 nell'esempio di temporizzatore a ritenuta in figura 5.16). Nel caso di temporizzatore a ritenuta per far ripartire il conteggio è quindi necessario un reset tramite bobina della variabile T_{2R} del temporizzatore stesso.

In figura 5.17 si rappresenta un contatore.

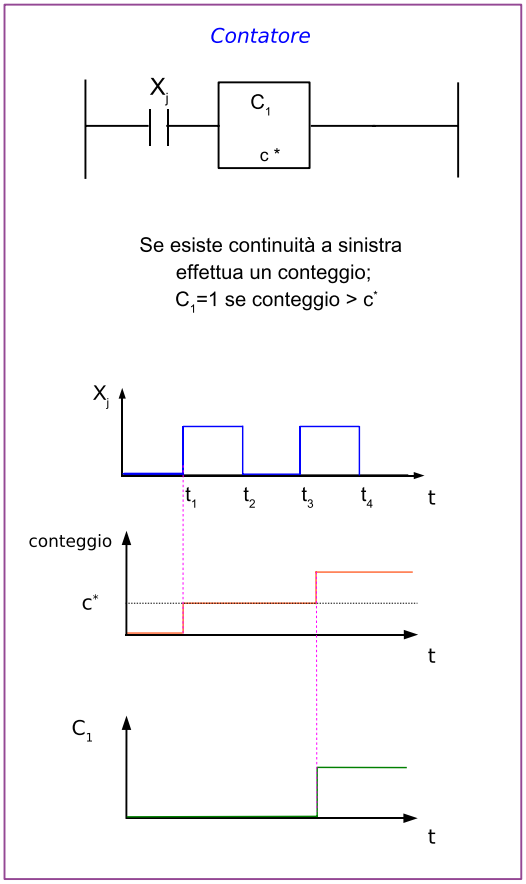

Figura 5.17: Contatore

Il contatore conta quante volte l'ingresso che lo pilota effettua un fornte dal livello logico basso a quello alto. Nell'esempio in figura 5.17 si nota che negli istanti t_1 e t_3 l'ingresso X_j associato al contatore effettui un passaggio dal livello basso a quello alto e in corrispondenza di questi istanti il contatore incrementa il suo valore. In tal caso il contatore conta 2, infatti in prossimità del secondo fronte dell'ingresso in t_3, la variabile C_1 associata al contatore assume il valore logico alto, segnalando che lo stato del contatore è di 2 conteggi.

Per poter essere resettato il contatore

5.1.5 Esempio del carrello con allarme

Si riprende l'esempio del carrello, considerando ora che

- √ Il tempo medio del ribaltamento del materiale nel carrello è di 10s;
- √ Dopo k viaggi, il sistema deve essere sottoposto a manutenzione, perciò va bloccato tramite un allarme.

Si aggiunge allo schema un *temporizzatore* T che, quando il carrello è giunto al fine corsa destro, causi un'attesa di durata $t*$ durante il ribaltamento; si introduce anche un *contatore* C che permetta di azionare un allarme dopo che il sistema abbia eseguito le operazioni per un dato numero di volte.

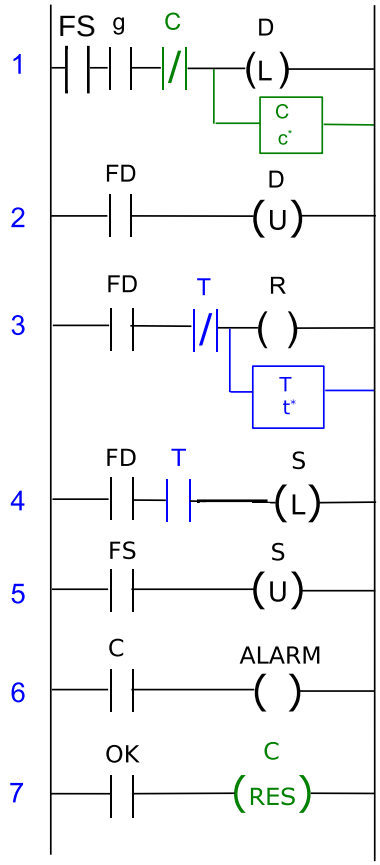

Figura 5.18: Esempio del carrello con allarme

Fasi del programma:

- ⋆ Se $C_1 < c^*$ allora il ciclo è abilitato a partire perchè nel primo piolo il contatto normalmente chiuso pilotato da C_1 offre continuità;
- ⋆ quando vengono contati c^* cicli allora il ciclo relativo ai primi 5 pioli si arresta e viene eseguita la routine di manutenzione (pioli 6 e 7) in quanto

C_1 assume il valore logico alto e offre continuità nel piolo 7, attivando la variabile ALARM che blocca il ciclo principale finchè non viene attivata la variabile OK. L'attivazione di OK porta C_1 al valore logico basso, consentendo al ciclo principale di ripartire e bloccando l'esecuzione dei pioli 6 e 7;

⋆ il *temporizzatore* inizia il conteggio quando il carrello raggiunge il fine corsa destro: T rimane a 0 finchè non è trascorso un tempo indicato dal parametro di tipo intero t^* del temporizzatore: quindi in questo caso il test di fine ribaltamento viene effettuato su T e non su FR;

5.1.6 Estensioni del linguaggio

Istruzioni di salto, operazioni aritmetico-logiche e comparazioni

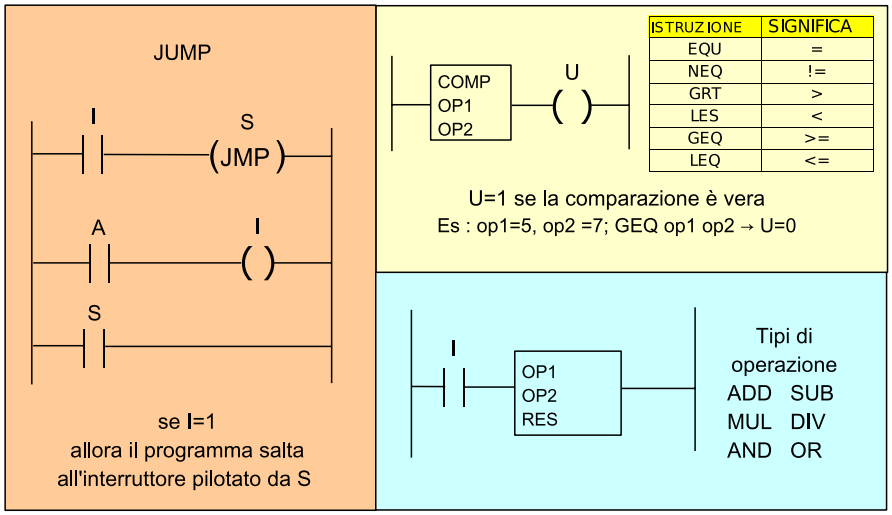

Figura 5.19: Istruzioni di salto e operazioni aritmetico-logiche

- La bobina di tipo JMP fa saltare il processo in esecuzione all'interruttore pilotato dalla variabile associata al salto (nel caso in figura 5.19 all'interruttore pilotato dalla variabile S). Nell'esempio in figura è l'attivazione della variabile A (che a sua volta attiva I che pilota la bobina JMP) a determinare il salto.

- L'istruzione $COMP$ effettua un confronto tra 2 variabili ($OP1$ e $OP2$ in figura) e offre continuità alla sua destra se $OP1 = OP2$. Altri comandi di confronto con funzionamento identico ma condizioni di confronto diverse sono schematizzati in tabella (esempio GEQ offre continuità alla sua destra se $OP1 \geq OP2$).

- L'istruzione aritmetica effettua l'operazione scelta (somma = ADD, differenza = SUB, ...) mettendo nella variabile RES il risultato dell'operazio-

ne; nell'esempio in figura, affinchè l'operazione venga eseguita, la variabile P che pilota l'istruzione aritmetica deve avere valore logico alto.

5.2 Il linguaggio SFC

Mentre il linguaggio ladder utilizza una logica vicina all'elettrotecnica, il linguaggio **SFC** (*Sequential Functional Chart*) è più ad alto livello ; presenta istruzioni schematiche e una logica a più alto livello, che riduce notevolmente i tempi di stesura di un programma.

Il linguaggio è costituito da due elementi fondamentali:

- Le **fasi** o **passi**, rappresentate da rettangoli e univocamente identificate da un numero. Possono essere di tre tipi e sono rappresentate nella figura 5.20.

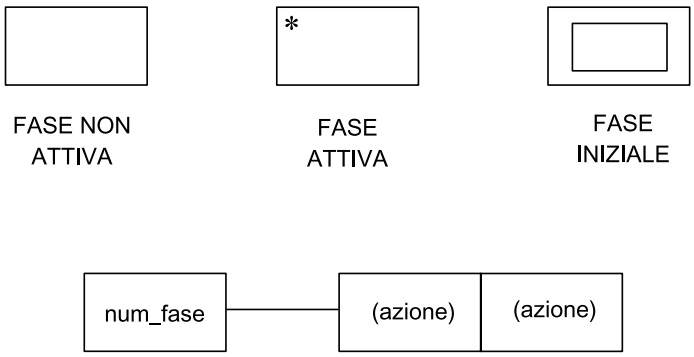

Figura 5.20: Tipi di passi o fasi

La fase iniziale è quella che avvia il programma e si attiva al tempo $t = 0$. A ciascuna fase sono associate una o più azioni, che vengono eseguite solo se la fase è attiva.

- Le **transizioni** consentono al programma di passare dalla fase a monte a quella a valle; vengono rappresentate con una barretta e indicate con la sigla T_n, dove n è il numero che identifica la singola transizione.

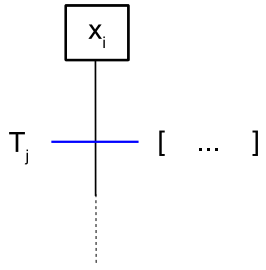

Figura 5.21: Transizione

A fianco della transizione viene posta la condizione che la realizza. Se si omette la freccia, si considera automaticamente l'orientamento verso il basso, perciò se si vuole andare dal basso all'alto è obbligatorio indicare il verso!

5.2.1 Regole di costruzione

- *Tra due fasi esiste sempre una e una sola transizione.*

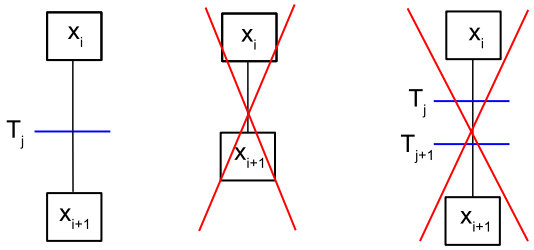

Figura 5.22: Regola di costruzione n°1

Nei casi in figura 5.22 solo il primo è accettabile, in quanto tra due fasi deve esserci almeno una transizione, inoltre non ci può essere più di una transizione.

- *Tra due transizioni esiste almeno una fase.*

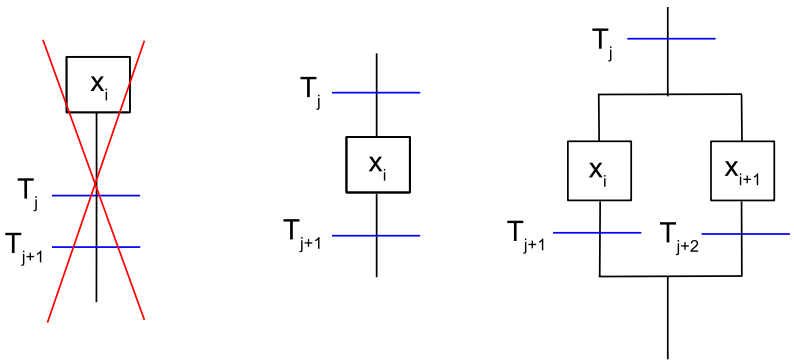

Figura 5.23: Regola di costruzione n°2

Tra i casi in figura 5.23 solo il primo non è accettabile in quanto tra una transizione e l'altra non vi è una fase, ma una transizione; negli altri casi tale ipotesi è verificata, in particolare nel caso alla destra tra la transizione T_1 e le altre vi sono due fasi, ma la regola è rispettata in quanto esiste almeno una fase.

5.2.2 Regole di evoluzione

1 **Condizione dell'SFC**: è l'insieme delle fasi attive di un SFC. La condizione cambia in seguito al superamento di una transizione e in genere è una variabile booleana che può essere vera o falsa. La condizione iniziale è l'insieme delle fasi attive di un SFC al tempo $t = 0$.

2 Una transizione si dice **superabile** se:

A tutte le fasi a monte sono attive (in tal caso la transizione è abilitata a scattare ossia ad essere superata);

B la condizione associata alla transizione (in genere una variabile di tipo booleano) è vera. Quando una transizione diventa superabile, subito viene superata, cioè le fasi a monte si disattivano e quelle a valle si attivano. Tale istantaneità non è invece garantita nelle reti di Petri.

5.2.3 Tipi di variabile

√ **Booleane** $x[0,1]$ il passo è attivo se la x vale 1, è inattivo se la x vale 0.

√ **A fronte** $\uparrow x, \downarrow x$ $\uparrow x$ vale 1 durante il fronte di salita della variabile booleana associata al passo; $\downarrow x$ vale 1 durante il fronte di discesa della variabile booleana associata al passo.

√ **Temporali** $t/X_n/d$ t indica che la variabile è di tipo temporale; X_n è marker della fase n, identifica variabile associata al passo n; d è il tempo che deve passare prima che tale condizione, che poi permette alla transizione di scattare, sia vera. Le variabili temporali sono spesso utilizzate per traslare nel tempo una variabile.

In figura 5.24 si può notare come la variabile a fronte $\uparrow X_1$ abbia un impulso in prossimità dei fronti di salita della variabile X_1 cui è associata (quindi negli istanti di tempo t_1 e t_3), mentre la variabile a fronte $\downarrow X_1$ presenta un impulso in prossimità dei fronti di discesa della variabile X_1 cui è associata (quindi negli istanti di tempo t_2 e t_4). La variabile temporale $t/X_1/d$ conta un tempo pari a d dall'istante in cui la variabile associata X_1 assume valore logico alto (in figura gli istanti t_1 e t_3), dopodichè si porta al livello logico alto finchè la variabile associata non torna a livello logico basso (in figura X_1 va a 0 in t_2 e t_4).

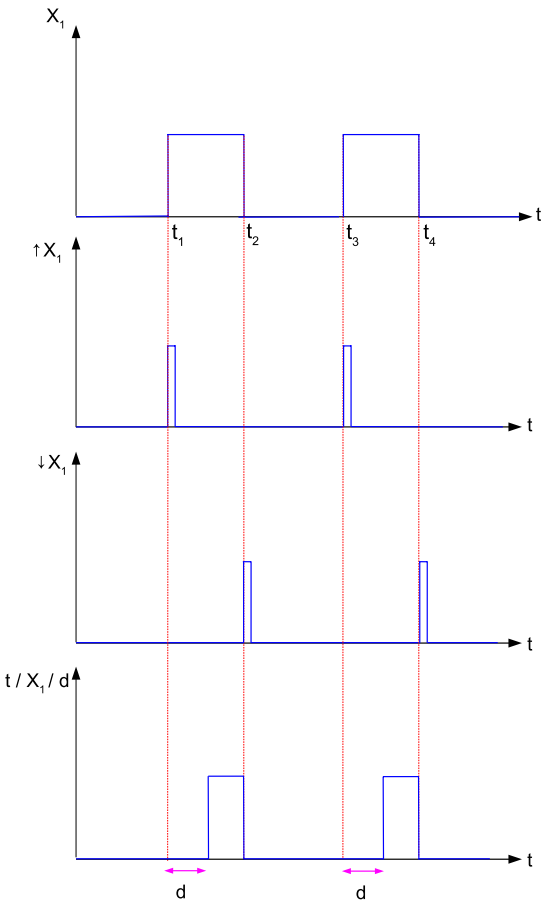

Figura 5.24: Tipi di variabili

5.2.4 Tipi di azioni

- **Continua**: eseguita con continuità finchè il passo è attivo.

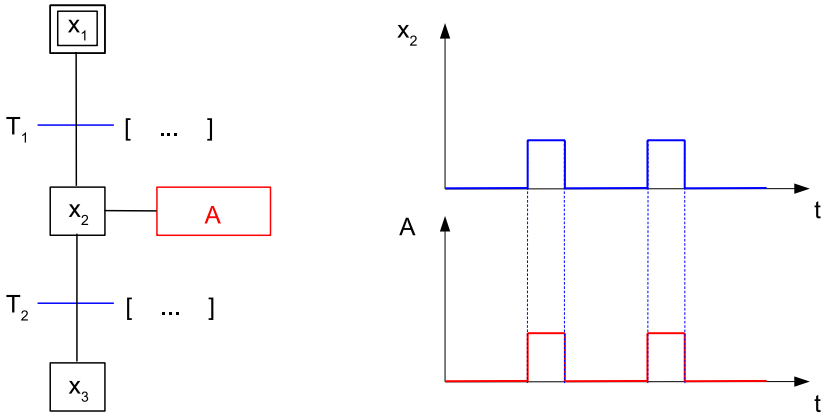

Figura 5.25: Azione continua

- ***Condizionata***: viene eseguita finchè il passo è attivo e la condizione relativa all'azione è vera; c'è quindi un AND tra la condizione C e la variabile X_i associata al passo i.

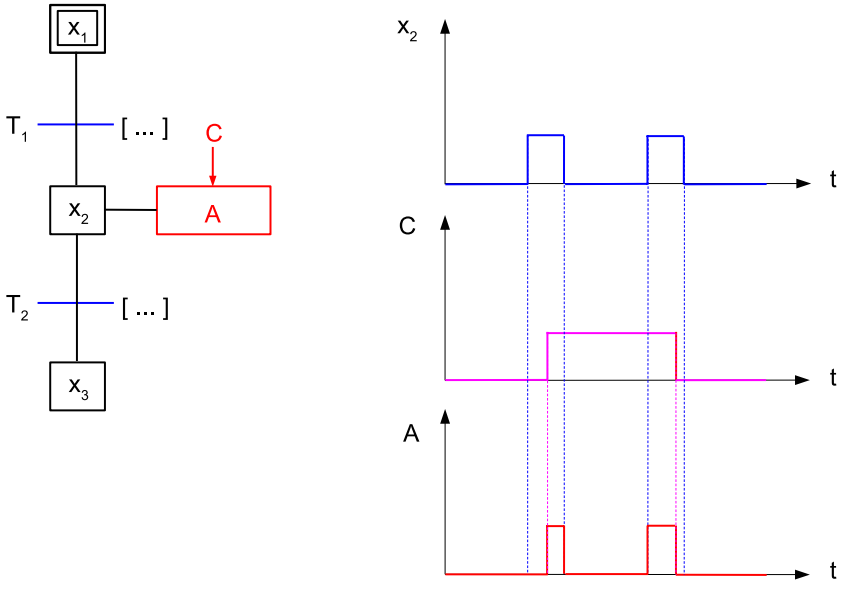

Figura 5.26: Azione condizionata

- ***Impulsiva***: esegue l'azione una sola volta sulla fase temporale in cui l'azione è attiva. L'ampiezza dell'impulso è pari al ciclo di scansione del programma (se il passo è abilitato).

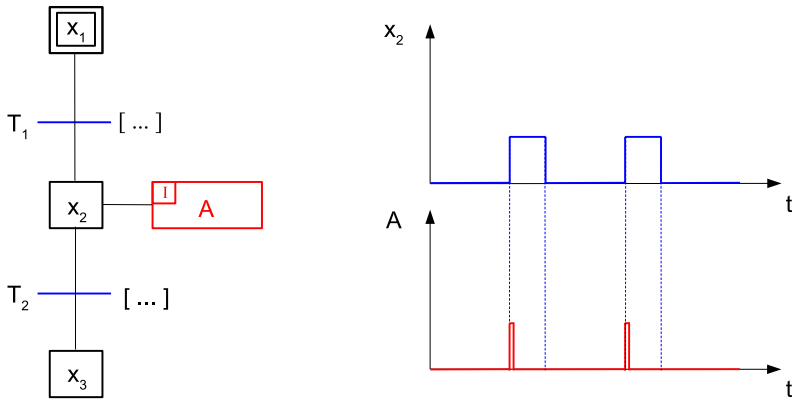

Figura 5.27: Azione impulsiva

- ***Set/reset***: setta (o resetta) una variabile finchè non viene invocata l'azione di reset (o di set) relativa.

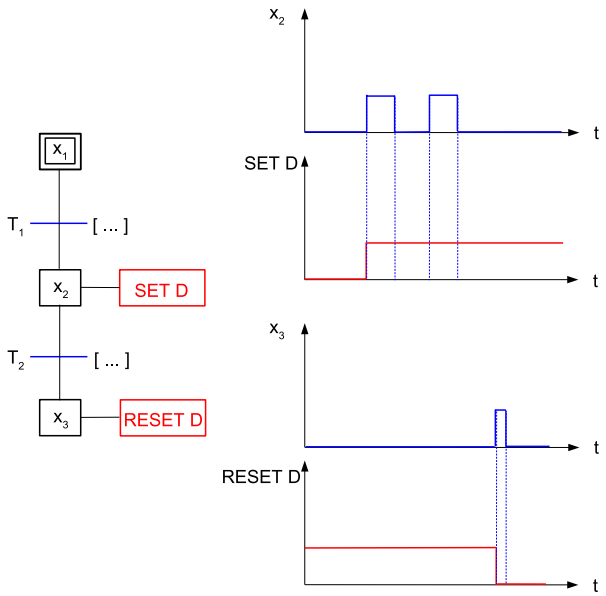

Figura 5.28: Azione set/reset

5.2.5 Tipi di costrutti

- **Scelta**: effettua un test del tipo if then else tra due transizioni per decidere in quale ramo convogliare il processo fermo al passo x_j. Le condizioni C_j e $Cj + 1$ delle transizioni di scelta devono essere *mutuamente esclusive*. La mancanza di condizioni mutuamente esclusive renderebbe il sistema non deterministico (potrebbero tra l'altro essere eseguiti entrambi i rami contemporaneamente).
- **Convergenza**: è in genere utilizzato per la richiusura di due rami a uno solo, in seguito a una biforcazione del processo per una *scelta*.

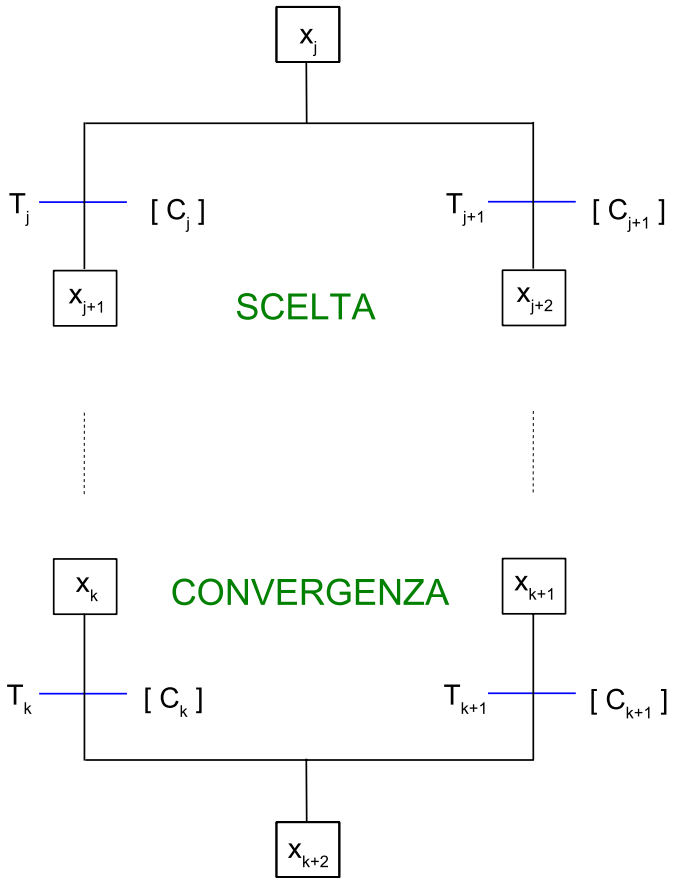

Figura 5.29: Costrutti scelta e convergenza

- **Parallelismo**: è il costrutto duale alla *scelta*; una volta che si verifica la condizione C_j, vengono eseguiti in parallelo due o più rami.
- **Sincronizzazione**: è in genere utilizzato per la richiusura di due o più rami che si sono bifocati in modo *parallelo*. Prima che il processo giunga nello stato x_{k+2}, tutti i passi a monte della transizione T_k devono essere attivi.

Figura 5.30: Costrutti parallelismo e sincronizzazione

Attenzione, utilizzare il costrutto di sincronizzazione a seguito di una scelta provoca un loop infinito in quanto solo uno dei passi a monte potrà essere attivo! Sono invece compatibili biforcazioni con parallelismo abbinate a richiusure di tipo convergenza.

5.2.6 Esempio del carrello

Si riprende l'esempio già risolto nel linguaggio Ladder Diagram con tre varianti.

Versione base

Nella versione base il carrello segue la dinamica già spiegata in precedenza, ma non sono presenti allarme e temporizzatore. Si utilizzano, quindi, solamente variabili booleane.

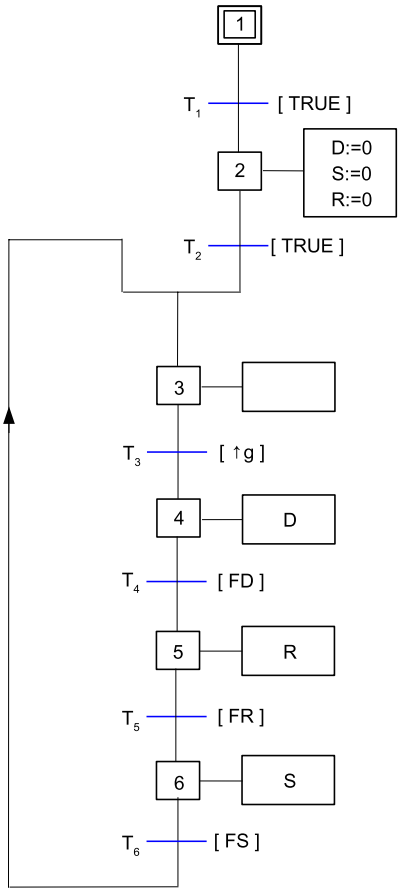

Figura 5.31: Esempio del carrello: versione base

Il programma effettua l'inizializzazione delle variabili al passo 2, che viene attraversato una sola volta dal programma (si è inoltre evitato di inizializzare le variabili al passo iniziale). T_2 e il passo 3 sono stati inseriti per favorire una ripresa del ciclo di programma (che termina alla transizione T_6). Il carrello inizia a muoversi al passo 4, raggiunto dopo che l'operatore ha abbassato la leva provocando un fronte sulla variabile g. La transizione successiva (T_4) verifica lo stato della variabile FD, che si attiva non appena il carrello raggiunge il fine corsa destro. Al verificarsi di tale condizione inizia l'azione di ribaltamento ($R = TRUE$ o R) che si conclude all'attivarsi di FR (fine ribaltamento). In tal caso

si muove il carrello verso sinistra (attivando la variabile S) e al raggiungimento del fine corsa sinistro il processo torna al passo 3, attendendo che l'operatore riabbassi la leva (provocando un altro fronte di salita sulla variabile g) per far ripartire il carrello.

Versione con temporizzatore per il ribaltamento

In questa versione si vuole simulare il tempo del ribaltamento, noto che il tempo massimo necessario per il ribaltamento è di $5s$; si utilizza perciò la variabile temporale così definita:

$$t/X_5/5s$$

in cui X_5 è la varibile che attiva la temporizzazione e $5s$ è il tempo che deve passare prima che la transizione T_5 scatti.

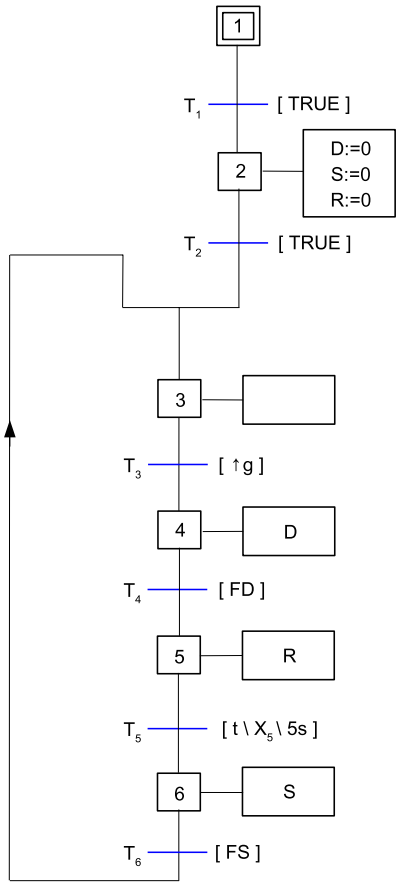

Figura 5.32: Esempio del carrello: versione con temporizzatore

Versione con allarme

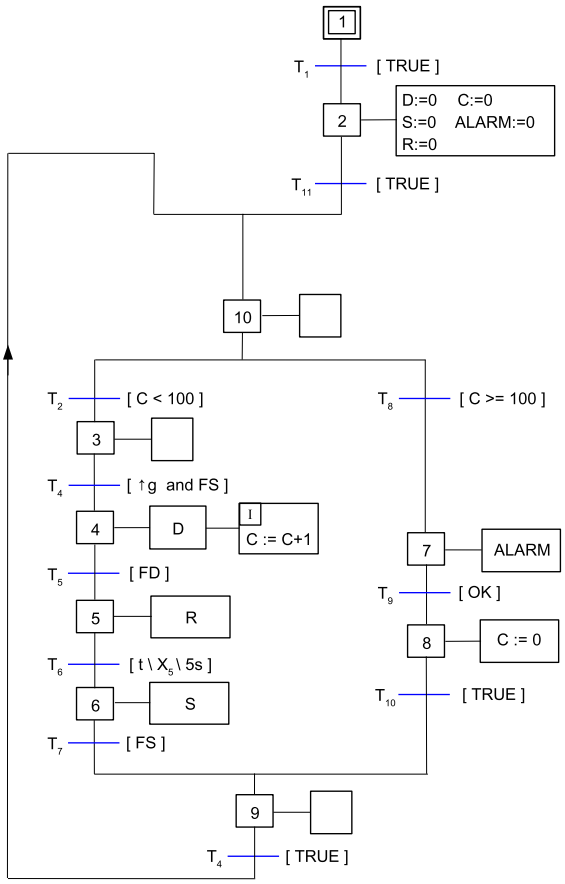

Figura 5.33: Esempio del carrello: versione con contatore

Rispetto alle versioni precedenti, in questo caso dopo 100 cicli, il processo lancia un allarme bloccando il sistema e consentendo all'operatore la manutenzione (che terminerà nel momento in cui si abiliterà la variabile OK). In questa versione è presente una ramificazione di tipo scelta sul contatore dei cicli (variabile C di tipo intero). Se C è minore di 100 si effettua il solito processo di sopstamento a destra, ribaltamento e ritorno a sinistra (in questa versione prima che il carrello riparta da sinistra si verifica che oltre all'abbassamento della leva, si raggiunga il fine corsa sinistro con una condizione di tipo AND). Se C è uguale o maggiore di 100, si passa alla routine di manutenzione, che attiva la variabile $ALARM$. La manutenzione termina quando l'operatore attiva la variabile OK; in tal caso il programma riazzera il contatore ed esce dalla routine. I rami di destra e sinistra si richiudono con una struttura di tipo convergenza; una sincronizzazione metterebbe in deadlock il programma.

5.3 Reti di Petri

Il linguaggio delle Reti di Petri è costituito da tre elementi fondamentali, due dei quali sono molto simili a quelli del linguaggio SFC:

1. I ***posti***, identificati con P_x, con una capacità di n **tokens**. Qui non sussiste un concetto del tipo *passo attivo*, ma si verifica il numero di tokens (identificati da un numero di puntini) presenti nel posto P_x per effettuare valutazioni sullo stato del processo;

2. Le ***transizioni***, identificate con T_x. Rispetto al linguaggio SFC, se una transizione è superabile, non si sa quando viene superata, *lo scatto non è deterministico*;

3. Gli ***archi*** legano un posto a una transizione (o una transizione a un posto) creando tra essi una relazione di flusso. Hanno un peso X_j (pari a uno se non viene indicato), ossia possono essere superati solo quando il numero di tokens nel passo a monte è pari o maggiore al peso dell'arco.

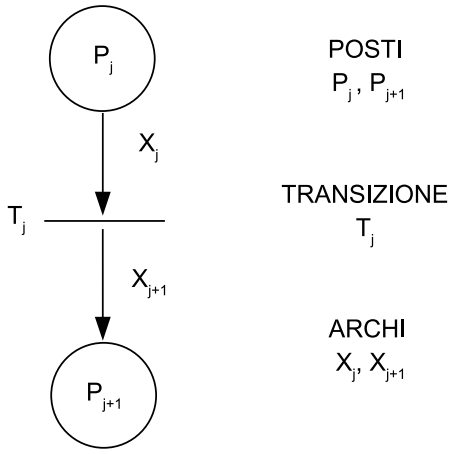

Figura 5.34: Posti, transizioni, archi

Note:

√ Nelle ***Reti di Petri ordinarie*** i pesi degli archi e il numero di tokens massimi per posto è pari a 1;

√ Nelle ***Reti di Petri temporizzate*** è possibile controllare l'istante di scatto di una transizione se questa è superabile.

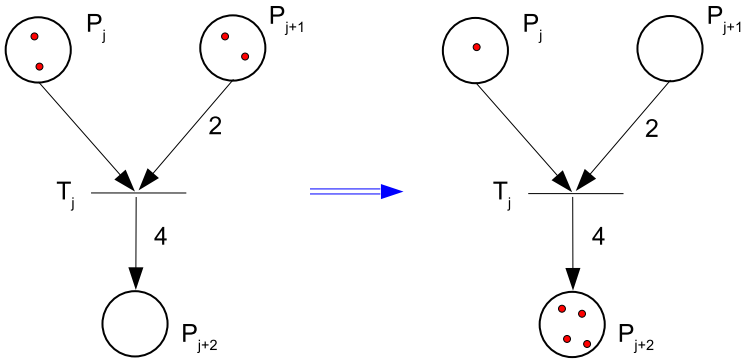

Figura 5.35: Esempio di rete di Petri

5.3.1 Definizioni

A **Rete non marcata**: tripletta $N = (P, T, F)$ con

P = insieme dei posti nella rete;

T = insieme delle transizioni nella rete;

F = relazioni di flussi con pesi degli archi.

1. $P \cap T = \emptyset$
2. $P \cup T \neq \emptyset$
3. $F \subseteq (P \times T) \cup (T \times P)$

B **Rete posti-transizioni**: $P_t = (P, T, F, W, M_0)$ con

W = Funzione del peso di tutti gli archi(se $W = 1$ si hanno le reti di Petri ordinarie); M_0 = ***Marcatura iniziale***: rappresenta tutti i tokens presenti all'inizio in ogni posto della rete.

C **Abilitazione di una transizione**: una transizione T_j è abilitata allo scatto nella marcatura M (uno stato generico della rete) se e solo se tutti i posti a monte hanno almeno un numero di gettoni maggiore o uguale al peso dell'arco che li collega alla transizione;

$\forall P \in \cdot T_j \ , \quad M(P) \geq W(P, T_j)$

$W(P, Tj)$ rappresenta il peso dell'arco orientato.

D **Preset e postset di posti o di transizioni**: il preset di un posto $(\cdot P_x)$ rappresenta l'insieme delle transizioni a monte connesse a quel posto attraverso gli archi; il preset di una transizione $(\cdot T_x)$ è l'insieme dei posti a monte connessi alla transizione attraverso gli archi. Il postset è complementare al preset, in particolare il postset di un posto $(P_x \cdot)$ rappresenta l'insieme delle transizioni a valle connesse a quel posto attraverso gli archi; il postset di una transizione $(T_x \cdot)$ l'insieme dei posti a valle connessi alla transizione attraverso gli archi

Esempio di determinazione di preset e postset

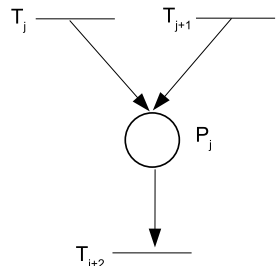

Figura 5.36: Esempio di determinazione di preset e postset

$$T_j\cdot = T_{j+1}\cdot = \{P_j\}$$
$$P_j\cdot = \{T_{j+2}\}$$
$$\cdot P_j = \{T_j, T_{j+1}\}$$
$$\cdot T_{j+2} = \{P_j\}$$

5.3.2 Scatto di una transizione

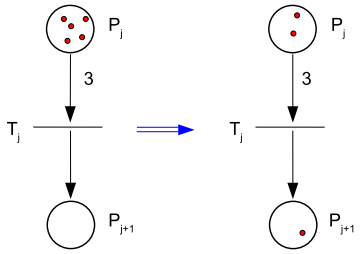

Figura 5.37: Esempio di scatto di una transizione

$$M_0 = [5, 0]$$
$$M_1 = [2, 1]$$

L'evoluzione temporale della rete all'istante temporale successivo porta lo stato dalla marcatura M_0 a una nuova marcatura M_1 come evidente in figura, provocando lo svuotamento di gettoni dei posti a monte ($\cdot T_j$) in numero pari al peso degli archi che puntano alla transizione T_j e il riempimento dei posti a valle ($Tj\cdot$) in numero pari al peso degli archi che partono dalla transizione T_j.

Se ci sono più transizioni abilitate a scattare contemporaneamente, solo una di esse in maniera casuale scatta (***non determinismo delle reti di Petri***)

5.3.3 Strutture fondamentali

† **Sequenza**: le transizioni scattano in modo sequenziale.

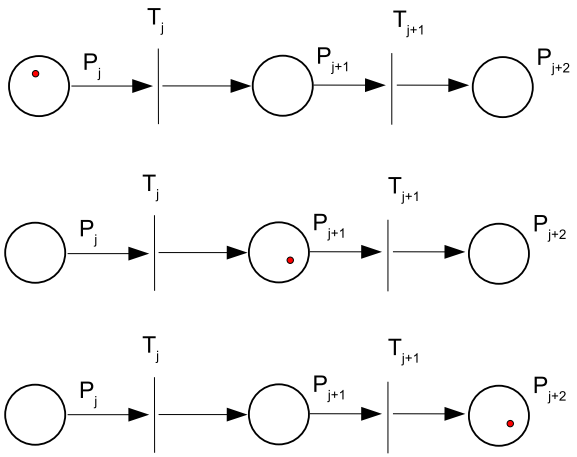

Figura 5.38: Struttura di tipo sequenza

† **Conflitto**: due transizioni T_j, T_{j+1} sono in **conflitto strutturale** se e solo se hanno almeno un posto d'ingresso in comune (vedi 5.39A.); tuttavia questa configurazione non è sufficiente per decidere se le due transizioni sono realmente in conflitto tra loro.

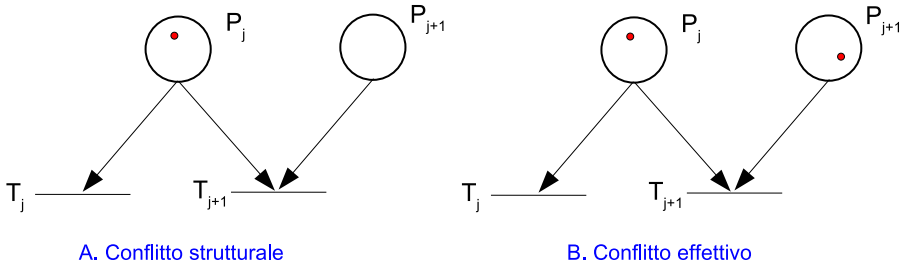

Figura 5.39: Conflitto strutturale ed effettivo

Due transizioni T_j, T_{j+1} sono in **conflitto effettivo** nella marcatura M se sono in conflitto strutturale, se sono abilitate entrambe in M e il numero di gettoni contenuti nei loro posti d'ingresso non è sufficiente a soddisfare contemporaneamente tutti i pesi degli archi che li collegano alle due transizioni (vedi 5.39B.). In tal caso lo scatto di una transizione disabilita l'altra.

Il conflitto stutturale dipende dalla topologia della rete, il conflitto effettivo anche dalla marcatura corrente della rete.

† **Concorrenza**: si verifica se due transizioni (T_j e T_{j+1}) nello stesso momento sono abilitate a scattare. In tal caso solo una delle due casualmente scatta.

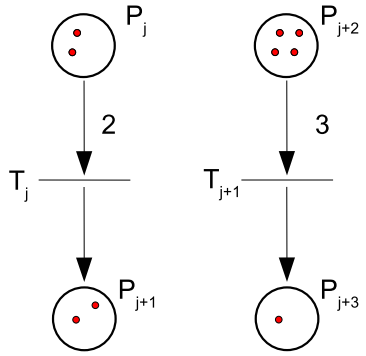

Figura 5.40: Struttura di concorrenza

† **Sincronizzazione**: è molto simile alla struttura omonima in linguaggio SFC, in questo caso tutti i posti a monte della transizione T_j devono dotarsi di un numero di tokens pari o maggiore al peso dell'arco che li collega alla transizione, sincronizzando così due o più processi concorrenti.

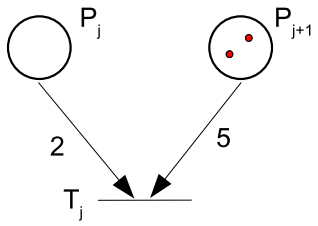

Figura 5.41: Struttura di sincronizzazione

Nel caso in figura 5.41, T_j può evolvere solo se P_j arriva a contenere almeno 2 tokens e P_{j+1} arriva a contenere almeno 5 tokens.

5.3.4 Proprietà delle reti di Petri

⋆ **Raggiungibilità**: la marcatura M_1 si dice raggiungibile da M se esiste una sequenza di scatti in grado di portare da M a M_1.

Esempio

Nell'esempio in figura 5.42 partendo dalla marcatura $M = [2, 0]$, la marcatura $M_1 = [0, 2]$ non è raggiungibile, mentre partendo dalla marcatura $M = [3, 0]$, la marcatura $M_1 = [0, 2]$ è raggiungibile.

⋆ **Reversibilità**: se da ogni marcatura M raggiunta da M_0, si può tornare a M_0, allora la marcatura è reversibile.

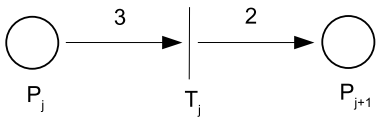

Figura 5.42: Esempio di rete per l'analisi di raggiungibilità

Esempio

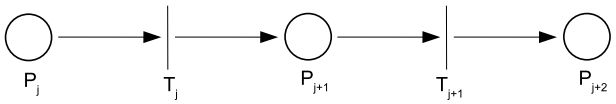

Figura 5.43: 1° esempio di rete per l'analisi di reversibilità

La marcatura $M_1 = [1, 1, 0]$, partendo dalla marcatura $M_0 = [0, 2, 0]$ non è reversibile. In configurazioni come questa non esiste alcuna marcatura reversibile.

Per garantire la reversibilità è necessario aggiungere un collegamento di tipo circolare tra i passi della rete.

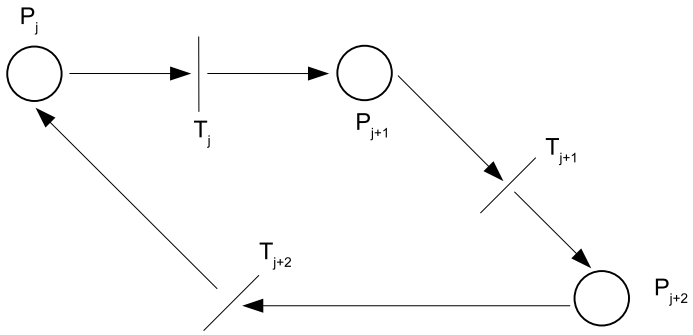

Figura 5.44: 2° esempio di rete per l'analisi di reversibilità

La marcatura $M_1 = [1, 1, 0]$, partendo dalla marcatura $M_0 = [0, 2, 0]$ è reversibile tramite la sequenza di marcature: $M_2 = [0, 1, 1]$ ottenuta facendo scattare T_{j+1} seguita da $M_3 = M_0 = [1, 1, 0]$ ottenuta facendo scattare T_{j+2}.

- ⋆ **Limitatezza**: in tutte le marcature M raggiungibili, il numero di gettoni di ogni posto è pari a k (con k limitato).
- ⋆ **Vivezza**: una transizione T_j è viva se e solo se da qualunque marcatura M raggiungibile se ne raggiunge un'altra M^* in cui T è abilitata. Se tutte le transizioni T sono vive, allora *la rete è viva*. Lo studio di questa proprietà aiuta ad evitare *deadlock* o *sifoni* (assorbimento in una parte della rete di gettoni senza restituzione).

5.3.5 Matrici d'ingresso, d'uscita e di incidenza

Matrice d'ingresso, si definisce come:

$$I|P||T| \text{ con } I(k,j) = W(P_k, T_j),\ \forall (P_k, T_j) \in F$$
$$\text{e}$$
$$I|P||T| \text{ con } I(k,j) = 0,\ \forall (P_k, T_j) \notin F$$

Ogni colonna è associata a una transizione e ogni riga è associata a un posto. L'elemento $I(k,j)$ rappresenta il peso degli archi entranti nella transizione.

Matrice d'uscita, si definisce come:

$$O|P||T| \text{ con } O(k,j) = W(P_k, T_j),\ \forall (P_k, T_j) \in F$$
$$\text{e}$$
$$O|P||T| \text{ con } O(k,j) = 0,\ \forall (P_k, T_j) \notin F$$

Ogni colonna è associata a una transizione e ogni riga è associata a un posto. L'elemento $O(k,j)$ rappresenta il peso degli archi uscenti dalla transizione.

Matrice d'incidenza, si definisce come:

$$C = O - I$$

Descrive la topologia della rete, anche se durante la determinazione di C si rischia di perdere informazioni (gli autoanelli si *vengono celati*).

Esempi di determinazione di matrici d'incidenza

1° esempio

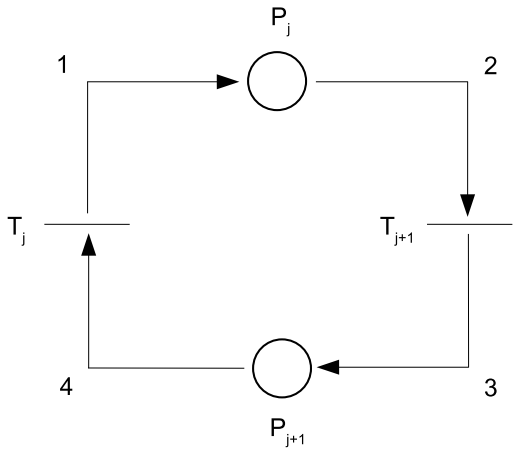

Figura 5.45: 1° esempio di rete per la daterminazione della matrice d'incidenza

$$I = \begin{bmatrix} P_j/T_j & P_j/T_{j+1} \\ P_{j+1}/T_j & P_{j+1}/T_{j+1} \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ 4 & 0 \end{bmatrix}$$

$$O = \begin{bmatrix} T_j/P_j & T_{j+1}/P_j \\ T_j/P_{j+1} & T_{j+1}/P_{j+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$$

$$C = O - I = \begin{bmatrix} 1 & -2 \\ -4 & 3 \end{bmatrix}$$

Il segno $-$ indica che l'arco è entrante nella transizione relativa.

2° esempio

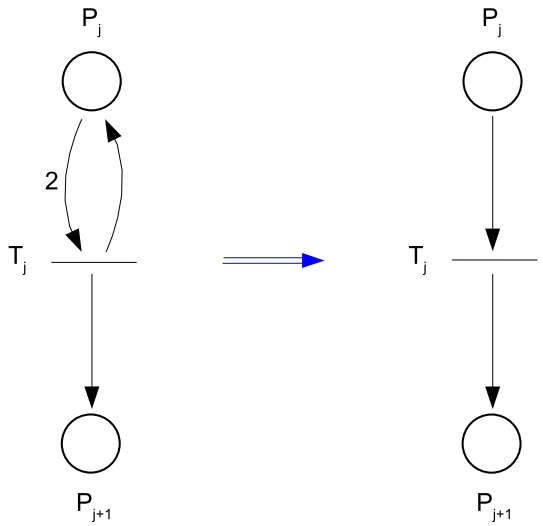

Figura 5.46: 2° esempio di rete per la determinazione della matrice d'incidenza

$$I = \begin{bmatrix} P_j/T_j \\ P_{j+1}/T_j \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

$$O = \begin{bmatrix} T_j/P_j \\ T_j/P_{j+1} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$C = O - I = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

In questo caso si perde l'autoanello tra P_j e T_j.