UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA TRIENNALE IN
INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE

GENERAZIONE AUTOMATICA DI CODICE PLC PER CONTROLLORI MODELLATI CON RETI DI PETRI

AUTOMATIC GENERATION OF PLC CODE FOR CONTROLLERS MODELED WITH PETRI NETS

Studente:
MATTEO SCOTINI

RELATORE:
CHIAR.MA PROF.SSA
SILVIA MARIA ZANOLI

CORRELATORE:
DOTT. ING. CRESCENZO PEPE

ANNO ACCADEMICO 2022-2023

Alla mia famiglia e alla mia ragazza per il supporto continuo in questo lungo e difficile percorso nel quale non è sempre stato facile andare avanti; ma è anche grazie a voi se sono qui oggi. Grazie per aver creduto in me, vi voglio bene.

INDICE

INTRODUZIONE

Nel presente elaborato si vuole affrontare la problematica relativa alla traduzione di codice per il controllore logico programmabile (PLC - Programmable Logic Controller dall'inglese) partendo dal modello logico delle Reti di Petri (Petri Net – PN dall'inglese).

L'obiettivo è quello di partire dal modello di un sistema di automazione industriale che modellato come un Sistema Dinamico ad Eventi Discreti (DEDS - Discrete Event Dynamic Systems dall'inglese) tramite il formalismo delle Reti di Petri (PN) per poi passare all'equivalente codice per l'implementazione su PLC utilizzando una *tecnica di traduzione*.

L'utilizzo delle PN risulta utile poiché permette di analizzare e modellare il comportamento dinamico di un sistema ad eventi discreti; il loro formalismo matematico, infatti, è adatto per modellare sistemi ad eventi discreti concorrenti, asincroni, paralleli; tali caratteristiche sono frequenti in ambito industriale.

Nel seguente lavoro di tesi, per prima cosa verrà introdotta l'automazione industriale partendo dai suoi concetti base; verrà introdotto il modello piramidale CIM (Computer Integrated Manufacturing) che governa un qualsiasi impianto industriale e verranno analizzati i vantaggi e svantaggi che l'automazione industriale ha portato nella società.

Nel capitolo 2 si introduce il concetto di "Sistema Dinamico ad Eventi Discreti", fornendone la sua definizione e tipologia, per poi passare allo strumento di modellazione che verrà utilizzato: le Reti di Petri. Verrà discusso il perché utilizzare la PN piuttosto che un altro modello di analisi come gli automi, e si analizzeranno i vantaggi che le PN portano nel loro utilizzo.

Nel capitolo 3 viene introdotto il PLC e lo Standard IEC 61131 che lo regola, soffermandoci principalmente sul terzo punto della norma, in cui sono riportati i linguaggi di programmazione. I linguaggi di programmazione analizzati sono i più diffusi: Ladder Diagram (LD) e Sequential Function Chart (SFC).

Introdurremo poi nel capitolo 4 due metodi di traduzione:

1. Da PN a LD;
2. Da PN a SFC;

Le metodologie di traduzione analizzate in questo elaborato sono frutto di una ricerca basata su lavori presenti in letteratura.

Per quanto riguarda la traduzione da PN a SFC, si è analizzato una tecnica che prevede di definire il modello di PN, verificare la sua binarietà, e distinguendo se sia o meno binaria, trovare la sua implementazione su PLC equivalente.

Per quanto riguarda il codice LD, invece, verrà analizzata una tecnica basata su diversi step che segue passo dopo passo l'evoluzione della PN.

Si analizzeranno poi le strutture fondamentali di modellazione di una PN: loop, sequenze, concorrenza, mutua esclusione, condivisione delle risorse; si riporta la PN di ogni struttura e la relativa implementazione per PLC utilizzando la tecnica di traduzione introdotta.

Verrà quindi analizzato come vengono gestite le problematiche affrontate precedentemente, come la mutua esclusione o sincronizzazione, introducendo la struttura semaforo (lo vedremo sia con la PN che con l'equivalente codice in SFC) e la priorità dinamica.

Per ultimo verranno analizzati due esempi in cui verranno applicati i metodi di traduzione affrontati.

Il tema della traduzione da PN a codice PLC non è molto diffuso nell'ambito della ricerca. Solitamente, i programmatori PLC si dedicano direttamente alla scrittura del codice e non si soffermano ad analizzare il modello PN associato.

L'obiettivo di questo elaborato è quello di fare un quadro completo di un qualsiasi sistema di automazione industriale: dal modello con PN fino alla traduzione in codice PLC.

Capitolo 1
AUTOMAZIONE INDUSTRIALE

In questo primo capitolo verranno fornite le basi dell'automazione, dai suoi concetti base al sistema di funzionamento secondo il modello CIM; verranno analizzati i vantaggi che essa ha portato e i suoi campi di applicazione.

1.1 *Automazione Industriale: concetti e definizioni*

"*L' automazione industriale consiste nell'impiego coordinato di soluzioni tecnologiche allo scopo di sostituire gran parte del lavoro umano con dispositivi diversi. L'automazione include e supera la semplice meccanizzazione, e cioè la sostituzione, iniziata con la rivoluzione industriale, del lavoro fisico dell'uomo con le macchine*" [1]. Si può definire l'automazione come quella disciplina che studia le tecnologie e le metodologie che garantiscono il controllo di energia e informazioni che serviranno poi a definire i processi produttivi [2].

Figura 1 Evoluzione Industria – UniverseIT: *cosa c'è da sapere e come impatterà le aziende*

Attualmente è in corso il periodo storico definito come Industria 4.0, ma anche se del termine 'automazione industriale' si è iniziato a parlare negli ultimi anni, essa ha origine 'antiche': viene considerato infatti come primo esempio di automazione industriale il "*Regolatore di Watt"* introdotto da James Watt negli anni '700, il quale era un sistema in grado di mantenere a velocità costante le locomotive a vapore [3].

Nonostante la grande evoluzione avvenuta dalla prima Rivoluzione Industriale alla quarta Rivoluzione Industriale (nella quale è nato il concetto di Industria 4.0) tutt'ora in atto, il mondo e la società sono in continua evoluzione: si sta infatti iniziando a parlare di *Industria 5.0*. Il termine comparve la prima volta in un articolo di Michael Rada nel 2015, in cui l'autore sosteneva il ritorno alla centralità della presenza umana in un processo industriale. La principale differenza quindi tra Industria 4.0 e Industria 5.0 è che la prima viene considerata una Rivoluzione Industriale (la quarta), mentre la seconda viene considerata più una Rivoluzione Culturale che Industriale, appunto con il ritorno della presenza umana, supportata però dalle nuove tecnologie introdotte dall'Industria 4.0 [6].

Il concetto di automazione ha diverse definizioni, può essere definito come quello "strumento" necessario affinché un macchinario funzioni in modo automatico appunto e senza l'intervento dell'uomo; sfrutta tutte quelle tecniche e tecnologie sia informatiche che elettromeccaniche per controllare i processi produttivi [7]. Inoltre, si può descrivere un sistema di automazione come un modello piramidale CIM il quale verrà analizzato nel paragrafo 1.2.

L'automazione industriale si suddivide in tre categorie [3], [7], [8]: rigida, programmabile o flessibile;

1. *Rigida*: in cui le operazioni da svolgere prevedono una successione rigida, fissa, e che possono essere svolte da un macchinario solo;
2. *Programmabile*: in cui le operazioni da svolgere possono cambiare di ordine;
3. *Flessibile*: (deriva dalla programmabile): riguarda la gestione della logistica (prodotti e materiali); entrano in gioco AGV, cobot ecc. Nella flessibile si può cambiare tipologia di produzione senza avere tempi morti causati dal cambiamento; questo è possibile se i prodotti finali sono simili (e gli impianti sono flessibili). Questi sistemi prendono il nome di Flexible Manufacturing System (FMS).

Si definisce un "processo produttivo automatizzato*"* la trasformazione di materiali con lo scopo di ottenere un certo prodotto finale desiderato, senza l'intervento dell'uomo [3].

Un processo automatizzato si può ottenere combinando tra loro tre componenti [3]:

1. Energia;
2. Informazione;
3. *Controllo*;

La fase del controllo rappresenta il cervello di un qualsiasi impianto industriale, la quale viene eseguita e svolta da un controllore che riceve informazioni dagli ingressi, li elabora tramite algoritmi, e manda un

segnale in output. Questo cervello in ambito industriale è rappresentato dal PLC - Controllore Logico Programmabile (Programmable Logic Controller dall'inglese). Il PLC è collegato elettricamente e meccanicamente ai sensori e attuatori, viene programmato utilizzando un linguaggio adatto definito dallo Standard IEC 61131 – 3 (che approfondiremo), e come risultato avremo il comportamento desiderato di un impianto di automazione industriale.

1.2 *Modello piramidale CIM*

Il modello piramidale CIM (Computer Integrated Manufacturing) è un modello che rappresenta visivamente (Figura 2) il funzionamento logico di un qualsiasi sistema industriale [3], [7].

Figura 2 Modello Piramidale CIM – [2]

Alla base della piramide si trova il *Livello Campo*, il quale è il livello più basso della gerarchia; qui si trovano tutti i sensori, componenti hardware, attuatori.

Poi salendo di un gradino troviamo la parte del *controllo;* quindi, ci sono gli strumenti, come il PLC, che elabora i dati racconti nel livello precedente, provenienti dai sensori di ingresso per poi tradurli in output.

Salendo ancora di un gradino troviamo il livello di *supervisione di cella*, che supervisiona il funzionamento di tutti i macchinari appartenenti alla cella; in questa sezione vengono svolte le funzioni di controllo e coordinamento delle attività, al fine di avere un comportamento ottimale.

Al penultimo livello viene gestita la base della produzione e viene fatto il coordinamento tra le varie celle per realizzare l'intero processo produttivo.

In cima alla piramide troviamo invece le attività aziendali, in cui vengono gestiti tutti i livelli sottostanti, avviene la pianificazione delle attività da svolgere, pianificazione acquisti ecc. [4], [5].

Come abbiamo visto, un sistema di controllo industriale ha una struttura gerarchica di tipo piramidale; tale struttura è descritta nello Standard ANSI/ISA-S88.01-1995, il quale classifica le funzioni di controllo su tre livelli: controllo di campo, controllo di procedure e controllo di coordinamenti [3]:

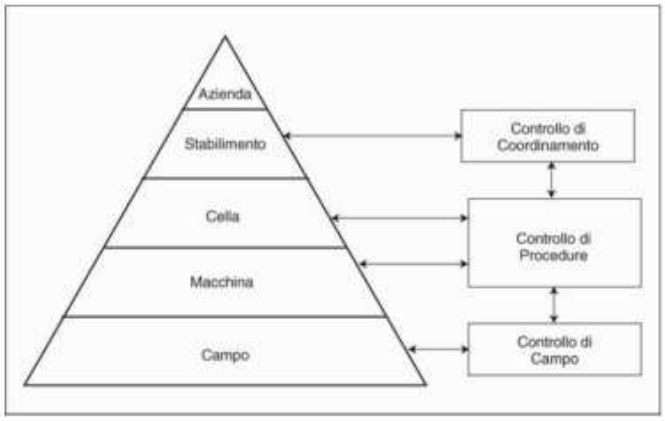


Figura 3 Sistema CIM secondo lo Standard ANSI/ISA-S88.01-1995

Controllo di campo: è il livello più basso nella piramide CIM. È affiancato al livello di campo e fanno parte di tale livello i sistemi di controllo dei singoli componenti di campo.

Controllo di procedure: si colloca ai livelli di macchina e di cella della piramide e rappresenta il controllo di gruppi strutturati di componenti di campo.

Controllo di coordinamento: è il livello più alto della piramide, pari al livello di stabilimento nella piramide. Esso riguarda il coordinamento e gestione delle celle di produzione: manda in esecuzione, dirige o ferma i vari sistemi di controllo.

Nel presente elaborato ci troviamo nel secondo gradino della piramide CIM, cioè il livello riguardante il controllo.

1.3 *Vantaggi e svantaggi dell'automazione industriale*

L'introduzione dell'automazione industriale ha sicuramente portato innumerevoli vantaggi [5], [8], tra cui il miglioramento della qualità dei

prodotti, l'aumento della velocità di produzione ottimizzando al massimo le risorse disponibili, l'accorciamento dei tempi di produzione, il minor costo di produzione, la possibilità di utilizzare lo stesso impianto per differenti prodotti (la produzione flessibile di cui abbiamo parlato). Questo è reso possibile grazie al IIoT (Industrial Internet of Things) che garantisce il controllo di risorse (umane e nel ciclo di produzione), condizioni di sicurezza, il controllo delle attrezzature e impianti ecc. [8].
Ovviamente, c'è stato il bisogno di introdurre delle leggi e norme, per esempio nell'industria alimentare.
Si può dire che l'automazione ha portato ad avere una crescita economica, il consumo di massa è aumentato in contemporanea con l'automazione, abbattendo i costi di produzione [4].
Un altro vantaggio che ha portato l'introduzione dell'automazione è quello sul lato ergonomico: i lavori che prima erano pericolosi se fatti dall'uomo, sono stati automatizzati, aumentando anche le qualità delle condizioni di lavoro.
L'obiettivo dell'automazione è quello di affidare a software e hardware le fasi delle lavorazioni ripetitive e pericolose, impiegando l'uomo in attività che valorizzino le loro capacità.
Anche l'occupazione ne ha risentito; si è iniziato a parlare di "job displacement" (disoccupazione tecnologica) [10], dal momento in cui antiche abilità e professioni potranno non servire più, e quindi la perdita di posti di lavoro non sono causati da licenziamenti, ma dal fatto che come conseguenza dell'introduzione delle macchine, per forza di cose, ci dovrà essere un calo del lavoro dell'uomo. Ma dall'altro lato, sicuramente l'introduzione dell'automazione ha fatto si che sono stati creati altrettanti posti di lavoro, per esempio programmatori, costruttori,

chi installa le strutture, chi monitora il processo ecc; di conseguenza, aumentando la produttività, è cresciuto anche il bisogno di ulteriori assunzioni.

L'idea base dell'automazione, però, non è quella di sostituire l'uomo, bensì di creare una "connessione macchina-uomo".

Il lavoro umano non è stato soppresso dall'automazione, ma riqualificato; un esempio ne sono state le occupazioni di controllo, monitoraggio e manutenzione sul macchinario.

Secondo uno studio fatto dall'IFR (International Federation of Robotics), pubblicato nel 2021 dal World Robot Report, ci sono in media 126 robot ogni 10'000 lavoratori (dato doppio rispetto al 2015 quando ne erano soli 66 in media, quindi aumento di occupazioni descritte sopra: manutenzione, monitoraggio ecc.).

Viceversa, uno svantaggio può essere la difficolta nel programmare sistemi complessi automatizzati; questo lavoro può essere svolto solo da una persona specializzata nella programmazione di tali macchinari: questo sta a significare un maggiore costo di gestione. Un altro svantaggio, se tale si può definire, è la rigidità di alcuni impianti: un impianto potrebbe essere utilizzato solo per svolgere un certo tipo di operazioni limitate, e quindi poca versatilità (l'automazione flessibile risulta migliore rispetto a quella rigida).

L'automazione industriale ha avuto e ha tutt'ora una crescita esponenziale, secondo quanto riportato da Automazione News, il settore dell'automazione ha avuto una crescita di fatturato nel 2022 pari al 23% in più rispetto al 2021; questo ci fa capire che è un mercato in continua espansione, e nonostante i vari problemi geopolitici che ci sono tutt'ora in atto che possono portare a problemi di approvvigionamento di

materiali elettronici e materie prime, le stime per il 2023 sono positive, ma con un aumento ridotto rispetto all'anno attuale.

Dando uno sguardo verso il futuro, "Il Vice President of Strategy and Innovation di Universal Robot", Anders Beck, in un articolo de "*Il Progettista Industriale"* ha dichiarato che il 2023 sarà un anno importante in ambito industriale. La chiave è l'interconnessione uomo-macchina nel lavoro quotidiano, per questo sono stati introdotti i *cobot*, cioè robot industriali pensati per lavorare al fianco dell'uomo; essi sono più flessibili, e più vantaggiosi in termini di costi rispetto ai classici robot.

1.4 *Applicazioni dell'automazione industriale*

Il settore dell'automazione industriale è uno di quei settori che trova posto in diverse applicazioni. Ha applicazione sia nei sistemi di automazione rigida che flessibile. I cobot, introdotti precedentemente, possono operare in ogni settore, anche molto diversi tra loro, grazie alla loro flessibilità che li contraddistingue.

I settori di maggior uso di cobot sono [9]:

- Settore automotive: i campi di applicazioni sono verniciatura, assemblaggio, saldatura;
- Settore meccanico: per esempio operazioni di carico e scarico di un macchinario, trasporto;
- Settore di fine linea: per esempio in una linea di produzione le operazioni di fine linea possono essere inscatolamento, packaging, pellettizzazione;

- Settore alimentare: i cobot sono dotati di certificazione TUV [11]: sono delle società di certificazione in ambito di sistemi di gestione della sicurezza alimentare, ambientale e della qualità;
- Settore elettronico: i cobot hanno una precisione fino a 0.03 mm.

Capitolo 2
MODELLAZIONE DI SISTEMI AD EVENTI DISCRETI LE RETI DI PETRI

2.1 *Sistemi dinamici a Eventi Discreti*

Possiamo definire un sistema dinamico a eventi discreto (DEDS - Discrete Event Dynamic Systems, o sistema guidato dagli eventi 'event-driven') come un sistema dinamico il cui comportamento dipende dal verificarsi di eventi istantanei con cadenza irregolare. L'evoluzione di un sistema di questo tipo è di tipo *asincrono*, basate cioè sulle occorrenze, avvenimenti e non su una temporizzazione definita a priori [3].
Un DEDS ha principalmente due caratteristiche fondamentali [12]:

1. Lo spazio degli stati è discreto;
2. L'evoluzione dello stato è causata dal verificarsi di eventi: in un sistema di questo tipo un evento coincide con l'inizio o il completamento di un'azione.

Il ruolo del tempo nei sistemi ad eventi discreti non è rilevante; c'è, accade tutto nel tempo, ma l'evoluzione dipende dal verificarsi o meno di eventi; il verificarsi di un evento coincide con una transizione di stato.
Nella metodologia asincrona dei sistemi DEDS, il clock coincide con il verificarsi di un evento, indipendentemente dal tempo trascorso tra l'accadimento di due eventi; infatti, gli eventi accadono all'inizio o alla

fine di un'attività (l'attività è un'azione che inizia al soddisfacimento di certe condizioni) [13].

Eventi possibili possono essere [12]: il superamento di un livello, la pressione di un tasto che dà il via a una certa attività, l'occupazione o il rilascio di una risorsa, il guasto di una macchina in un sistema di produzione ecc.…, cioè tutte quelle operazioni che causano il cambiamento di stato. Invece, le variabili di stato per un DEDS sono le condizioni logiche di una risorsa (ferma o in movimento, alta o bassa, piena o vuota).

I sistemi dinamici sorgono nei domini della produzione, della robotica, del traffico veicolare, della logistica (trasporto e deposito di merci, organizzazione ed erogazione di servizi), reti informatiche e di comunicazione: tutte queste applicazioni richiedono operazioni di controllo e coordinamento per garantire il flusso ordinato degli eventi.

Attraverso il modello di un sistema dinamico a eventi discreti è possibile definire il comportamento astratto del sistema, calcolato dal verificarsi di eventi: questa verifica di eventi viene detta "*traccia degli eventi"*. Possiamo definire allora un Sistema Dinamico ad Eventi Discreti come quel modello matematico che rappresenta l'insieme (finito) di queste tracce [13].

I sistemi a eventi si dividono tra due categorie [3]:

1. *Logici*: sono modelli in cui è rilevante solo l'ordine di accadimento della traccia degli eventi; la traccia è formata dall'insieme degli eventi (il tempo in cui accade l'evento non è importante);
2. *Temporizzati*: sono modelli in cui è rilevante sia l'ordine della traccia degli eventi sia l'accadimento nel tempo: in questo caso la

traccia è formata da una coppia dipendente evento-tempo. A loro volta quelli temporizzati si dividono in deterministici e stocastici:

- Deterministici: in cui l'intervallo tra il verificarsi di due eventi è noto a priori;
- Stocastici: duale rispetto al deterministico, l'intervallo tra due eventi è casuale;

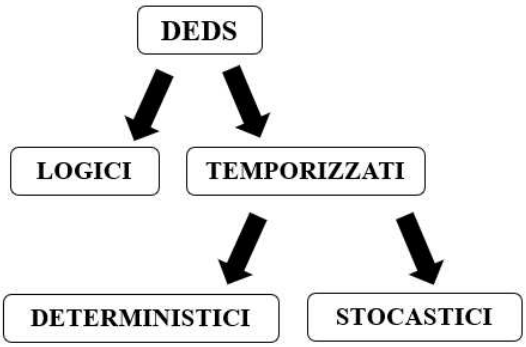


Figura 4 Schema modelli DEDS

I modelli logici vengono utilizzati per svolgere un'analisi delle proprietà qualitative di un sistema, detta *analisi strutturale* (verrà approfondita in un paragrafo del capitolo 4); invece i modelli temporizzati, dal nome, aiutano lo studio delle proprietà prestazionali di un sistema, come i tempi di lavorazione o il numero di pezzi prodotti in un certo tempo [12].

Tra i modelli logici più comuni ci sono gli *Automi a stati finiti* e le *Reti di Petri*.

Utilizzando gli automi, per esempio, un modo per analizzare la dinamica di un sistema è quello di studiare il linguaggio generato da esso. In questo elaborato, però, ci soffermeremo solamente sui modelli logici modellati tramite le Reti di Petri. Anche i linguaggi di programmazione per PLC sono destinati al controllo logico dei sistemi a eventi discreti; quando introdurremo lo Standard IEC 61131-3, i linguaggi possono

essere visti e pensati come modelli di controllo per i DEDS. Inoltre, sono uno strumento di modellazione grafica e matematica applicabile a molti sistemi in ambito industriale [3].

2.2 *Modello Logico: le Reti di Petri*

Le Reti di Petri (Petri Net dall'inglese - PN) sono il più classico esempio di modello logico di un sistema ad eventi discreto. Vennero introdotte per la prima volta nel 1962 dal ricercatore tedesco Carl Adam Petri nella propria tesi di dottorato intitolata "Kommunication mit Automaten", in cui le presentavano come uno strumento adatto alla modellazione di sistemi concorrenti, descrivendo le relazioni tra transizioni ed eventi (non a caso vengono nominate anche reti Posti-Transizioni o reti P/T). Il modello della rete è considerato un modello logico alla pari degli automi, ma tra di essi ci sono alcune differenze [3], [12]:

- Negli automi non si possono rappresentare sistemi a stati infiniti con un numero di nodi finito, viceversa a quanto accade nelle PN;
- In una rete, lo stato del sistema e la transizione sono concetti distribuiti, cioè lo stato finale del sistema totale è formato da più stati parziali e indipendenti. Stessa cosa per una transizione, che influenza solo una parte del sistema finale.

Allora possiamo dire che le Reti si possono considerare come il modello più efficace per studiare e rappresentare i sistemi dinamici a eventi discreti, ne danno una loro rappresentazione grafica. Allo stesso tempo il formalismo della rete è molto adatto per l'implementazione come

linguaggio informatico (come vedremo, l'SFC è graficamente simile a una PN). Esse hanno alcuni vantaggi: essendo un formalismo matematico e *grafico*, sono di facile comprensione e permettono di applicare numerose tecniche di analisi per studiarne le proprietà; inoltre, permettono di rappresentare concetti fondamentali di un sistema dinamico, tra cui sincronizzazione, concorrenza, condivisione di risorse, conflitti. Hanno una rappresentazione grafica *compatta*: le reti di Petri permettono di rappresentare sistemi che hanno un numero infinito di stati con un numero finito di nodi [3], [12], [13].

Inoltre, le PN hanno una rappresentazione *modulare*: le parti costituenti un sistema le posso considerare come moduli indipendenti e analizzarli singolarmente indipendentemente dagli altri; la rete risulta quindi scomponibile in più sottoreti che si possono unire tra loro mediante operatori di rete.

Per esempio, supponiamo di avere a disposizione un impianto di automazione industriale costituito da due macchinari separate da un nastro trasportatore. Essendo un modello modulare, il sistema risulta scomponibile nel seguente modo: si modella prima la sottorete relativa alla prima macchina, poi la sottorete della seconda macchina, e infine la sottorete del nastro. Concluse le singole modellazioni delle sottoreti, esse vengono unite tra loro per formare il sistema complesso finale [12].

2.2.1 *Definizioni Reti di Petri*

In questo paragrafo ci soffermeremo a dare le definizioni base della rete di Petri e le loro regole di evoluzione [3], [12], [13], [22]:

Definizione (Rete)

Una Rete di Petri è una quintupla PN = (P, T, F, W, MO), dove:

- P = {p1, p2, …, pm} è un insieme finito di posti, $|P| < \infty$;
- T = {t1, t2, …, tn} è un insieme finito di transizioni, $|T| < \infty$;
- $F \subseteq P \times T \cup T \times P$ è un insieme di archi (relazione di flusso), ci dice quali coppie ordinate PT o TP (non valgono PP né TT perché come vedremo sono vietate) sono connesse da un arco, e dice quindi il verso dell'arco
- W: F → (1, 2, 3, . . .} è una funzione peso, associa ad ogni arco orientato un numero intero positivo;
- M0: P → (0, 1, 2, 3, . . .} è la marcatura iniziale, ci dice quanti gettoni (o token) ci sono all'inizio in ogni posto della rete.

Inoltre, devono valere queste due condizioni:

- $P \cap T = \emptyset$;
- $P \cup T \neq \emptyset$, in una rete ci devono essere almeno un posto o una transizione;

Una rete può essere rappresentata tramite un grafo bipartito, composto da due tipi diversi di nodi:

1. Posti;
2. Transizioni;

Nella simbologia tradizionale, i posti vengono indicati con cerchi, le transizioni con quadrati o barrette e le loro connessioni con archi orientati. Gli elementi base di una Rete di Petri sono:

1. Posto: 
2. Transizione: 
3. Arco orientato: 
4. Peso dell'arco (1 se omesso): 
5. Token (o gettone); 

Per ottenere dei modelli progettati con reti di Petri, si devono combinare tra di loro gli elementi introdotti, facendo attenzione a rispettare una regola: la relazione F lega tra di loro posti/transizioni e transizioni/posti, ma non permette passaggi di token tra due posti o tra due transizioni.
La progettazione di modelli con PN definisce dei vincoli di progettazione: non si possono collegare tra loro due posti né due transizioni. I passaggi permessi e vietati sono [12]:

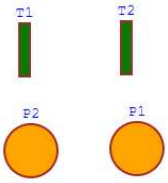


Figura 6 Passaggi vietati

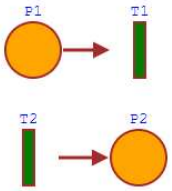


Figura 5 Passaggi permessi

I posti della rete di Petri contengono l'informazione sullo *stato* del sistema, mentre le transizioni rappresentano gli eventi che modificano lo stato del sistema. Per ogni posto o transizione della rete si possono definire l'insieme degli elementi collegati indicando anche la direzione del flusso: in aiuto a questo, vengono introdotte le funzioni di *Preset* e *Postset* [12].

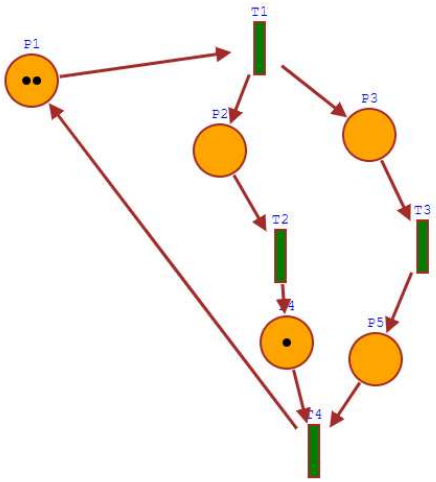
Definizione (preset e postset)

Sia $X = P \cup T$. Si definisce *preset* di un elemento $\alpha \in X$ l'insieme così definito: $Pre(\alpha) = \{\delta \in X \mid (\delta, \alpha) \in F\}$.

Si definisce *postset* di un elemento $\alpha \in X$ l'insieme così definito:

$Post(\alpha) = \{\delta \in X \mid (\alpha, \delta) \in F\}$.

Scrivendo il postset e il preset in modo più semplificativo: $\bullet\alpha = Pre(\alpha)$, $\alpha\bullet = Post(\alpha)$, cioè: l'insieme Preset sono tutti quegli elementi collegati a monte, e viceversa, il Postset sono tutti gli elementi collegati a valle. Un piccolo esempio di Pre e Post:



Il Post set di T1 sono quei posti che sono a valle della transizione stessa, in questo caso sono i posti P2 e P3. Invece, per quanto riguarda il Pre set sempre di T1, è solo il posto P1. Il Pre set di P1 invece è la transizione T4. Come si può notare,

si può fare il pre e post sia del posto che della transizione.

Scrivendolo nella forma abbreviata definita, si ha:

•T1 = {P1}, T1• = {P2, P3}, •T2 = {P2}, T2• = {P4}, •T3 = {P3}, T3• = {P5}, •T4 = {P4, P5}, T4• = {P1}.

2.2.2 *Regole di evoluzione della rete*

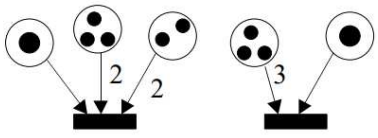
Quando viene modellato un sistema dinamico tramite la rete di Petri, all'inizio dovrà contenere un certo numero di token in determinati posti; la loro definizione viene detta marcatura iniziale, che rappresenta lo stato del sistema in quel momento. Dalla situazione iniziale, poi, tramite delle regole di evoluzione, ci sarà un passaggio di token tra i vari posti della rete, raggiungendo altre possibili marcature [12], [13].

Definizione (Abilitazione di una transizione)

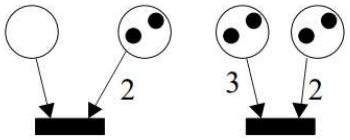
Una transizione t si dice abilitata allo scatto nella marcatura M se e solo se:

$$\forall p \in \bullet t,\ M(p) \geq W(p, t)$$

Una transizione si dice abilitata se tutti i posti a monte di essa hanno un numero di token maggiore o uguale al peso dell'arco che li collega. L'effetto dello scatto di una transizione è il raggiungimento di una nuova marcatura.



In questo caso, entrambe le transizioni sono abilitate, in quanto i posti a monte di essa hanno un numero di token maggiore al peso dell'arco.



In questo esempio, invece ci sono due transizioni non abilitate allo scatto: nella prima transizione, per far sì che scatti, il primo posto deve contenere almeno un token. Nella seconda transizione, invece, il posto P1 ha bisogno di un ulteriore token.

Nel caso in cui il peso dell'arco viene omesso, è sottointeso che sia considerato 1, quindi basta un token per abilitare la transizione allo scatto.

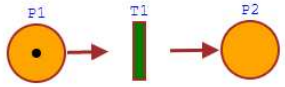
Definizione (Scatto di una transizione)

Data la marcatura M, lo scatto di una transizione t abilitata in M produce una nuova marcatura M' tale che: [6]

1. $\forall p \in {\bullet}t - t{\bullet} \rightarrow M0\,(p) = M(p) - W\,(p, t)$
2. $\forall p \in t{\bullet} - {\bullet}t \rightarrow M0\,(p) = M(p) + W\,(t, p)$
3. $\forall p \in t{\bullet} \cap {\bullet}t \rightarrow M0\,(p) = M(p) - W\,(p, t) + W\,(t, p)$
4. Altrove $\rightarrow M0\,(p) = M(p)$

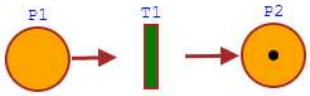
Le condizioni (sinistra) indicano i posti che sono solo a monte, quelli solo a valle, quelli che sono sia a monte che a valle e quelli che non sono né a monte né a valle; lo scatto di una transizione consegue la rimozione e l'aggiunta di un token dai posti a monte a quelli a valle, in base al peso dell'arco che li collega. Una nota è che non per forza il numero di token della rete completa rimane invariante dopo lo scatto delle transizioni, questo accade perché i pesi degli archi sono diversi tra loro.

Come si è detto nella parte iniziale del capitolo, lo scatto di una transizione è un evento locale della rete, influenza una piccola parte e non il sistema completo. Infatti, lo scatto e l'abilitazione dipendono solo dai posti che sono collegati alla transizione. Secondo quanto definito:



facendo evolvere questa piccola rete, lo scatto di t1 dovrebbe togliere il token dal posto a monte (P1) e aggiungerlo in quello a valle (P2).

Facendo scattare t1:



Lo scatto di t1 ha rimosso il token nel posto di input aggiungendolo al posto di output, raggiungendo una nuova marcatura (un nuovo stato).

2.2.3 *Non determinismo*

Le regole di evoluzione definite precedentemente, non coprono tutti i possibili casi. Può capitare che in una rete, le transizioni abilitate a scattare siano contemporaneamente più di una. [12] L'approccio più semplice, se si verifica questo caso, è quello di scegliere un elemento casuale (approccio non deterministico). Questo, però, ha senso solo in ambito concettuale di analisi e teorico, non pratico. Se siamo in un sistema industriale, non si possono far scattare transizioni casuali, ma devono scattare secondo una ben precisa logica studiata a priori. Una volta che una transizione abilitata scatta, per decidere quale sarà la futura transizione abilitata a scattare si deve fare una nuova valutazione della rete, in quanto la marcatura raggiunta dallo scatto della precedente transizione può aver abilitato nuove transizioni e aver disabilitato alcune di quelle abilitate in precedenza.

La parte delle Reti di Petri sull'analisi strutturale non verrà trattata in questo capitolo, ma avrà un paragrafo del capitolo 4 dedicato (in cui si

vedranno i modelli delle strutture delle PN e la loro implementazione in codice SFC utilizzando la tecnica di traduzione trattata).

2.2.4 *Strutture fondamentali*

Le strutture di una Rete di Petri sono tutte quelle problematiche che si incontrano nel corso di modellazione di un sistema industriale e che caratterizzano la loro evoluzione. Tra le fondamentali troviamo: sequenza, parallelismo, sincronizzazione, conflitto, mutua esclusione. In questo paragrafo daremo le loro definizione, mentre il modello lo vedremo nel capitolo della traduzione [3], [12], [13]:

- *Sequenza*: un evento si veridica solamente dopo che si è verificato il precedente, seguendo quindi un preciso ordine;
- *Parallelismo*: gli eventi possono verificarsi indipendentemente dall'ordine;
- *Sincronizzazione*: gli eventi devono essere ricongiunti in un unico punto per poter procedere;
- *Conflitto*: solo un evento può accadere tra più eventi;
- *Mutua esclusione*: l'accesso a una risorsa avviene in modo mutuamente esclusivo, uno alla volta;

2.2.5 *Proprietà delle Reti di Petri*

Diamo le definizioni delle proprietà delle Reti [12]:
Definizione (raggiungibilità)

Una marcatura M1 si dice raggiungibile a partire da una determinata marcatura M se esiste almeno una sequenza di transizioni che facendole scattate a partire da M si ottenga M1.

Definizione (Insieme di raggiungibilità)
Si definisce insieme di raggiungibilità R (N, M0) di una rete N con marcatura iniziale M0 l'insieme più piccolo di marcature tale che:

- $M0 \in R(N, M0)$;
- se $M1 \in R(N, M0)$ e $\exists t \in T \mid M1[t > M2, \rightarrow M2 \in R(N, M0)$;

Calcolare l'insieme di raggiungibilità sarà un passo fondamentale per introdurre il metodo di traduzione per il passaggio da una rete all'equivalente codice per PLC come si vedrà.

In un sistema di automazione industriale, una caratteristica fondamentale è quella di ripristinare una condizione dopo per esempio un guasto, oppure si vuole tornare alla condizione di partenza dopo una serie di attività. Queste caratteristiche dipendono dalla reversibilità e dallo home-state.

Definizione (Rete reversibile)
Una rete di Petri N con marcatura iniziale M0 si dice reversibile se $\forall M \in R(N, M0)$, M0 è raggiungibile da M.

Definizione (Home-State)
Uno stato della rete di Petri Mi è detto essere home state se $\forall M \in R(N,M0)$ si ha che Mi è raggiungibile da M.

Un'altra caratteristica fondamentale che deve avere una rete di Petri è la limitatezza; serve per controllare se i posti accumulano token durante l'evoluzione.

Definizione (Posto k limitato)
Un posto di una rete si dice k-limitato se in tutte le marcature raggiungibili a partire da una marcatura iniziale il numero di gettoni presenti nel posto non supera mai un valore prefissato k.

Definizione (Rete k-limitata e limitata)
Una rete si dice k-limitata se tutti i posti sono k-limitati. Una rete si dice limitata se è k-limitata per qualche valore finito di k.

- *Caso particolare*: un caso particolare di Reti di Petri limitata è quando k = 1; in questo caso la rete si dice *binaria* (o safe). Le PN binarie sono molto importanti per modellare per esempio delle macchine (in lavorazione o libera), un buffer (vuoto o pieno) e altri strumenti che hanno le condizioni logiche zero o uno. Risulterà un caso particolare anche quando vedremo il metodo di traduzione da PN a codice per PLC, distinguendo il caso di una rete binaria o no.

Un'altra proprietà è la vivezza, la quale è quella proprietà tale per cui in una rete c'è la possibilità di un'evoluzione futura [6].

Definizione (Transizione viva)
Una transizione t si dice viva se e solo se per ogni marcatura M raggiungibile dalla marcatura iniziale esiste una marcatura M∗ raggiungibile da essa tale che t è abilitata in M∗.

Definizione (Marcatura viva)
Una marcatura M si dice viva se e solo se per ogni transizione t esiste una marcatura M∗ raggiungibile da M tale che t è abilitata in M∗.

Definizione (Rete viva)
Una rete si dice viva se e solo se tutte le sue transizioni sono vive. Una rete si dice viva se e solo se tutte le marcature raggiungibili dalla marcatura iniziale sono vive.
Possiamo dire che in una rete viva tutte le transizioni possono scattare infinite volte qualunque sia la marcatura iniziale di partenza.

Definizione (Marcatura morta)
Una marcatura M si dice morta se e solo se nessuna transizione è abilitata in M.
Reversibilità, limitatezza e vivezza sono proprietà indipendenti.

Oltre alla rappresentazione grafica, le reti di Petri possono essere rappresentate in moto matematico; questa rappresentazione si dice *matriciale.* Così facendo, possiamo descrivere l'evoluzione di una PN sottoforma di equazioni di stato [12].

Capitolo 3
PLC E LINGUAGGI DI PROGRAMMAZIONE

In questo capitolo ci soffermeremo ad analizzare il PLC e lo Standard IEC 61131 che lo regolarizza; parleremo poi di due dei cinque linguaggi specificati da esso (SFC e LD) e delle loro equivalenze. SFC e LD saranno i due linguaggi utilizzati per le metodologie di traduzione trattate.

3.1 *PLC*

Il controllore logico programmabile (PLC – dall'inglese Programmable Logic Controller) è un dispositivo industriale che ha il compito di gestione e controllo dei processi logico-sequenziali; è nato principalmente per far eseguire in maniera del tutto automatica i processi di lavorazione industriale [2], [3].

Uno dei motivi del successo che i PLC hanno avuto nel corso degli anni, è la facilità di programmazione tramite linguaggi grafici, più semplici rispetto a quelli testuali.

Già nei secoli precedenti al nostro, c'erano già stati dei primi meccanismi automatici, tra cui l'orologio a pendolo, il regolatore di Watt o l'orologio ad acqua; poi, grazie all'introduzione dell'elettricità, furono inventati i primi dispositivi elettronici e meccanici, come relè. Tutto questo però aveva degli svantaggi, tra cui la poca flessibilità: non erano

molto veloci, erano costosi e non erano riconfigurabili, aspetto importantissimo nei PLC attuali. La rivoluzione elettronica risolse questi problemi: negli anni 60 aumentò la richiesta dei grandi utilizzatori di automazione industriale dello sviluppo di un sistema di controllo-logico sequenziale programmabile.

Molti autori fissarono la nascita del primo PLC nel 1968, anno in cui la General Motors (azienda che era la maggior utilizzatrice di impianti industriali) negli USA specificò le caratteristiche per una nuova generazione di controllori; le caratteristiche principali erano [3]:

- Permettere una facile manutenzione di codice; la modularità permetteva questo;
- Occupare poco spazio;
- Basso costo;
- Facilmente programmabili;
- Robustezza, per far sì che potevano lavorare in ambienti industriali, quindi da bassissime temperature ad altissime, in presenza di vibrazioni, onde e interferenza elettromagnetica;
- A memoria espandibile;
- Comunicazione con altri sistemi.

Il primo PLC fu introdotto da Allen-Bradley nei primi anni ’70, basato su microprocessore. Allo stato attuale, i PLC sono multiprocessore e gestiscono funzioni e compiti molto complessi [2], [3].

Si può affermare che il PLC fu pensato come un componente che si interfaccia in modo semplice con il mondo esterno e con gli utenti. Infatti, come vedremo più avanti, il PLC è dotato di un Sistema Operativo e con diverse modalità di programmazione.

Nel 1993, il Comitato Elettrotecnico Internazionale (IEC) ha emanato una normativa Standard – IEC 61131 – la cui parte 1 regola la struttura hardware e software del PLC.

Dallo Standard IEC 61131 - 1, *il PLC è un sistema elettronico a funzionamento digitale, destinato all'uso in ambito industriale, che utilizza una memoria programmabile per l'archiviazione interna di istruzioni orientate all'utilizzazione per l'implementazione di funzioni specifiche, come quelle logiche, di sequenziamento, di temporizzazione, di conteggio e di calcolo aritmetico, e per controllare, mediante ingressi e uscite sia digitali che analogici, vari tipi di macchine e processi.*

Prima dell'introduzione dello Standard IEC 61131, ogni costruttore di PLC sviluppava un suo linguaggio per programmarlo. Questo risultava un problema, in quanto rendeva molto complicato il passaggio di un programma da una macchina all'altra; questo problema venne risolto allora dallo Standard, definendo i cinque linguaggi di programmazione di cui parleremo nel capitolo successivo.

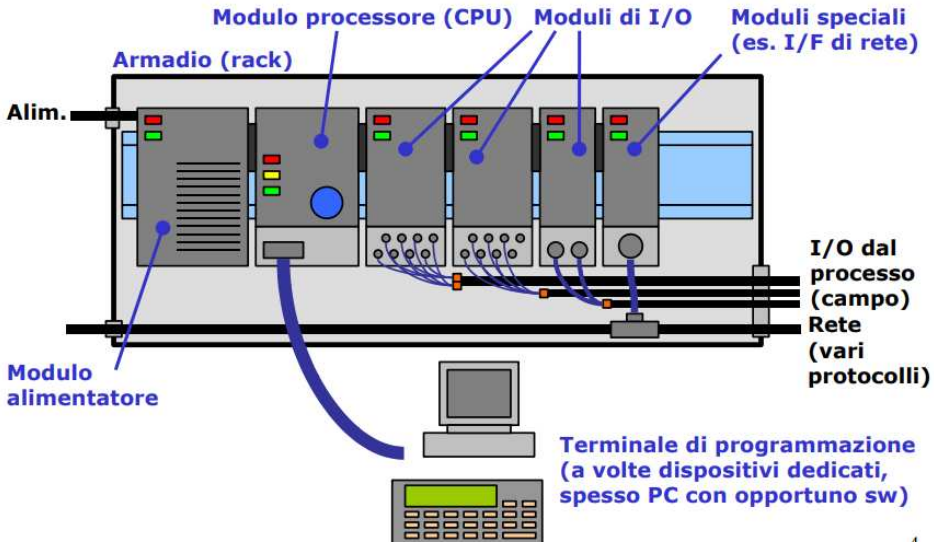


Figura 7 Struttura di un PLC – [22]

I componenti chiave di un PLC sono cinque [2]:

- Armadio;
- Modulo processore;
- Moduli I/O;
- Alimentatore;
- Terminale di programmazione.

1. Armadio: l'Armadio (o Castello o Rack) contiene tutti gli altri moduli di cui abbiamo detto, e ne assicura la loro connessione elettrica, meccanica, e la schermatura. È di forma rettangolare con un'apertura su un solo lato per l'introduzione dei moduli, e nel retro è presente un circuito stampato con vari connettori per garantire i collegamenti elettrici. È metallico e per ragioni di sicurezza è connesso a terra.
2. Modulo processore: è il cuore del PLC: contiene uno o più microprocessori che eseguono i programmi del sistema operativo e quelli del programmatore; controlla tutte le operazioni attraverso le istruzioni contenute all'interno della memoria. Il suo funzionamento prevede un ciclo:
   - Aggiornamento dell'area di memoria con i valori provenienti dagli ingressi fisici;
   - Esecuzione del programma utente;
   - Esecuzione dei programmi di gestione del sistema;
   - Scrittura sulle uscite fisiche.

   Il ciclo appena descritto prende il nome di "Ciclo a copia massiva d*egli ingressi e delle uscite".*

   La parte della lettura sugli ingressi e della scrittura sulle uscite viene gestita completamente dal Sistema Operativo.

Importante è la velocità di elaborazione del modulo, che è definita dal *tempo di scansione*, cioè quel tempo che passa tra due attivazioni successive della stessa porzione del programma applicativo, a cui appartiene il tempo di aggiornamento degli ingressi e uscite. Da indicazione del tempo necessario a effettuare le fasi del ciclo descritto precedentemente. Quindi, più ingressi e uscite ci saranno, maggiore sarà questo tempo. Il Tempo di scansione è definito in millisecondi per Kiloword di programma (1 Kiloword equivale a 1024 parole). Il tempo di scansione è diverso dal *tempo di risposta*, che è l'intervallo massimo di tempo che passa tra la rilevazione di un evento e l'esecuzione dell'azione associata [2].

Per quanto riguarda il Sistema Operativo del PLC, è formato da una serie di programmi che sono memorizzati in modo permanente; questi programmi consentono il controllo delle attività, l'elaborazione dei programmi utente. Un PLC può lavorare in tre modalità operative:

- Esecuzione: vengono eseguiti i programmi utente aggiornando ingressi e uscite;
- Validazione: vengono eseguiti i programmi, ma l'aggiornamento delle uscite è disabilitato (questa modalità è riservata a verificare la correttezza del codice);
- Programmazione: è utilizzata per caricare il codice nella memoria del PLC. (A sua volta, la Memoria è suddivisa in diverse aree ROM, RAM).

3. Moduli I/O: una delle specifiche introdotte dalla General Motors era la possibilità di connessione tra controllore e sensori/attuatori.

Il PLC è costituito da una serie di moduli che permettono tale comunicazione. Questi moduli allora fungono da tramite tra la logica interna del PLC e i segnali esterni: devono realizzare l'adattamento tra i livelli di tensione e corrente del PLC e tra tensioni e correnti utilizzate per la trasmissione dei segnali. Il PLC opera con una logica elettronica a bassa potenza e deve essere interconnesso a segnali I/O con livelli di tensione e corrente molto elevati rispetto a lui stesso. Questa è una caratteristica fondamentale del PLC.

Il PLC completo prevede la possibilità di ospitare un certo numero di moduli, in base alla dimensione dell'armadio e in posizioni precise.

4. Alimentatore: deve fornire l'alimentazione necessaria al corretto funzionamento di tutti i moduli che compongono il PLC.
5. Terminale di programmazione: il terminale di programmazione permette la connessione di una tastiera o display per controllare visivamente il programma.

3.2 *Standard IEC 61131-3*

Perché ci fu il bisogno di introdurre uno standard per il PLC? Quando vennero introdotti i primi controllori logico programmabili, non c'era in vigore nessuno standard per quanto riguarda la programmazione; quindi, ogni azienda poteva fare e programmare a modo proprio. Questo però risultava un problema nel momento in cui si doveva trasferire un progetto su una macchina differente; questo è il motivo per cui venne introdotto lo STANDARD IEC 61131 [3].

La norma doveva quindi definire degli standard a cui ogni sistema doveva attenersi per garantire la corretta implementazione di codici su sistemi diversi, al fine di evitare l'incompatibilità tra macchine diverse. Lo Standard è strutturato su sette punti:

1. Panoramica generale;
2. Hardware;
3. *LINGUAGGI DI PROGRAMMAZIONE;*
4. Guida per l'utente;
5. Comunicazione;
6. Logica;
7. Guida per l'applicazione.

In questo elaborato verrà analizzata la terza parte, quella dei linguaggi di programmazione. La terza parte dello standard è a sua volta suddivisa in cinque linguaggi di programmazione, divisi a loro volta in linguaggi testuali e grafici:

- Testuali:
    - Istruction list 'IL': è un linguaggio di basso livello, simile all'assembly;

    - Structured Text ‘ST’: è un linguaggio di alto livello, simile al Pascal, che offre la possibilità di sviluppare sistemi più complessi con meno istruzioni rispetto al ’IL’;
- Grafici:
    - Ladder Diagram ‘LD’: è un linguaggio basato sui sistemi a relè, dove la logica booleana è espressa con bobine e contatti.
    - Function Block Diagram ‘FBD’: è un linguaggio a blocchi, in cui ognuno di essi esegue algoritmi per elaborare gli ingressi e valorizzare le uscite;
    - Sequential Functional Chart ‘SFC’: è un linguaggio nato per suddividere le attività di controllo in semplici passaggi, per descrivere il comportamento il più simile al sistema che si vuole ottenere;

A causa del fatto che, prima della comparsa dei PLC, la logica booleana (circuiti) era la via principale del controllo industriale e anche grazie alla semplicità e familiarità con i precedenti sistemi di controllo, il LD divenne il linguaggio più utilizzato. Ma LD non era però stato progettato per descrivere il controllo sequenziale dei sistemi. A sua volta, il linguaggio più indicato per descrivere il comportamento logico-sequenziale di un sistema di automazione industriale è l’SFC.

Nel 1975 fu istituita in Francia una commissione per cercare una soluzione alla realizzazione di sistemi più complessi in ambito di automazione industriale. La soluzione fu il GRAPHe de Cordination Etaper Transitions (GRAFECET), che nel 1987 venne usato come standard internazionale dal Comitato Elettrotecnico Internazionale, e poi

successivamente introdotto nello STANDARD IEC sottoforma di SFC [3].

In questo elaborato si è focalizzata l'attenzione in particolare in due dei cinque linguaggi:

- LD: la programmazione è basata sul controllo di segnali I/O. È il linguaggio più utilizzato in ambito industriale ed è supportato da ogni tipo di PLC, e si può implementare su ogni macchina.
- SFC: la programmazione è basata sugli stati. È un linguaggio più intuitivo rispetto al LD, infatti la sua struttura è molto simile al pensare dell'uomo.

3.3 *Ladder Diagram – LD*

Il Linguaggio a Contatti (Ladder Diagram dall'inglese, o Diagramma a Scala per la sua forma) è il più diffuso linguaggio di programmazione per PLC; è anche il più antico: si può definire infatti il LD come la traduzione informatica dei relè che venivano utilizzati per controllori logici sequenziali di una volta, prima dell'introduzione del PLC.

Essendo stato il primo linguaggio introdotto, si è voluto rimanere sulla stessa lunghezza d'onda dei relè, permettendo un passaggio il più lineare possibile; infatti, la simbologia del Ladder richiama una simbologia puramente elettrica. Esso rappresenta infatti dei circuiti sottoforma di equazioni booleane, in cui operatori sono i contatti, mentre il risultato dell'equazione è rappresentato sottoforma di bobina [3].

3.3.1 *Rappresentazione grafica*

Dal nome "Diagramma a scala" è facilmente immaginabile la forma che ha un programma scritto in codice Ladder. Si divide il codice in due parti:

1. *Montanti*: sono due, rappresentano l'alimentazione e hanno al loro interno le equazioni booleane con i vari contatti;
2. *Pioli*, in numero variabile, che sono invece le espressioni booleane vere e proprie.

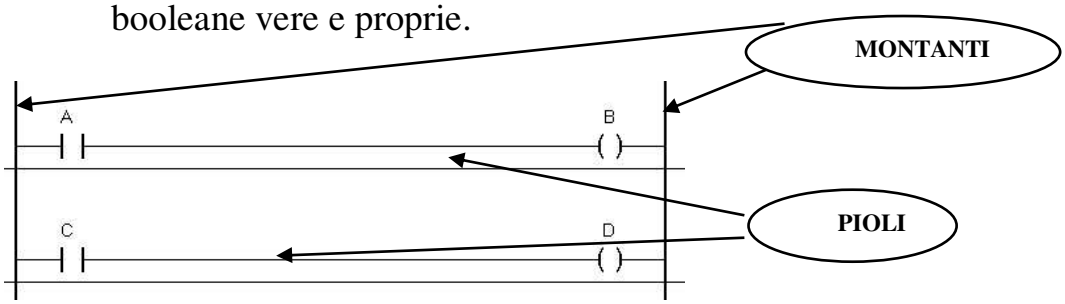
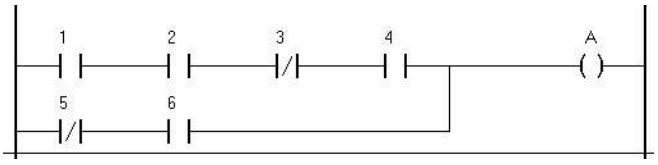


Figura 8 Struttura codice LD – Twincat Beckhoff

Il montante rappresenta l'alimentazione: nel linguaggio Ladder Diagram, la corrente va solamente da sinistra a destra; questo di fatto definisce una *priorità standard* (diversamente dal linguaggio SFC).



Per capire la priorità viene riportato un piccolo esempio: la bobina A può essere attivata dalle sequenze: 1 – 2 – 3 – 4 o 5 – 6, ma non dalla sequenza 5 – 6 – 4 o 1 – 5 – 6 – 4.

I pioli invece rappresentano il corpo dell'equazione booleana tramite dei contatti; quest'ultimi permettono il passaggio della corrente in baso al loro stato (se verificati o meno) che andranno ad alimentare una bobina di uscita. La linea orizzontale del codice viene anche detta "rung".

3.3.2 *Elementi del Ladder*

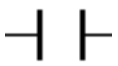
Si introducono gli elementi base che costituiscono un codice, senza l'introduzione di temporizzatori, poiché si vuole un sistema che è guidato dagli eventi.

Gli elementi che costituiscono un programma scritto il LD sono principalmente due:

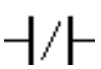
- *CONTATTI*: rappresentano le varie condizioni logiche da verificare per attuare l'uscita corrispondente; ci sono contatti normalmente aperti e normalmente chiusi;
- *BOBINE*: è l'uscita corrispondente ai contatti;

In seguito, verranno analizzati più nel dettaglio.

- *CONTATTO NORMALMENTE APERTO*: essendo una logica booleana, se il suo bit associato vale 1 (ON), il sistema operativo lo chiuderà assicurando che la corrente scorra. Invece se il bit associato vale 0, rimane aperto, e il passaggio di corrente viene interrotto;



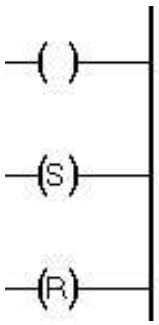
- *CONTATTO NORMALMENTE CHIUSO*: è il duale del contatto aperto; se il bit associato vale 0 (OFF), cioè verificato, il sistema operativo lo chiude e ci sarà il passaggio di corrente. Invece, se il bit vale 1, (ON), il contatto è come se fosse aperto;



- *BOBINA*: va inserita in fondo a destra dei vari pioli: se tutte le condizioni logiche (contatti aperti e chiusi) ad essa associati sono verificate (tutte ON), viene attuata. La bobina equivale all'output.

Nella scrittura di un codice, un ruolo centrale viene ricoperto dai *qualificatori:*

- *COIL***:** risulta attivo finché la condizione associata risulta vera;
- *SET***:** viene utilizzato quando si vuole una continuità dell'output, anche quando le condizioni associate non sono più vere;
- *RESET***:** serve per stoppare l'azione iniziata dal Set.



Come si vedrà anche nel linguaggio SFC, il ruolo del qualificatore è centrale: si prenda l'esempio del nastro trasportatore presente nel laboratorio universitario. Per il suo corretto funzionamento, c'è bisogno del S/R, in quanto con il solo coil l'azione non durerebbe nel tempo, cosa che un nastro ha bisogno di avere.

A differenza dell'SFC, il LD è un linguaggio di basso livello, quindi ai fini della traduzione da rete di Petri a codice equivalente per PLC, risulterà un'operazione più complessa.

3.4 *Sequential Function Chart – SFC*

L'SFC (Sequential Function Chart) è un linguaggio di programmazione definito dallo Standard IEC 61131-3; descrive il funzionamento sequenziale dei sistemi a eventi discreti e di macchine e processi. Infatti, la maggior parte di esse hanno un funzionamento di tipo sequenziale, cioè svolgono compiti e operazioni collegate tra loro tramite delle transizioni. Descrivere il funzionamento significa descrivere il comportamento desiderato del sistema [3].
L'SFC, anche se definito dallo Standard come un linguaggio di programmazione per PLC a tutti gli effetti, in realtà non può essere considerato un linguaggio vero e proprio [3], in quanto ha bisogno del supporto degli altri linguaggi per essere implementato (le condizioni degli step/transizioni possono essere implementate tramite il LD o ST per esempio).

3.4.1 *Vantaggi nell'utilizzo dell'SFC*

- È di facile comprensione: il progetto passa dalla richiesta del cliente, alle tecniche di traduzione e alla realizzazione, in cui ci possono lavorare molte persone diverse; per questo serve un linguaggio semplice e intuitivo, comprensibile da tutti;

- Appoggia l'approccio top-down: cioè il funzionamento del macchinario viene scomposto in sottoprogrammi, riducendo così la complessità. Non a caso è un uno dei linguaggi più adatti per implementare su PLC il modello delle reti di Petri;
- Facilita la visualizzazione del funzionamento sequenziale della macchina [3];

3.4.2 *Elementi dell'SFC*

Una struttura base di codice dell'SFC è come quella in figura. Gli elementi base sono:

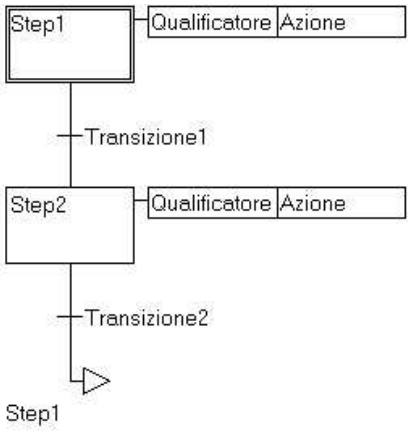


Figura 9 Struttura SFC - Twincat Beckhoff

1. *STEP* (o fase, passo, stato): descrivendo il comportamento di un sistema a eventi, lo STEP per essere attivato ha bisogno di un'azione ad esso associata (evento). L'accadimento di questo evento genera il passaggio allo step successivo; nel corso dell'evoluzione esso può essere attivo/inattivo. Si indica con il doppio rettangolo lo stato iniziale del sistema, e il proprio nome deve essere univoco per tutti gli altri stati;
2. *AZIONE*: è associata allo step, cioè sono condizioni che vengono eseguite nel momento in cui lo step diventa attivo. Graficamente essa viene rappresentata tramite un rettangolino associato allo step al cui interno ci sono le azioni;

3. *TRANSIZIONE*: è il passaggio da uno step all'altro; viene associata sempre a una condizione che deve essere verificata e sono scritte sottoforma di operazioni booleane. In SFC si rappresenta come trattino che taglia l'arco orientato;
4. *ARCO ORIENTATO*: esso collega due step tra loro e sancisce il passaggio da uno stato all'altro, indicando l'avvenuta evoluzione del sistema. In caso non ci fosse lo step dopo una transizione, si usa la funzione "jump to" e si scrive il nome dello step su cui si deve ritornare. Va dall'alto al basso;

Un ruolo fondamentale nella stesura di un codice SFC, come detto anche per il LD, è dato dai *qualificatori;* ce ne sono diversi tipi [3]:

- *N*: indica l'azione normale, cioè è eseguita finché è attivo lo stato. Si può anche omettere essendo il qualificatore standard;
- *P:* indica la pulsazione, cioè l'azione è eseguita una sola volta dal momento in cui lo stato diventa attivo;
- *S:* indica l'azione di Set, cioè l'azione rimane attiva finché non viene eseguita la stessa azione ma con il qualificatore R di Reset;
- *R*: complementare al Set, finisce l'azione iniziata dal Set;
- *L*: indica il time limited, cioè l'azione finisce dopo un certo tempo prestabilito, e in caso l'azione esce dallo stato, si interrompe;
- *D*: indica il time delayed, cioè l'azione è eseguita dopo un certo tempo;
- *SD*: stored/delayed, cioè l'azione è eseguita con un set ma dopo un certo tempo;

- *DS*: viceversa all'SD, l'azione è eseguita con un set e dura un certo tempo dall'attivazione;
- *SL*: stored/time limited, cioè l'azione è sempre eseguita come un set e poi viene terminata dopo un certo tempo con il reset;

Come nel LD, viene fornito un piccolo esempio per capire l'importanza dei qualificatori: si supponga di avere a disposizione la stazione giostra presente nel laboratorio universitario; è suddivisa in più postazioni, tra cui un trapano; supponiamo debba fare un foro sui pezzi.
Il trapano ha due sensori, trapano alto che sta a indicare che è in posizione di riposo, trapano basso che è in lavorazione. Per l'azione di foratura non basta il solo impulso e nemmeno l'azione normale, in quanto il trapano appena arrivato in posizione di lavorazione tornerebbe subito in quella di riposo, non completando correttamente la lavorazione; serve tenere il trapano in lavorazione per un certo intervallo di tempo, affinché completi l'operazione. Verrà allora utilizzato prima il qualificatore Set, poi una volta completata, si attuerà il Reset.

3.4.3 *Regole di Evoluzione*

Nella scrittura del codice, ci si deve attenere a delle regole di evoluzione, che sono principalmente molto simili a quelle introdotte nella rete di Petri: non a caso è l'SFC è il linguaggio più adatto per implementare su PLC un modello di rete [3]:

- Step-transizione-step: tra due step collegati, ci deve sempre essere una transizione in mezzo, e viceversa, tra due transizioni ci deve sempre essere uno step. Cosa che avviene identicamente nelle PN tra posti e transizioni;

- Per superare una transizione, si devono sempre verificare due condizioni (le stesse nelle PN):
    - la condizione della transizione deve essere vera;
    - tutti gli step a monte della transizione devono essere attivi;

  In questo caso si dice che la transizione è abilitata allo scatto;
- Se una transizione scatta, cioè le sue condizioni sono tutte vere, allora tutte gli step a monte della transizione vengono disattivati e tutti quelli a valle attivati;

Una volta scritto il codice in SFC, viene riportato su PLC: quest'ultimo ha un *funzionamento ciclico [3]*.

L'esecuzione dell'SFC richiede una serie di attività da eseguire in un certo ordine in ciascun ciclo di scansione, definite dallo Standard. Una regola fondamentale dell'evoluzione è che uno step non può essere attivato disattivato nello stesso ciclo di scansione.

Lo standard definisce le seguenti attività in ordine di esecuzione:

1. Determinare l'insieme degli step attivi;
2. Valutare tutte le transizioni associate agli step attivi;
3. Eseguire le azioni;
4. Azioni attive;
5. Disattivare gli step attivi che precedono condizioni di transizione vere e attivare i corrispondenti step successivi;
6. aggiornare le condizioni di attività delle azioni;

Ai fini della traduzione da rete di Petri a linguaggio equivalente SFC, risulta più istantanea in quanto l'SFC è un linguaggio di alto livello, diversamente dal LD.

3.5 *SFC e LD: confronto*

Avendo analizzato i soli due linguaggi SFC e LD, quali sono i vantaggi e svantaggi?

Sicuramente l'SFC è visivamente più intuitivo, quindi più semplice da essere utilizzato. Essendo un diagramma a blocchi, modulare, il codice è di facile manutenzione: in caso appunto ci sia il bisogno di una modifica, si andrebbe a modificare solo il blocco corrispondente, facilmente rintracciabile. Riesce visivamente a mostrare le situazioni come divergenze, convergenze, parallelismi (strutture che verranno analizzate in un paragrafo del capitolo 4).

D'altra parte, l'SFC non è supportato da tutti i PLC, e come si è visto nello Standard IEC 61131-3, non è propriamente un linguaggio di programmazione, ma al suo interno vengono utilizzati altri linguaggi.

Per quanto riguarda il linguaggio LD, invece, essendo stato il primo linguaggio per PLC in circolazione (deriva dai relè), è supportato da tutti i PLC, e rispetto all'SFC è un linguaggio di programmazione di livello più basso.

3.5.1 *Equivalenze tra SFC e LD*

Nel paragrafo precedente, si è detto che il linguaggio SFC potrebbe essere non supportato da tutti i PLC, a differenza del LD. Allora può essere utile cercare una procedura che permetta la traduzione di un codice in SFC nel suo equivalente LD.

Nel presente lavoro non ci soffermeremo su questo aspetto. Al livello esemplificativo si riportano nel seguito le traduzioni da SFC a LD di transizione e step.

Quello mostrato in figura 10 sta a significare che una generica transizione dall'SFC viene trasformata in un contatto normalmente aperto o chiuso nel LD.

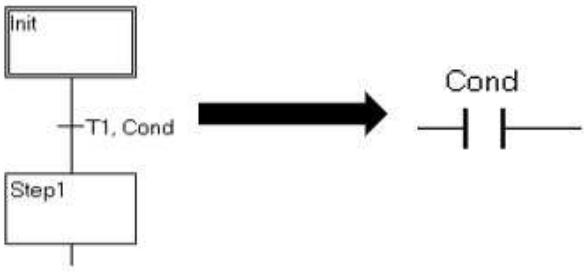


Figura 10 Traduzione di una transizione da SFC in LD

Invece, per la traduzione di uno step da SFC a LD:

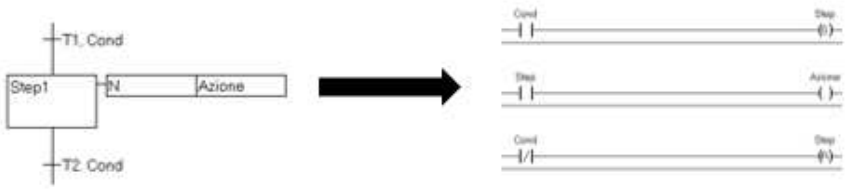


Figura 11 Traduzione di uno step da SFC in LD

Capitolo 4
METODOLOGIA DI TRADUZIONE

L'obiettivo di questa tesi è quello di studiare una metodologia di implementazione di codice per PLC partendo dal modello definito dalla rete di Petri. Per metodo di traduzione si intende un qualsiasi metodo che, applicato ad un modello matematico, consente in modo automatico di generare l'equivalente codice per il PLC. Come visto in un paragrafo del capitolo 2, le PN sono uno strumento molto efficace per modellare i Sistemi ad eventi discreti poiché sono in grado di rappresentare differenti situazioni.

Dopo una approfondita analisi della letteratura, due metodi di traduzione sono risultati di maggiore interesse:

- Un metodo riguardante il codice SFC, chiamato "Tecnica di mappatura a uno a uno";
- Un metodo riguardante il codice LD;

La metodologia della mappatura a uno a uno ha come obiettivo quello di preservare la corrispondenza diretta tra la rete di Petri e la sua implementazione su PLC.

In questo capitolo verranno analizzati:

- Tecnica di mappatura a uno a uno da PN a SFC;
    - Caso di PN binaria;
    - Caso di PN non binaria;
- Metodo di traduzione da PN a LD:
    - Mappatura a uno a uno;

4.1 *Traduzione da PN a SFC*

Lo schema di trasformazione è il seguente [14]:

Figura 12 Schema traduzione da PN a SFC

Di seguito verranno spiegati i vari step dello schema.

1. *SPECIFICATORE DI INFORMAZIONI*: in questa prima fase va descritto in modo generale il sistema sottoposto a studio; vanno definite le caratteristiche che deve avere e le operazioni che deve svolgere il sistema;
2. *MODELLAZIONE UTILIZZANDO LE RETI DI PETRI*: in questa parte va modellato il sistema definito dallo *specificatore di informazioni*, tramite il formalismo delle PN.
3. *ANALISI PROPRIETA'*: il terzo step è quello di analizzare le varie proprietà di una rete di Petri; in particolare, l'analisi è volta al rilevamento di eventuali situazioni di deadlock, stati in cui la rete rimane bloccata. Altre caratteristiche importanti sono la reversibilità, la raggiungibilità, la vivezza.
4. *VERIFICA BINARIETA'*: questo punto è l'ultimo prima della generazione del codice, e vanno distinte due situazioni differenti [3]:
    a. Se la rete risulta binaria, la traduzione è istantanea: ad ogni posto della PN equivale uno Step dell'SFC; una rete di Petri si dice binaria (o safe) se nel corso della sua evoluzione tutti i posti andranno a contenere al massimo un token (gettone); questa rete viene detta limitata per k = 1;
    b. Se la rete non risulta binaria, si deve calcolare il Grafo di Raggiungibilità; tramite quest'ultimo poi saremo in grado di definire il codice SFC equivalente alla PN; (in una PN non binaria i posti potranno contenere anche più di un token);
5. *GENERAZIONE CODICE*: L'ultimo punto è quello della stesura del codice per PLC utilizzando i linguaggi definiti dallo Standard IEC 61131-3;

Verrà applicato il metodo proposto a entrambe le tipologie di PN, binaria e non binaria.

Prima di distinguere i due casi, si deve introdurre la struttura del "Grafo di Raggiungibilità" che servirà poi per la verifica della binarietà della rete.

Per capire se una PN è binaria o meno, si è utilizzato il "grafo di raggiungibilità" e il conseguente insieme di raggiungibilità.

- ***Grafo di Raggiungibilità*:** è uno strumento grafico che permette di analizzare e visualizzare tutte le possibili marcature raggiungibili dalla rete a partire da una marcatura iniziale. L'insieme delle marcature raggiungibili viene detto "insieme di raggiungibilità". Un caso particolare è quello in cui la rete contiene posti che accumulano 'token' in numero crescente ed illimitato, cioè una rete illimitata. In questo caso, non è possibile calcolare il grafo di raggiungibilità in quanto infinito. In questi casi può essere conveniente ricorrere al grafo di copertura che invece risulta finito. Nel grafo di copertura, la marcatura dei posti che nell'evoluzione della PN tendono ad acquistare un numero infinito di token, viene rappresentata col simbolo 'ω'; un grafo di questo tipo viene detto "grafo di copertura" [3], [12].

4.1.1 *Il caso di PN binaria*

Nel caso in cui si ha una PN Binaria, utilizzando la tecnica di mappatura a uno a uno, è possibile trovare una corrispondenza *a uno a uno* tra i posti della PN e gli step del codice SFC.

Se un posto nella rete contiene un token, l'equivalente step nel codice SFC risulterà attivo, e le transizioni associate, se verificate, potranno scattare. Caso contrario, se un posto non contiene un token, l'equivalente step risulterà disattivato in attesa di essere poi attivato.

Verrà analizzato un piccolo esempio di PN binaria a cui verrà applicata la metodologia (lo stesso esempio verrà fatto anche nel caso di una PN non binaria). Questo è l'esempio del client/server (o produttore/consumatore) con buffer di richiesta limitato a un posto [3].

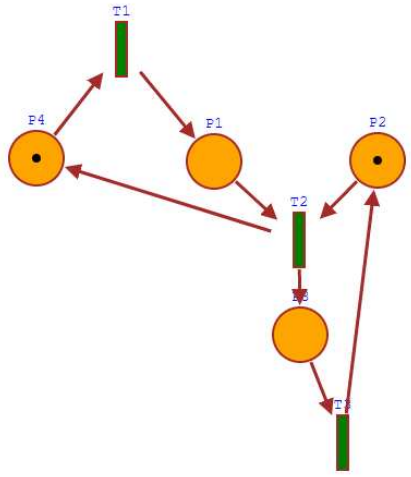


Figura 13 PN binaria

I posti della PN si possono raggruppare in due blocchi: uno che rappresenta la produzione di un servizio, mentre l'altro che rappresenta l'acquisizione del servizio. Analizzando la struttura tramite il "Grafo di Raggiungibilità", si potrà verificare che la PN è binaria.

Partendo dalla marcatura iniziale [0101], e facendo scattare le transizioni superabili, troviamo il grafo di raggiungibilità.

Introduciamo una *simbologia di rappresentazione compatta*:

In questo caso la marcatura iniziale è: [0101]; piuttosto che scrivere i token di ogni posto, si può scrivere il numero di posto in cui è presente il gettone. Per esempio:

- La marcatura [0101] diventa 2,4;
- La marcatura [1100] diventa 1,2;
- Ecc per le altre marcature.

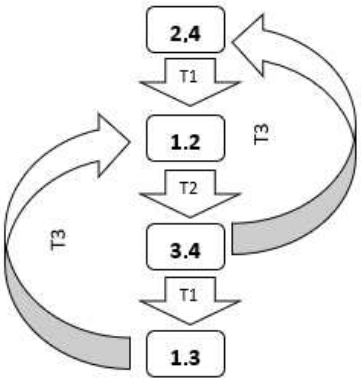


Figura 14 Grafo di Raggiungibilità della PN binaria

L'insieme di raggiungibilità della rete è:

$$R(N) = \{[0101], [1100], [0011], [1010]\} =$$
$$= \{[2,4], [1,2], [3,4], [1,3]\}$$

Si ha al massimo un token per ogni posto, quindi PN binaria.

Si studiano le proprietà: la rete risulta limitata, safe e reversibile; risulta anche viva [3].

Infine, essendo una PN binaria, l'equivalente SFC, utilizzando la tecnica di "mappatura a uno a uno" introdotta, sarà graficamente simile alla PN:

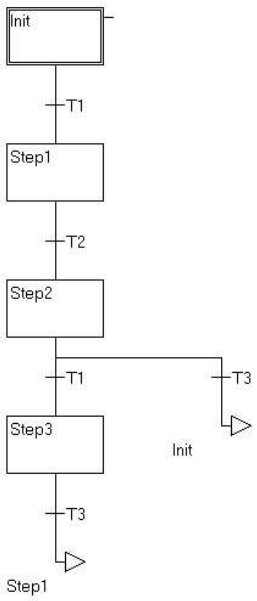
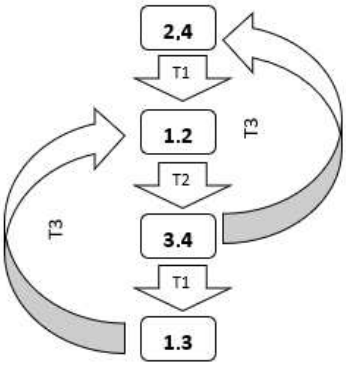


Figura 15 Codice SFC della PN binaria

Mettendolo in relazione alla sua PN, si può notare le equivalenze introdotte dalla metodologia [3].



Le marcature raggiungibili dalla rete sono quattro, esattamente come gli step dell'SFC equivalente.

4.1.2 *Il caso di PN non binaria*

Rispetto al caso precedente, i posti non sono limitati a contenere un solo token nel corso della sua evoluzione.

La differenza rispetto alla PN binaria sta nel fatto che in questo caso non c'è un'esatta corrispondenza a uno a uno come nel caso della binaria; quindi, la struttura equivalente dell'SFC sarà leggermente diversa.

Viene applicata la metodologia allo stesso esempio di cui si è parlato nel caso binario, ma questa volta a capienza del buffer non unitaria [3]:

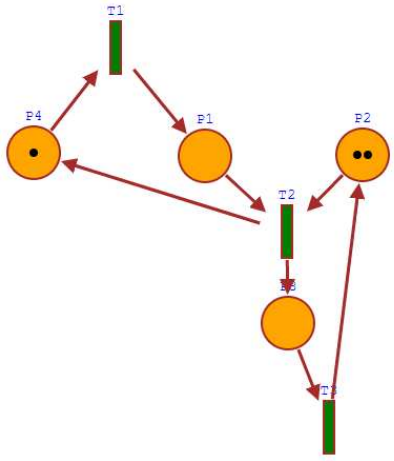


Figura 16 PN non binaria

P2 nella marcatura iniziale contiene già due token, quindi PN non binaria.

Per trovare il codice equivalente SFC si passa attraverso il grafo di raggiungibilità, che si deve calcolare.

Si parte dalla marcatura iniziale: [1 0 2 0], e facendo scattare le transizioni si trova l'insieme di raggiungibilità, come nel caso precedente:

R(N) = {[0 2 0 1], [1 2 0 0], [0 1 1 1], [1 1 1 0], [0 0 2 1], [1 0 2 0]}

In questo caso l'insieme di raggiungibilità ha cardinalità 6.

La PN è non binaria, perché rispetto al caso precedente, nei vettori delle marcature raggiungibili non c'è al massimo un token per ogni posto, ma

in questo caso compare anche ‘2’, indicando la presenza di due token nel posto.

Una volta calcolato il grafo di raggiungibilità, si può applicare la metodologia della mappatura a uno a uno. Rispetto alla PN binaria, la traduzione richiede quindi un passaggio in più; nella PN binaria si poteva passare dalla PN all’ SFC sostituendo ad ogni posto uno step (il grafo serviva solo per verificare che la rete fosse binaria); in questo caso si deve passare per forza tramite il grafo di raggiungibilità.

L’equivalente codice SFC della PN sarà:

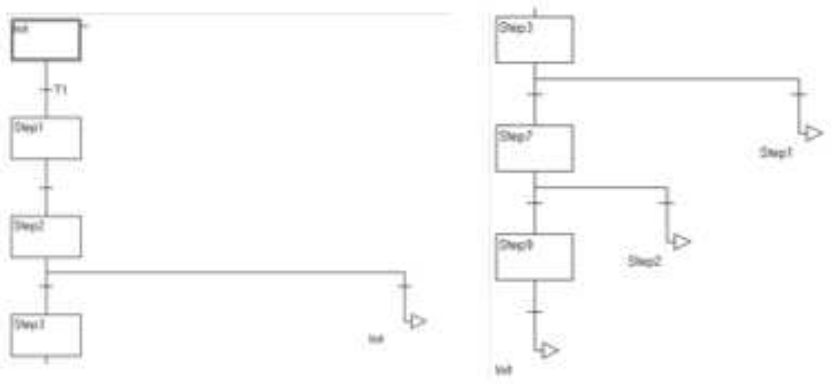


Figura 17 Codice SFC della PN non binaria

Nel caso della PN non binaria, mettendo a confronto il modello della PN con il codice equivalente, non vi è graficamente una corrispondenza posti-step.

(Nella PN si avevano quattro posti, mentre nell’SFC si hanno sei step, diversamente dal caso binario in qui l’equivalenza era istantanea e perfetta).

Nella Capitolo 5 verrà analizzato un esempio in cui viene applicata la tecnica della mappatura a uno a uno ad un impianto di industria chimica.

4.2 *Traduzione da PN a LD*

Nel caso di traduzione da PN a codice equivalente LD, la metodologia risulta più complessa e meno intuitiva rispetto a quella del codice SFC.
Il metodo utilizzato per trascrivere la rete di Petri in codice LD prevede la suddivisione dell'algoritmo in tre parti [15]. Per ogni marcatura:

1. Il primo step è quello di determinare quali transizioni sono abilitate;
2. Il secondo step riguarda l'attivazione delle transizioni, cioè la rimozione dei token dai posti di input e la creazione dei token nei posti di output;
3. Il terzo step è invece quello di attivare le uscite corrispondenti ai posti nel quale è presente il token.

Inoltre, si deve generare un piolo di Ladder aggiuntivo che imposta le condizioni iniziali del sistema; questo blocco corrisponde alla marcatura iniziale della rete di Petri.

Il processo di trascrizione è organizzato nelle seguenti *fasi* [15]:

- *Creare variabili ausiliarie***:** queste variabili rappresentano lo stato degli elementi della PN (posti e transizioni). Così facendo, ogni posto della rete di Petri ha due variabili associate:
    - la variabile di output del programma, rappresentata dal nome del posto;
    - la variabile ausiliaria che rappresenta la presenza di un token su di essa;

La variabile ausiliaria è denominata usando il nome del posto seguito da una parola, per esempio “Locale”. Quindi un generico posto P1 è associato alle due variabili “P1” e “P1Local”.

Lo stesso accade con le transizioni:

- la variabile di input del codice è rappresentata dal nome della transizione “T1”;
- la variabile ausiliaria che rappresenta l'abilitazione della transizione è definita aggiungendo la parola “Local”, “T1Local”.

- *Abilitazione delle transizioni***:** ogni transizione è definita come una bobina ed è associata a un piolo del codice LD. Questa bobina è associata alla variabile ausiliaria "T1Local". Sempre sullo stesso ramo, i posti di ingresso alla transizione analizzata vengono associati a contatti normalmente aperti. Infine, i posti di uscita sono associati a contatti normalmente chiusi in rami successivi.
- *Scatto delle transizioni***:** In questa fase, per ogni transizione vengono creati tanti pioli LD quanti sono gli archi ad essa collegati. Per gli archi di input, il piolo ha a bobina di ripristino (Reset), mentre per gli archi di uscita, il piolo ha una bobina impostata (Set). Ogni bobina è nominata con il nome della variabile ausiliaria associata “P1Local”. Per ogni gradino viene creato un contatto normalmente aperto corrispondente alla transizione e denominato con il nome della variabile ausiliaria “T1Local”.
- *Attivazione delle uscite***:** Un piolo di codice LD è associato a ciascuna variabile di uscita (una per ognuna); questa variabile è

rappresentata da una bobina. Ciascun posto associato alla variabile di uscita è rappresentato da un contatto normalmente aperto e definito con il nome del variabile ausiliaria "P1Local". Se più di un posto è associato ad una variabile di uscita, i contatti vengono inseriti in parallelo.

- *Condizioni iniziali*: le condizioni iniziali del programma sono definite con l'aggiunta di un ulteriore ramo LD. In questo piolo, ogni posto è rappresentato da un contatto normalmente chiuso definito con il nome della variabile ausiliaria associata a quel posto. Così, le bobine in questo ramo si accenderanno solo durante il primo ciclo del PLC quando tutti i contatti sono falsi. Ogni bobina corrisponde a un posto con la marcatura iniziale.

Nel capitolo 5 verrà analizzata questa tecnica di traduzione applicata a un impianto di industria chimica.

4.2.1 *Traduzione istantanea da PN a LD: Mappatura a uno a uno*

Analizzando la tecnica di tradizione da PN a LD del paragrafo precedente, anche in questo la metodologia prevede di cercare corrispondenze tra la struttura della PN e il corpo del codice LD [14], [16], [20].

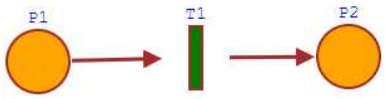
In particolare la corrispondenza tra posti e LD viene rappresentata da un'equazione booleana: i pioli (linea orizzontale) e gli elementi associati, equivalgono a delle *equazioni booleane;* si vogliono trovare allora gli elementi dipendenti e indipendenti tra PN e codice LD: nelle PN, il posto è l'elemento dipendente dell'equazione booleana, mentre nel codice

Ladder l'elemento dipendente è definito dall'output; inoltre, l'elemento indipendente della rete è la transizione, mentre quello del codice LD è il contatto normalmente aperto o normalmente chiuso.
Nella scrittura di un codice LD, i contatti possono rappresentare funzioni logiche booleane:

- I contatti posti in serie rappresentano la funzione booleana AND (nulla se omesso: A&B = AB);
- I contatti in parallelo rappresentano la funzione booleana OR (A+B);

4.2.1.1 Regole di traduzione istantanee

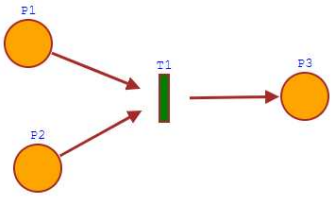
1. Questo è un esempio classico di una sequenza di una PN formata da due posti e una transizione.



La sua traduzione in codice LD è istantanea:

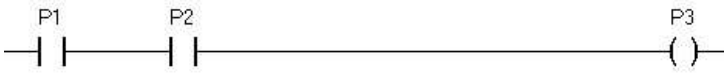
Secondo quanto scritto nelle regole di traduzione, il solo posto in ingresso alla transizione T1 nella PN rappresentato dal posto P1 viene rappresentato a sua volta dal contatto normalmente aperto nel codice LD, che, come detto, rappresenta l'elemento indipendente. Invece l'output, M, rappresenta l'elemento dipendente. L'equazione booleana è semplicemente M = A in quanto c'è un solo ingresso.

2. In questo secondo esempio si ha una PN formata da tre posti e una transizione [14], [16], [20]:



la condizione necessaria affinché la transizione sia superabile, è che i posti 1 e 2 abbiano entrambi un token. La conseguenza sarà il passaggio del token nel posto 3.

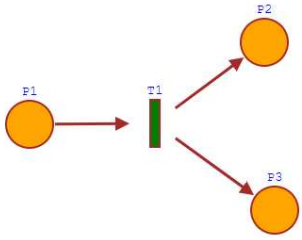
Il Ladder Diagram associato a questa rete di Petri, utilizzando la tecnica di mappatura a uno a uno è molto semplice: i posti 1 e 2 equivalgono a degli ingressi nel LD (come si può vedere dall'arco orientato), mentre il posto 3 rappresenta un'uscita. Allora la sua traduzione da PN a SFC risulta essere:



Si può osservare la corrispondenza a uno a uno tra i posti in ingresso della PN e gli input nel LD. Ugualmente con l'output.

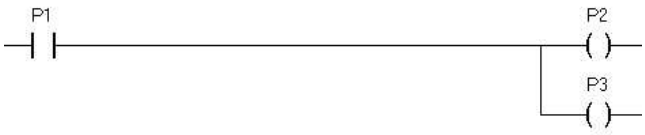
Dal codice LD, i contatti A e B sono messi in serie tra loro perché entrambi necessitano il token e l'attivazione, e la loro equazione booleana sarà legata dall' AND. La funzione booleana che li lega è: C = A & B (o C = AB);

3. Si ha come prima tre posti e una transizione, ma questa volta come input si ha un solo posto, mentre come output se ne hanno due.



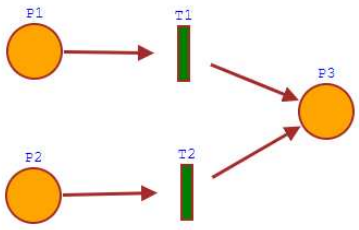
La condizione necessaria affinché la transizione sia superabile è la presenza di un token nel posto di input P1. La conseguenza sarà il passaggio di un token nei posti P2 e P3.

Come nell'esempio precedente, la traduzione in codice LD è molto semplice:



Utilizzando la tecnica di mappatura a uno a uno, a ogni posto di input della PN equivale un contatto di ingresso nel LD: il posto A equivale a un contatto di input, e come output si avranno due bobine corrispondenti ai posti C e D. Quindi se A è verificata, cioè nella PN ha un token, verranno attivati gli output (bobine) C e D. La funzione booleana che li lega è: A = C; A = D; C = D;

4. Si hanno tre posti ma questa volta due transizioni aventi in ingresso posti distinti ma in uscita un posto in comune. Questa PN può equivalere a due processi che evolvono parallelamente, cosa molto comune nei processi di automazione industriale:



La condizione necessaria per l'attivazione del posto P3, cioè che vi sia un token, è che almeno una delle due transizioni siano abilitate allo scatto, e non più necessariamente entrambe come nel primo esempio.

La traduzione al LD di questa PN è la seguente:

Rispetto all'esempio (1) in cui i contatti erano in AND, questa volta dovranno essere messi in OR (+) tra loro, poiché affinché venga attivato l'output C, almeno un input deve essere verificato, però ne basta solo

uno, o A o B. La funzione booleana che li lega è del tipo: C = A or B (C = A + B); C = A; C= B;

4.3 *Strutture di controllo: PN e SFC*

Si ritiene necessario analizzare le strutture di controllo tipiche delle PN e commentare sulla relativa traduzione in SFC. Le strutture di controllo sono costituite da loop, strutture alternative e parallele, sincronizzazione, divergenze, convergenze.

Analizzando tali strutture, si contribuirà al miglioramento della produttività del sistema in analisi. Analizziamo le singole strutture, partendo dalla Rete di Petri e trovando l'equivalente codice SFC [17]:

1. *Sequenza*: una semplice combinazione di posti e transizioni è chiamata sequenza. Dalla sequenza si suddividono tre strutture di controllo:

    a. *Loop*: è un ciclo di sequenza che corrisponde a un controllo di ripetizione, in cui una transizione si ricollega a un passo precedente.

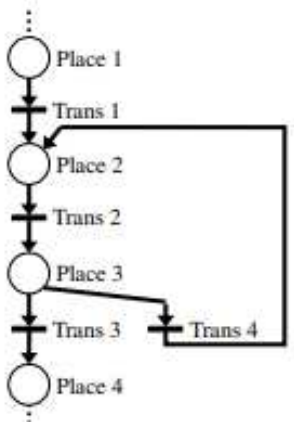
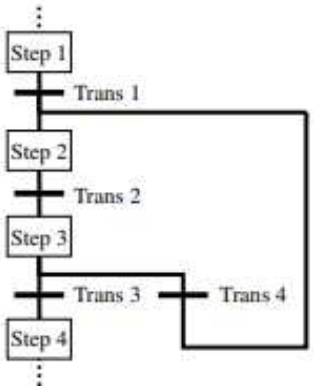


Figura 18 PN loop [17]

La Rete di Petri raffigurante un loop è quella in figura 18, in cui l'abilitazione della Trans 4 è di fatto quella che dà il via al loop.

Viene tradotta la PN utilizzando il metodo di mappatura a uno a uno; l'SFC equivalente risulta essere (figura 19):

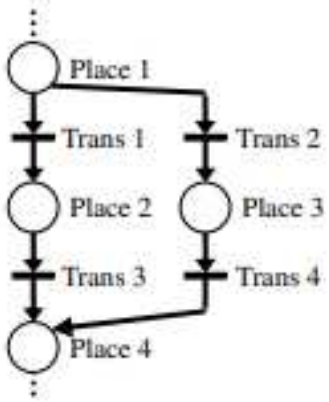


Come si è visto all'inizio di questo capitolo, se la PN è binaria, c'è una corrispondenza a uno a uno tra i posti della PN e gli step dell'SFC, come in questo caso. Mettendo in relazione PN e SFC, posti e step coincidono.

Figura 19 SFC loop [17]

b. *Sequenza alternativa*: Una sequenza alternativa corrisponde ad un controllo della selezione. Si è nel caso in cui viene attivato un ramo dell'SFC piuttosto che un altro, appunto selezione. Il caso in cui entrambe le condizioni sono attive contemporaneamente riveste particolare importanza e verrà analizzato in seguito.

La modellazione della sequenza alternativa tramite la Rete di Petri è (figura 20):



Si può osservare come il posto 1 sia in ingresso a due transizioni: la scelta di scatto di una o dell'altra darà luogo a percorsi che sono tra loro alternativi.

Figura 20 PN sequenza alternativa [17]

L'equivalente codice SFC della rete, utilizzando la mappatura a uno a uno è quella in figura 21.

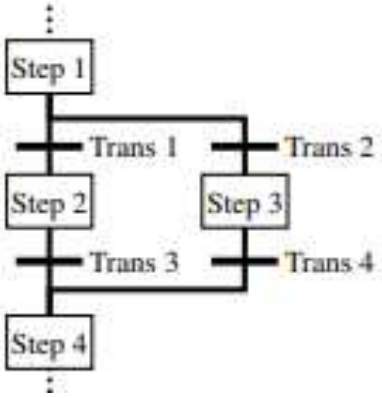


Figura 21 SFC sequenza alternativa [17]

Come si può osservare, ad ogni posto della PN equivale uno step nel codice SFC.

Questo può risultare un caso più complicato quando entrambe le condizioni risultano attive contemporaneamente, e si dovrà gestire la mutua esclusione. Un modo di gestione può essere con l'introduzione della priorità dinamica (di cui si parlerà più avanti), che gestisce la problematica di assegnazione dello scatto delle transizioni in conflitto.

c. *Sequenza simultanea*: Una sequenza simultanea corrisponde a processi che si svolgono in parallelo (figura 22).

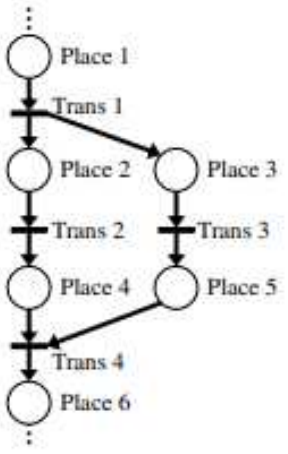
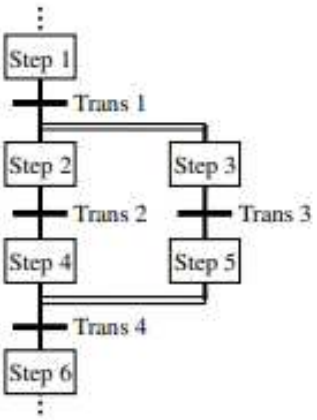


Figura 22 PN sequenza simultanea [17]

Lo scatto di una transizione mette un token in due posti, attivando di fatto due processi.

L'equivalente SFC, utilizzando la tecnica di mappatura a uno a uno è (figura 23):



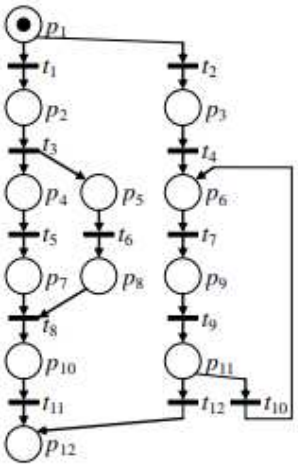
Si può osservare che ad ogni posto della rete equivale uno step del codice SFC.

Le linee delle due transizioni risultano marcate a delimitare in quanto è una sincronizzazione.

Figura 23 SFC sequenza simultanea [17]

4.3.1 *Esempio di PN con all'interno le strutture analizzate*

Verrà analizzato un esempio di PN che racchiude tutte le strutture analizzate precedentemente, con l'equivalente codice SFC utilizzando la tecnica di traduzione a uno a uno [17].



Per prima cosa viene modellata la PN, si calcola il grafo di raggiungibilità si trova il codice SFC.

In questa Rete si possono distinguere tutte e tre le strutture analizzate sopra: il loop, la sequenza simultanea e alternativa.

Figura 24 Modello di PN [17]

Per la conversione in codice SFC, utilizzando la tecnica della mappatura a uno a uno, si eseguono le seguenti fasi:

1. Il sistema va modellato attraverso la PN;
2. Si calcola il Grafo di raggiungibilità per verificarne la binarietà;
3. In caso fosse binaria, la traduzione in SFC è istantanea: ad ogni posto equivale uno step del codice SFC;
4. In caso fosse non binaria, si riposta su PLC il grafo di raggiungibilità e non direttamente la PN, associando ad ogni nuovo step una marcatura raggiunta nel grafo;

Si calcola il grafo di raggiungibilità della rete: il vettore marcatura è di dimensione 12, pari al numero dei posti. La marcatura iniziale sarà

[1 0 0 0 0 0 0 0 0 0 0 0 0] (doppio rettangolo perché iniziale): partendo dalla marcatura iniziale, si fanno scattare tutte le transizioni superabili fino alla conclusione del grafo, poi vengono controllati i numeri che compaiono all'interno del vettore:

- Alternanza di soli 0 e 1 → binaria;
- Altrimenti → non binaria;

Si calcola il grafo di raggiungibilità applicando la simbologia compatta precedentemente introdotta: in questo caso la marcatura iniziale è: [1000000000000] = 1, e così via per tutte le altre marcature.

Il grafo risulta essere allora (figura 25):

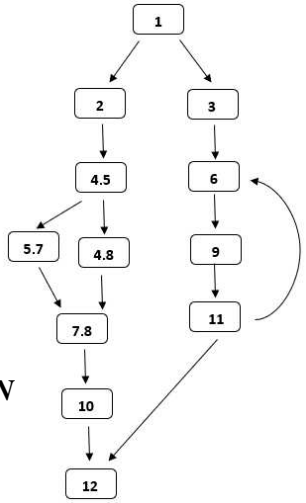


Figura 25 Grafo associato alla PN

L'insieme di raggiungibilità è il seguente:

R(N) = {[1], [2], [3], [4,5], [6], [5,7], [4,8], [9], [7,8], [11], [10], [12]}

Le marcature raggiungibili sono 12, e in tutte c'è la presenza di al massimo 1 token per ogni posto: questo implica che la rete di Petri è binaria.

La tecnica di mappatura a uno a uno, allora, dice che c'è un'esatta corrispondenza a uno a uno tra i posti della rete e gli step del codice SFC equivalente: se nella PN si hanno 12 posti, nell'SFC equivalente si dovranno avere 12 step.

L'SFC equivalente risulta essere:

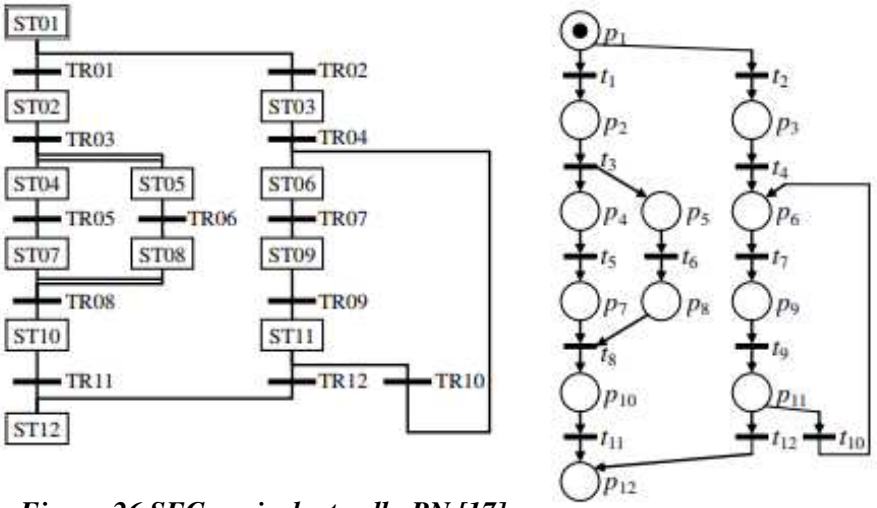


Figura 26 SFC equivalente alla PN [17]

Confrontando il codice SFC con il suo modello di PN, si trova di fatto la corrispondenza a uno a uno tra i posti e gli step, come dice la tecnica di traduzione utilizzata.

.

Una situazione che merita particolare attenzione è quella che riguarda la gestione della mutua esclusione, cioè quando nel modello di PN si rappresenti un conflitto. Nella tipologia di PN considerate, i conflitti vengono gestiti in maniera non deterministica e nel calcolo delle marcature raggiungibili si considerano tutte le possibili evoluzioni di scatto.

Se questo caso non viene gestito, può capitare che l'esecuzione avvenga sempre e solo per un solo ramo di codice, solitamente il ramo di destra, ma ciò dipende dal compilatore del PLC. Si dovrà cercare allora una soluzione per garantire che entrambi i rami possano essere attivati.

Questa problematica verrà affrontata con l'introduzione della *priorità dinamica.*

4.4 *Priorità dinamica e Semaforo*

Nel paragrafo precedente si sono analizzate le principali strutture di un sistema, cioè tutte quelle strutture che si possono incontrare durante la fase di progettazione e di controllo di un processo di automazione industriale.

Queste strutture vanno ora analizzate al fine di evitare eventuali problemi, tra cui mutua esclusione, sincronizzazione, accesso a una zona condivisa da più processi. In questo capitolo si cercherà una soluzione per analizzarli.

Si è introdotto il termine *priorità dinamica:* essa entra in gioco nel momento in cui due transizioni sono abilitate contemporaneamente, ma risultano tra loro in conflitto condividendo risorse comuni, ovvero serve per gestire la *mutua esclusione* di più transizioni.

Si è detto che in linguaggio SFC una transizione è superabile quando tutti gli stati a monte di essa sono attivi; lo scatto della transizione disattiva gli stati a monte e provoca l'attivazione degli stati a valle. Il problema della traduzione in SFC di due (o più) transizioni in conflitto è l'indeterminatezza che non può essere presente nell'SFC. La soluzione da cercare è quella che si può scegliere di far scattare una transizione piuttosto che un'altra in basa a una priorità da definire [3].

Si consideri la PN in figura 27:

Modellando la PN con la tecnica di mappatura a uno a uno, essendo una rete binaria, la traduzione è istantanea: ad ogni posto della PN equivale uno step dell'SFC. Per gestire il conflitto, non potendo lasciare la sua risoluzione indeterminata, una soluzione a questo è utilizzare una PN con priorità.

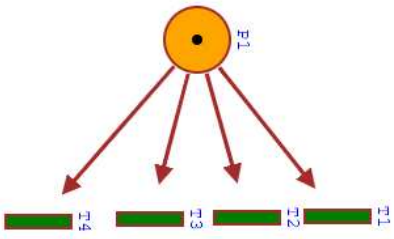


Figura 27 PN divergenza

L'SFC equivalente è quello in figura 28:

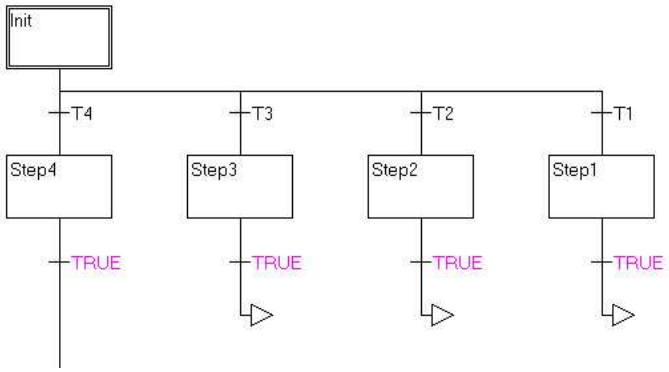


Figura 28 SFC divergenza

Si consideri attivo lo stato iniziale 'Init', quindi sono superabili le transizioni T1, T2, T3, T4. Lo stato che poi sarà quello di arrivo dipenderà dal verificarsi delle *condizioni* delle varie transizioni.

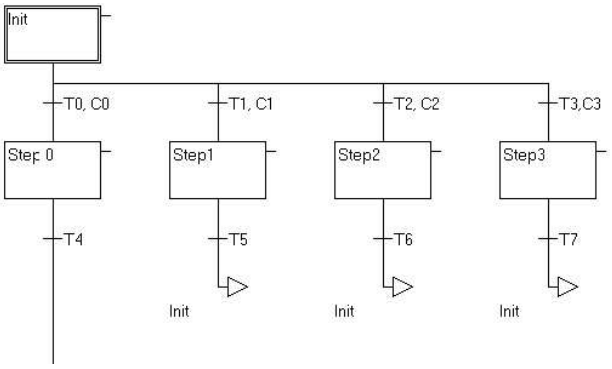
In un codice SFC corretto, le condizioni delle transizioni dovrebbero essere mutuamente esclusive.

Lo Standard introdotto dice che un solo step deve essere attivato quando ci troviamo in situazioni come questa. Questo della selezione non è un problema così banale e va gestito correttamente.

In genere, in caso di due o più transizioni superabili, scatta prima quella a destra, definendo una *priorità fittizia* che però è decisa a priori dalla specifica implementazione del PLC e non dal progettista.

Come si può risolvere questo problema?

Si riprenda l'SFC della divergenza:



La soluzione da cercare è quella di fare in modo che le condizioni associate alle varie condizioni siano in mutua esclusione tra di loro, cioè si vogliono rendere in mutua esclusione tra loro le C0, C1, C2, C3.

Un modo per renderle tali, applicando la logica booleana, è quello di metterle in relazione, definendo però una transizione T a priorità maggiore rispetto alle altre. Scrivendo le condizioni nel modo seguente, la T0 risulterebbe avere la priorità maggiore [3]:

- C0 = condizione0;
- C1 = condizione1 & (NOT condizione0);
- C2 = condizione2 & (NOT condizione1) & (NOT condizione0);

- C3 = condizione3 & (NOT condizione2) & (NOT condizione 1) & (NOT condizione0).

Il problema della selezione (o divergenza) è uno dei più frequenti in ambito industriale.

Un esempio può essere quando si è in presenza di una macchina che lavora due tipologie di pezzi: pezzi A e pezzi B, si ipotizzi di due colori diversi.

Si consideri che i pezzi siano trasportati da un unico nastro trasportatore, e che una volta arrivati al 'fine corsa' posto alla fine del nastro, un sensore di colore venga attivato.

Il modello matematico modellato tramite la PN (figura 29), e poi tradotto nel codice equivalente SFC tramite la mappatura a uno a uno è:

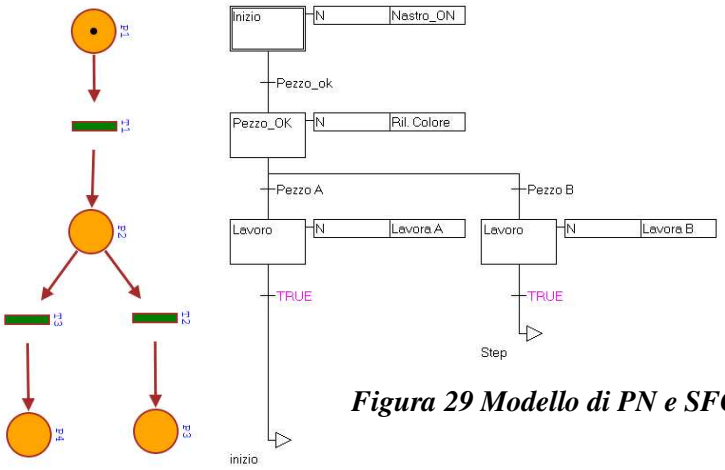


Figura 29 Modello di PN e SFC associato

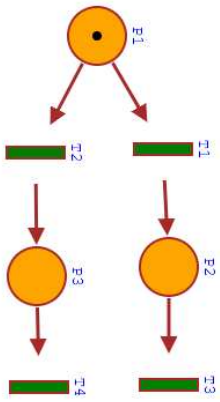
Questo è un classico esempio di mutua esclusione; i due rami non possono essere attivati contemporaneamente; il sensore di colore rileva il colore A o il colore B.

Talvolta, la soluzione mostrata in precedenza di assegnazione di implicite priorità alle transizioni non soddisfa le esigenze progettuali. Infatti, se ad esempio le condizioni *condizione0* e *condizione1* fossero sempre vere contemporaneamente, solo la condizione C0 sarebbe sempre verificata, e la transizione T1 non sarebbe mai superabile.

A tale scopo *s*i introduce la *priorità dinamica* [18].

4.4.1 *Priorità dinamica*

Per ovviare che nel caso di contemporaneo verificarsi di due condizioni venga attivata sempre solo una transizione, va definita una "regola" che gestisce questo caso. La soluzione è per esempio l'introduzione della *priorità dinamica* [18].

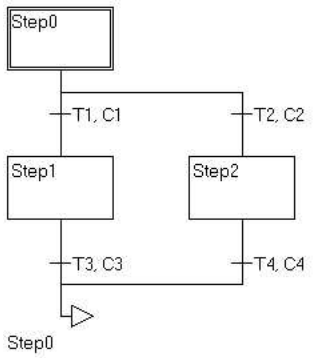


Si supponga di avere la PN in figura: se si ha un token nel posto P1, e se le condizioni delle T1 e T2 risultano entrambe vere, solo una di esse può scattare, in base a una priorità (non possono scattare contemporaneamente nelle PN). Però può anche succedere che a scattare sia sempre e solo la stessa.

Nello Standard SFC, la priorità è costante, e quindi il processo servito è sempre lo stesso, come nell'esempio che si è visto.

Si riporta la PN nell'equivalente codice SFC utilizzando la tecnica di mappatura a uno a uno; essendo una PN binaria, l'equivalente SFC è

semplice da calcolare: ad ogni posto della PN equivale uno step dell'SFC:



La mutua esclusione nasce nel momento in cui lo Step0 risulti attivo, e le condizioni delle transizioni C1 e C2 risultino contemporaneamente vere (se non fossero entrambe vere non risulterebbe un problema).

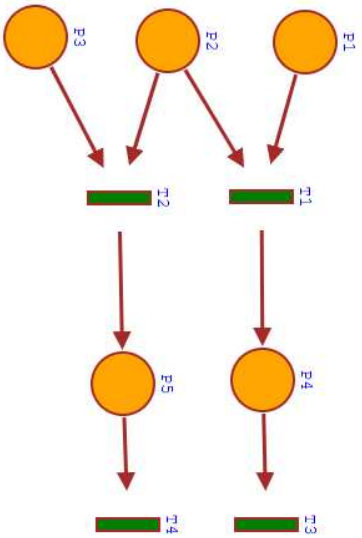
Le transizioni T1 e T2 sono relative all'assegnazione della risorsa al primo e secondo processo. Se tutte e due le condizioni delle transizioni C1 e C2 sono vere e lo Step0 è attivo, solo una tra T1 o T2 scatta, e si deve "decidere quale" con una priorità [18].

Per capirlo meglio si può vedere con un esempio: si supponga di avere un robot che deve prelevare dei pezzi provenienti da due linee diverse. Un sensore alla fine di ogni linea sta a indicare "pezzo disponibile" e quindi pronto per essere prelevato dal robot. Ci si può trovare in tre situazioni differenti:

1. Pezzo disponibile solo sulla prima linea: in quel caso il robot prende quel pezzo ma che è anche l'unico. La condizione dell'altra linea sarà "false" in quanto il pezzo non è ancora arrivato sul fine corsa;
2. Pezzo disponibile sulla seconda linea: duale rispetto al caso precedente. Il pezzo sulla seconda linea è anche l'unico;
3. Pezzo disponibile su entrambe le linee: questo è il caso da risolvere e qui siamo in presenza della mutua esclusione: il robot vede

disponibile il pezzo su entrambe le linee, le condizioni "pezzo disponibile" sono vere per tutte e due. Quindi la risorsa, in questo caso il pezzo, viene assegnata alla linea con priorità maggiore, definita in fase progettuale o prestabilita dal software.

La rete che modella questo caso è la seguente:



Nella situazione normale, standard, la priorità è costante, quindi, se si chiamano le due linee A e B, può accadere che il robot prelevi il pezzo sempre sulla linea A (sotto l'ipotesi che il pezzo sia disponibile su entrambe le linee). È qui che si deve introdurre la priorità dinamica.

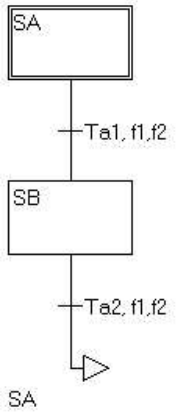
La *priorità dinamica* ha due condizioni da rispettare [18]:

1. se solo un processo ha una richiesta per prendere la risorsa, allora può prenderla (come nei punti 1 e 2 dell'esempio precedente);
2. Se due processi hanno richieste per prendere la risorsa nello stesso momento, quello che non ha utilizzato quella risorsa nella

precedente richiesta simultanea, questo turno può prenderla, ha precedenza.

La soluzione sarà quella di introdurre un codice SFC ausiliario in appoggio al principale, così che vada a modificare le condizioni delle transizioni, avendo così vera solo una delle due, e non tutte e due contemporaneamente.

L'SFC ausiliare è composto da 2 stati e 2 transizioni [18]:



f1 e f2 sono le condizioni delle transizioni del codice ausiliare, mentre nel principale le condizioni sono C1, C2, che a loro volta dipendono dalle f1 e f2. Quindi f1 e f2 mi andranno a modificare C1 e C2 (f1 e f2 saranno all'interno delle C1, C2).

Lo stato iniziale è SA, cioè al primo ciclo di esecuzione la precedenza viene data a T1, poi nel ciclo successivo sarà attivo SB e la precedenza ce l'avrà T2.

La mutua esclusione nasce nel momento in cui sia f1 che f2 sono vere, cioè quando entrambi i processi richiedono l'accesso a una risorsa condivisa. Finché una delle due è falsa, siamo nel caso generale in cui solo un processo fa richiesta, e quindi prende lui la risorsa, essendo anche l'unico.

Per la gestione dello scatto, le condizioni C1 e C2 vengono scritte mettendo in relazione booleana lo stato S (SA o SB) e le f1 e f2. La trascrizione si può riassumere nella tabella di verità:

f1	f2	S	C1	C2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

Le prime 6 righe non incidono sulla mutua esclusione, poichè si verifica il caso in cui o solo f1 è vera, o solo f2 (o zero entrambe o solo una). La mutua esclusione entra in gioco nel momento in cui f1=1 e f2=1, cioè quando sono entrambe vere (ultime due righe).

- Le prime due colonne (fi, f2) indicano l'attivazione di una delle due o di entrambe le condizioni;
- La colonna S indica che se S=0, lo stato attivo è SA, altrimenti se S=1 lo stato attivo è SB, e stabilisce quindi a chi spetta la precedenza di scatto al prossimo ciclo di esecuzione.

Il caso in cui S=1, sta a indicare che nel ciclo precedente rispetto a quello attuale, la risorsa è stata acquisita dal processo 1, e quindi ora la precedenza spetta al secondo processo, cioè la condizione C2 dovrà essere vera e la C1 falsa.

La condizione da verificare, utilizzando ad esempio la logica booleana diventa:

- C1 = f1(NOTf2) + f1S: per risultare verificata la condizione C1 si dovrà avere sicuramente f1=1, perché C1 è riferito ad essa. Le due condizioni poi gestiscono l'eventualità della contemporanea condizione f2=1 tenendo conto del criterio di alternanza gestito dal codice SFC ausiliario.

- C2 = f2(NOTf1) + f2(NOT S): similmente per l'attivazione di C2. In questo caso però f2 dovrà essere uguale a 1, per l'attivarsi di C2 e la rete ausiliare gestisce poi l'alternanza.

Con queste condizioni, si ha la priorità dinamica. Viceversa, per avere la priorità costante, le due condizioni devono essere:

- C1 = f1;
- C2 = f2(NOT f1).

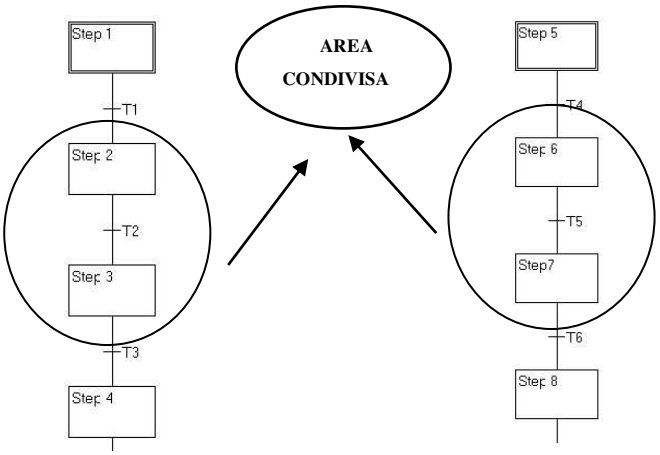
Cioè basta che una sia vera. In questo caso si vede facilmente che t1 ha maggiore priorità rispetto a t2, in quanto la condizione standard è C1=f1, mentre l'altra scatta solo quando è negata la f1 (dipende dalla prima).

Si è analizzato l'accesso in mutua esclusione a una determinata risorsa introducendo la priorità dinamica.
Ora si analizza il caso dell'accesso di un processo a una zona condivisa e il caso della sincronizzazione tra processi, modellati introducendo la struttura *semaforo [3].*

4.4.2 *Semaforo*

Si supponga di avere due sequenze che lavorano in modo indipendente, ma che a un certo punto devono accedere a una determinata area condivisa. Questa zona risulta una risorsa condivisa e i due processi vi possono accedere in maniera mutuamente esclusiva tra loro.
Si può pensare ad esempio a due robot R1 e R2 che devono prendere un pezzo proveniente da un nastro in una zona comune di interazione, e dove se si muovessero contemporaneamente andrebbero ad urtare.

L'obiettivo che si vuole è che l'accesso al pezzo dei robot avvenga in modo esclusivo, cioè o si sposta R1 o R2. Il problema va risolto con l'accesso controllato alla sezione critica.
Si immagini di avere due sequenze:



Gli stati che si riferiscono ai robot operanti nell'area condivisa sono: Step2, Step3, Step6 e Step7. Questi devono risultare mutuamente esclusivi, cioè se il sistema si trova negli stati Step2, Step3, non si devono poter attivare Step6 e Step7, o viceversa.
La condizione della T1 e la condizione della T4, cioè le transizioni che una volta superate portano all'interno della sezione critica dovranno tener conto di questa limitazione.
La tecnica vista nel paragrafo precedente non risolve questo problema; si analizza perché. Vediamo di capire perché e come risolvere il problema in esame.

Si supponga che la prima sequenza sia arrivata allo Step1 e la seconda sequenza allo Step5, cioè immediatamente prima di entrare nella zona condivisa. Indicando le condizioni con la stessa numerazione delle transizioni, analizziamo il comportamento del codice SFC se C1 e C2 fossero le seguenti:

- C1 = Condizione1;
- C4 = Condizione4 & (NOT Condizione1).

Lo stato che diventa attivo è lo Step2. Però così facendo può accadere che se la condizione C1 diventi falsa mentre si è nello Step 2 o nello Step3, risulterebbe superabile anche l'altra condizione C4 di accesso alla sezione, attivando gli Step 6 e 7.

Così facendo non viene rispettata la mutua esclusione perché si avrebbero entrambi i processi nella zona condivisa.

Allora c'è bisogno di introdurre una struttura che regolamenti questo accesso, e una soluzione può essere quella dell'introduzione della struttura *semaforo*, simile alla struttura semaforo del linguaggio di programmazione C [3], [19].

Il semaforo è una struttura che garantisce la mutua esclusione.

Utilizzando una variabile di tipo semaforo, se una sequenza accede alla sezione critica, attiverà un nuovo stato S (semaforo), che l'altro ramo vedrà come disattivo ('semaforo rosso').

Si analizzerà prima il modello con la Rete di Petri e poi il codice SFC equivalente modellato con la tecnica di mappatura a uno a uno.

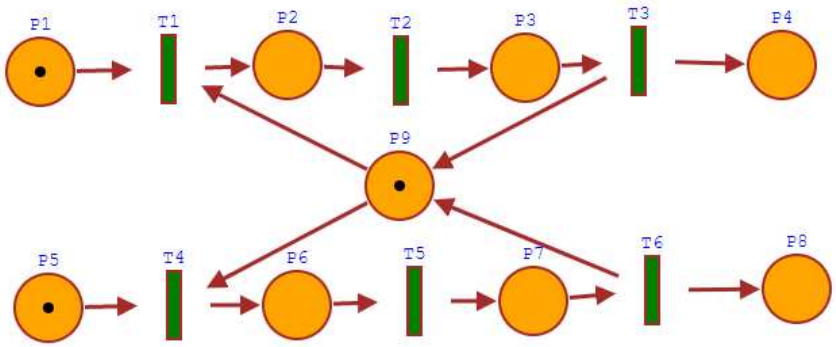
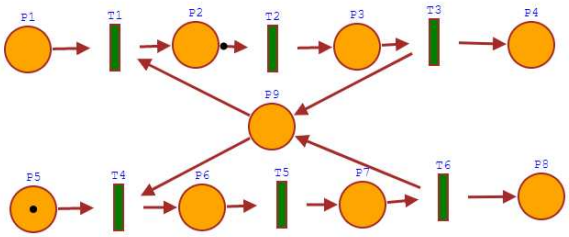


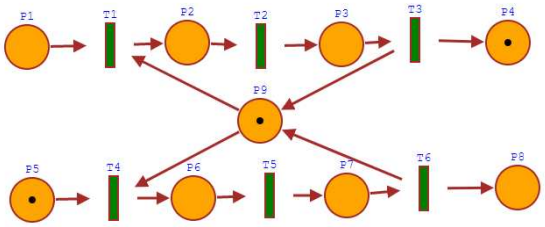
Figura 30 PN di due sequenze da regolare con il Semaforo

Il semaforo, che nella PN equivale al posto P9, risolve il problema: nella marcatura iniziale ci deve essere il token nel posto del semaforo (in quanto all'inizio è sbloccato) e nei posti immediatamente prima di entrare nella sezione condivisa. Si faccia evolvere la rete:



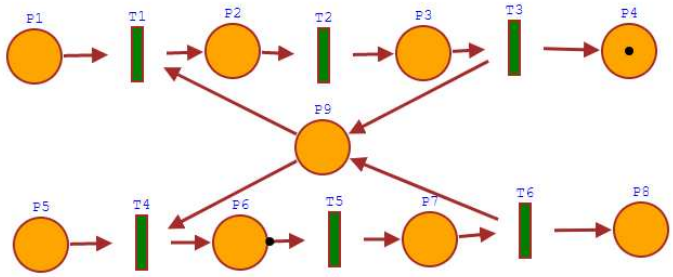
Si noti che all'attivazione della T1, il semaforo perde il token, proprio perché è stato acquisito dal processo.

Poi, viene rilasciato, e torna disponibile dallo scatto di T3:

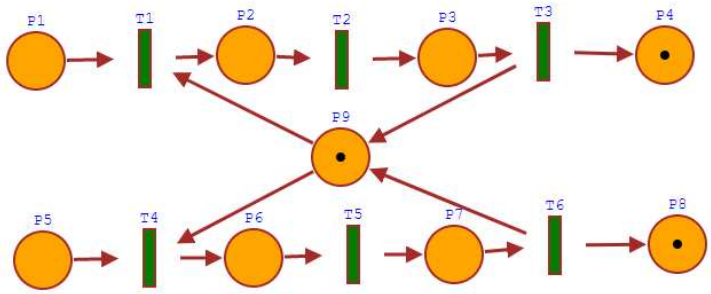


Questa è l'evoluzione al primo ciclo della PN con il semaforo: il primo processo ha completato le proprie lavorazioni, poi rilascia il semaforo mettendolo disponibile per altri processi.

Adesso in questa situazione, in attesa c'è solo il processo due, che ha il semaforo disponibile; quindi, può completare anch'esso le proprie operazioni:



In conclusione, l'evoluzione della PN giunge alla marcatura finale:



Entrambi i processi hanno svolto le proprie operazioni (il token è arrivato in fondo) e il semaforo è tornato disponibile in attesa che i processi facciano nuovamente richiesta.

Dalla PN, tramite la tecnica di mappatura a uno a uno, si trova l'equivalente codice SFC; essendo una rete binaria, la sua traduzione è istantanea: ad ogni posto della PN equivale uno step dell'SFC [3]:

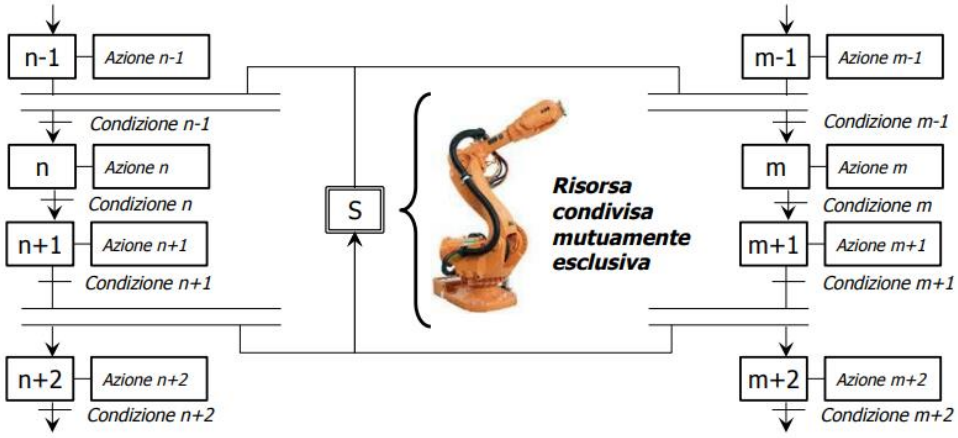


Figura 31 SFC equivalente alla PN con la Struttura Semaforo [3]

In questo esempio si possono osservare due sequenze indipendenti, A e B, con una risorsa condivisa da entrambe, il manipolatore.
Il problema della mutua esclusione nasce nel momento in cui entrambi i processi fanno richiesta di accesso al robot e nello stesso istante, questo perché nel caso solo una delle due faccia richiesta per l'utilizzo del robot, non ci sarebbe nessun problema. Allora qui entra in gioco il semaforo.
Il Semaforo essendo uno step necessario al superamento delle fasi di n-1 o m-1 (doppia linea) risulterà attivo o inattivo [3].

a. Attivo significa che nessuna sequenza ha fatto richiesta per acquisire la risorsa condivisa, e quindi la sequenza che arriva prima la può acquisire;
b. Inattivo invece che c'è un'altra sequenza che sta utilizzando la risorsa condivisa, e si deve mettere in attesa che venga sbloccato dall'altra sequenza.

Si distinguono tre casi:

1. La sequenza A arriva per prima in zona critica: gli step a monte della condizione (n-1) sono lo step (n-1) e S. Se entrambi attivi si attiverà lo step n, e poi successivamente (n+1). Se fosse vera la condizione (n+1), questa provocherebbe la disattivazione dello step (n+1) e di S, e l'attivazione dello step (n+2). Così facendo, lo step S torna inattivo, cioè nessuna sequenza è in sezione critica, in modo che qualcun altro può accedervi.
2. Si supponga che la sequenza A sia in sezione critica, ma che contemporaneamente arrivi anche la sequenza B a fare richiesta di

accesso. In questo caso la sequenza B permane nella fase m-1 in attesa.

3. Si supponga che la sequenza A abbia completato le operazioni nella zona condivisa e quindi ha sbloccato il semaforo, mettendolo inattivo. In quel caso, siccome B era in attesa che S si sbloccasse, può accedervi, e così anch'esso potrà completare le sue operazioni.

L'aggiunta dello Step Semaforo garantisce l'accesso in mutua esclusione su una risorsa condivisa, in un modo migliore rispetto alla sola logica booleana.

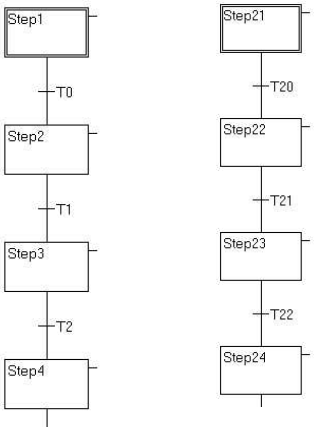
Lo step Semaforo S è stato indicato come stato iniziale (doppio rettangolo), dal momento che in genere che all'attivazione del processo la risorsa è libera e disponibile, e chiunque arrivi per primo può acquisirla.

Per evitare il problema di cui si parlava sopra, le condizioni delle varie transizioni devono essere in mutua esclusione tra loro in modo da garantire che in un momento qualsiasi anche l'altra sequenza entri in sezione critica.

4.4.2.1 Sincronizzazione

L'introduzione della struttura semaforo risulta utile anche nella gestione della sincronizzazione [3], [19], cioè quando due sequenze devono essere sincronizzate in un punto per garantire il corretto funzionamento del sistema.

Si supponga di avere due sequenze, ma che la seconda sequenza non possa superare un certo step finché la prima sequenza non abbia terminato una certa operazione. Si vuole che la sequenza 2 (destra) non superi lo step 23 finché la prima sequenza (sinistra) non abbia completato le azioni dello step 2.



Una tecnica sbagliata è quella di scrivere le condizioni delle transizioni raggruppate in AND tra loro (come si è visto negli esempi precedenti); poiché così facendo le sequenze in attesa diventerebbero tutte e due anziché una, come si voleva.

L'utilizzo del semaforo risolve il problema [3].

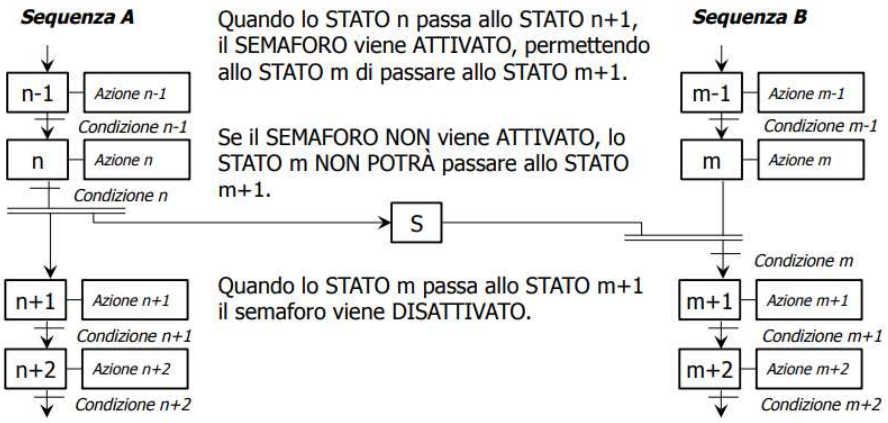


Figura 32 Sincronizzazione risolta con il Semaforo [3]

Con la struttura semaforo, si vuole che la sequenza ad attendere sia solo una, in questo caso la B. Quindi la sequenza A potrebbe scorrere indipendentemente dall'evoluzione di B, mentre B deve attendere A. Allora quando la condizione n è vera, vengono attivati sia lo step (n+1) sia lo step S, e quest'ultimo fa attivare lo stato (m+1) per la sequenza B. B deve attendere finché A non abbia completato le proprie operazioni.
La necessità di sincronizzazione è un esempio che si può trovare frequentemente in industria, ad esempio se si hanno due processi, due lavorazioni, dove il primo deve completare operazioni su pezzi che poi serviranno al secondo processo.

Capitolo 5
ANALISI RISULTATI

In questo capitolo verranno analizzati due esempi a cui verranno applicati le metodologie di traduzione affrontate nel capitolo 4 (stesso esempio per entrambe le metodologie):

1. Traduzione da PN a SFC;
2. Traduzione da PN a LD;

5.1 *Esempio da PN a SFC*

Si supponga di avere una rete di Petri binaria; utilizzando la tecnica di mappatura a uno a uno si sa che è possibile trovare una corrispondenza *a uno a uno* tra i posti della Rete di Petri e gli step del codice SFC.

Se un posto nella rete contiene un token, l'equivalente step nel codice SFC risulterà attivo, e le transizioni associate, se verificate, potranno scattare; Caso contrario, se un posto non contiene un token, l'equivalente step risulterà disattivato in attesa di essere poi attivato [15].

Si prenda un esempio di tank utilizzato in industria chimica.

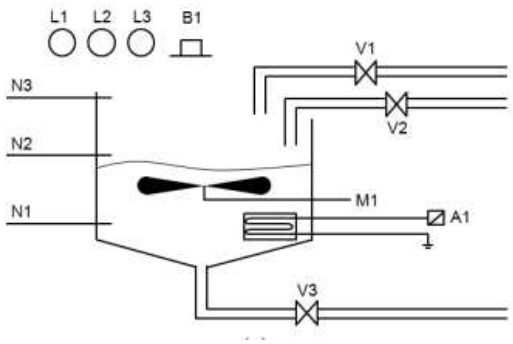


Figura 33 Sistema da studiare [15]

Applicando la metodologia di traduzione analizzata, il primo passo è quello di definire in modo generale il sistema da analizzare; va definito il funzionamento del

sistema [15].

Descrizione del processo: il boiler prende come ingressi due fluidi diversi, rispettivamente solvente e soluto, il cui flusso viene regolato tramite le valvole V1 e V2. Vengono quindi miscelati insieme per formare la miscela composta finale che verrà estratta facendola fluire tramite la valvola V3. Il processo di miscelazione ha bisogno del mescolatore M1 e del riscaldatore A1; ci sono tre sensori per il rilevamento del liquido a livelli differenti: N1, N2, N3. Inoltre, al fine di permettere all'operatore di monitorare il sistema è richiesta l'accensione di tre lampade per la visualizzazione dello stato: L1 che indica che il sistema è pronto per partire, L2 indica che il serbatoio si sta riempiendo e L3 indica che si sta scaricando. Inoltre, è prevista la presenza di un tasto (B1) per l'azionamento del sistema.

Modellazione con rete di Petri: il secondo punto della metodologia è quello di modellare il sistema utilizzando il modello matematico definito dalla rete di Petri; vanno quindi definiti i posti e transizioni che simuleranno il sistema sottoposto a studio.

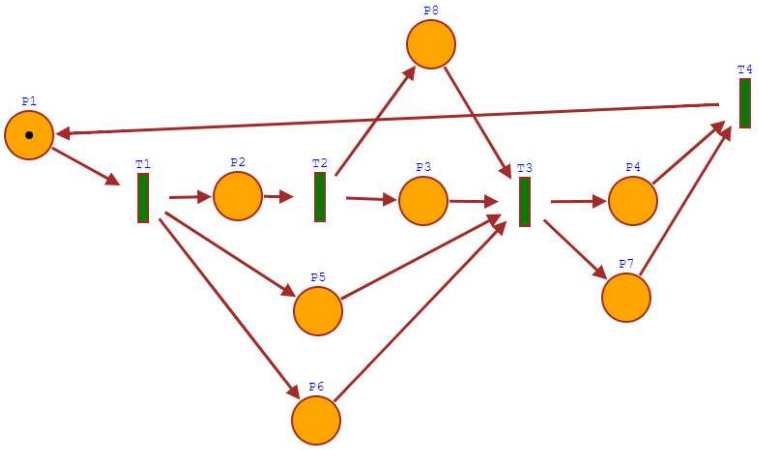


Figura 34 PN esempio tank

POSTI	Descrizione	TRANSIZIONI	Descrizione
P1	Il sistema è pronto per essere azionato	**T1**	Inizio processo
P2	Fluisce il primo fluido	**T2**	Livello primo liquido raggiunto
P3	Fluisce il secondo fluido	**T3**	Livello liquido finale raggiunto
P4	Fluisce il composto finale	**T4**	Livello più basso raggiunto
P5	Riscaldatore ON		
P6	Il serbatoio si sta riempiendo		
P7	Il serbatoio si sta svuotando		
P8	Miscelatore ON		

Analisi proprietà: il terzo passo è quello di analizzare le proprietà della rete; si deve evitare che nel corso di evoluzione vengano raggiunti dei deadlock, in cui il sistema non riesca più a ripartire: alla fine del primo ciclo, in cui il serbatoio viene svuotato, il sistema deve essere in grado di ripartire da capo, si deve verificare la reversibilità, la vivezza, la raggiungibilità.

Verifica binarietà: Il quarto passo è quello di verificare se la rete è binaria o meno; si calcola il grafo di raggiungibilità della rete utilizzando la metodologia di rappresentazione compatta introdotta precedentemente per facilitarne la visualizzazione.
La marcatura iniziare della rete è: [10000000] = 1;

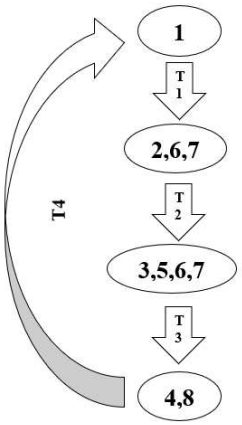


Figura 35 Grafo di Raggiungibilità associato alla PN

L'insieme di raggiungibilità della rete è:

$$R(N) = \{[1], [2,6,7], [3,5,6,7], [4,8]\}$$

Come si è detto in precedenza, una rete è binaria se ogni suo posto contiene al massimo un token: dall'insieme di raggiungibilità, tra tutte le marcature raggiungibili, si hanno al massimo un token e questa è la conferma che la rete è binaria.

Dalla metodologia proposta, si sa che in questo caso si può trovare una corrispondenza *a* uno a uno tra i posti della PN e gli step dell'SFC equivalente, e il risultato dovrà coincidere. Verificata la binarietà, l'ultimo step è quello di generare l'equivalente codice per PLC.

L'equivalente codice in SFC della PN sopra riportata utilizzando questa metodologia trattata è:

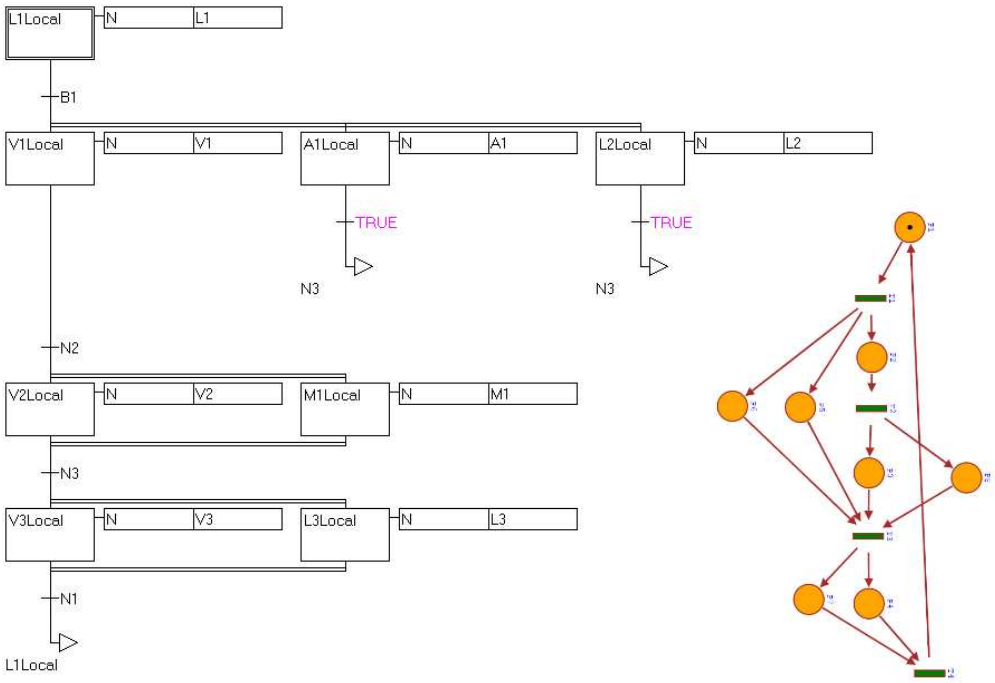


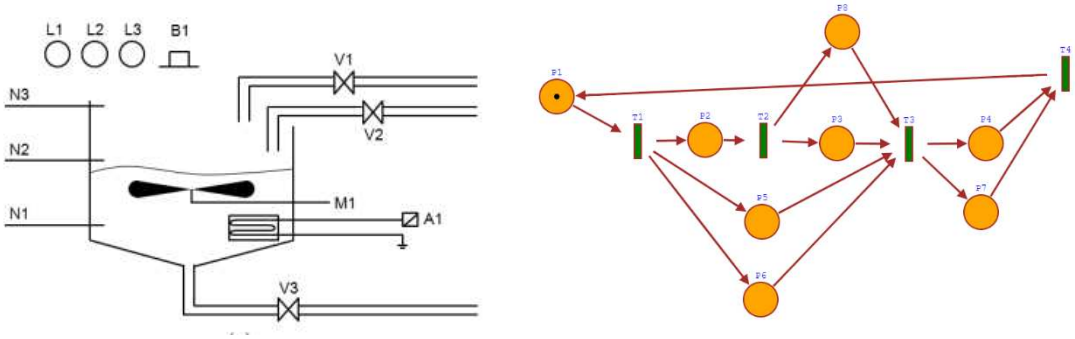
Figura 36 SFC equivalente alla PN

Mettendo in relazione la PN con il codice SFC equivalente, si possono trovare le corrispondenze descritte dalla metodologia; nella rete di Petri si avevano otto posti, e nell'SFC equivalente si hanno otto step, come dice la metodologia trattata.

Partendo dal modello di PN di un sistema, tramite la tecnica di traduzione trattata, si è riusciti ad arrivare al codice equivalente per PLC.

5.2 *Esempio da PN a LD*

Si riprende l'esempio del tank e si applica la metodologia di traduzione di cui si è parlato nel capitolo 4, da PN a codice LD [15].



Per applicare la metodologia, basterà solamente avere a disposizione la PN, e passo dopo passo applicare i passaggi descritti precedentemente. Si è detto che ad ogni posto/transizione vengono associate due variabili: per esempio un posto P1 avrà associate le variabili:

- P1;
- P1Local;

- Stessa identica cosa per tutti gli altri posti e transizioni).

Prima di analizzare la metodologia, introduciamo la corrispondenza tra i posti della PN e le variabili del PLC:

POSTI NELLA PN	VARIABILI DEL PLC
P1	L1
P2	V1
P3	V2
P4	V3
P5	A1
P6	L2
P7	L3
P8	M1

Dalla metodologia, il *primo blocco* è quello di verifica delle transizioni abilitate [15]:

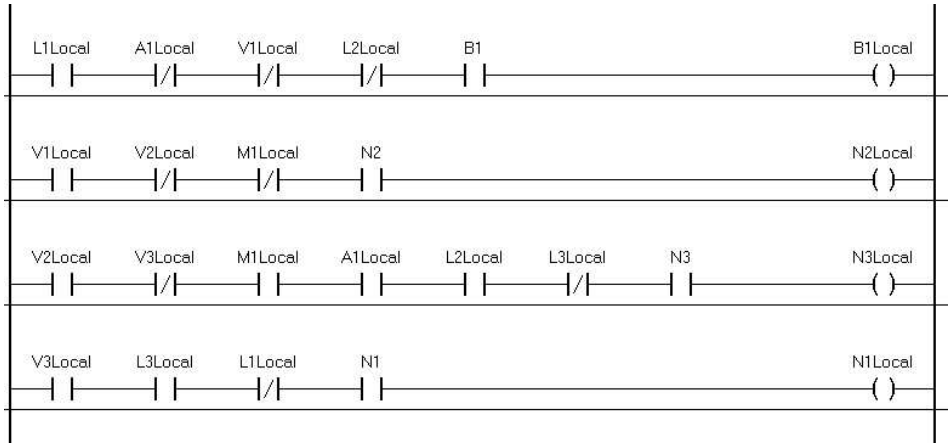
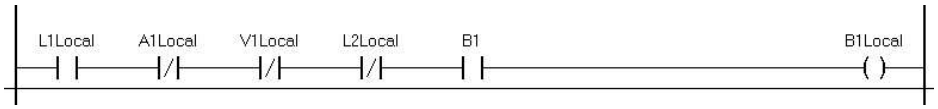


Figura 37 Codice LD primo blocco [15]

Per l'abilitazione ci sono quattro pioli di Ladder, uno per ogni transizione (B1, N1, N2, N3), e si sa che i posti di ingresso (precondizioni) sono impostati come contatti normalmente aperti e i posti di uscita come contatti normalmente chiusi. Analizziamo ora il piolo della prima transizione:



La transizione è associata come una bobina (B1Local), mentre i posti di input come contatti normalmente aperti (solamente L1 è in ingresso alla transizione B1), mentre i posti di uscita come contatti normalmente chiusi (i posti in output da B1 sono: V1, A1, L2). Stessa identica cosa per le altre tre transizioni.

Il *secondo blocco* di codice riguarda l'attivazione delle transizioni: c'è la rimozione dei token i dai posti di input e creazione dei token sui posti di output. Per ogni transizione vengono creati tanti pioli LD quanti sono gli archi ad essa collegati: avendo a disposizione in totale 16 archi tra entranti e uscenti, si dovranno avere 16 pioli di codice Ladder. Dalla tecnica descritta, per gli archi di input, il piolo ha una bobina di ripristino (Reset), per gli archi di uscita, il piolo ha una bobina impostata (Set).

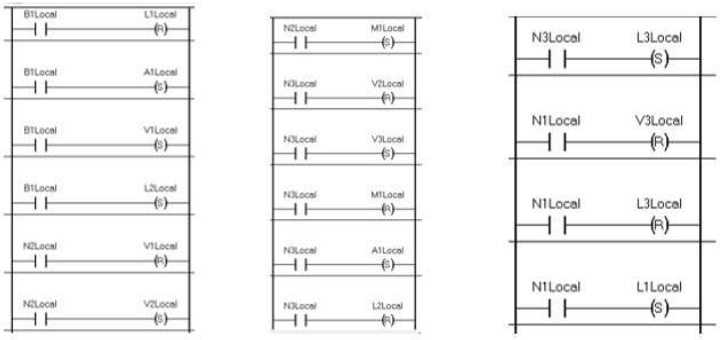
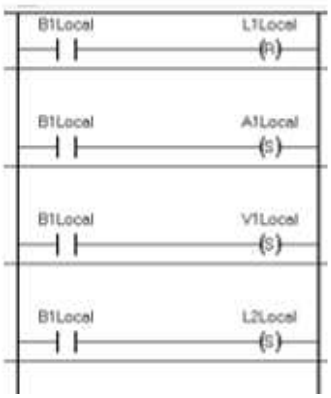


Figura 38 Codice LD secondo blocco (da sx a dx)

Analizziamo il codice relativo alla prima transizione, T1 nella PN, (per le altre la tecnica sarà identica) rinominata nel codice PLC come B1 (si veda la tabella per le corrispondenze dei nomi dei posti con le variabili del PLC).



B1 (T1) ha come ingresso il posto L1 (P1), e come uscita i posti V1 (P2), A1 (P4), L2 (P5). Il posto L1, essendo in ingresso verrà resettato (perde il token) mentre quelli di uscita vengono settati (acquisiscono il token).

Il *terzo blocco* è quello riguardante l'attivazione delle uscite.

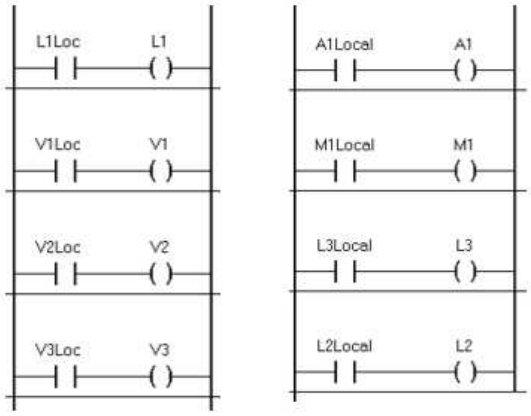


Figura 39 Codice LD terzo blocco (da sx a dx)

Dalla metodologia, ad ogni variabile di uscita è associato un piolo: avendo a disposizione otto posti, ci saranno otto pioli; ogni posto

associato alla variabile di uscita è rappresentato da un contatto normalmente aperto.

Il *quarto blocco* rappresenta le condizioni iniziali: dalla PN, essendoci nella marcatura iniziale solo un token nel posto L1, nel corrispondente codice Ladder, sarà settata solo la L1Local.

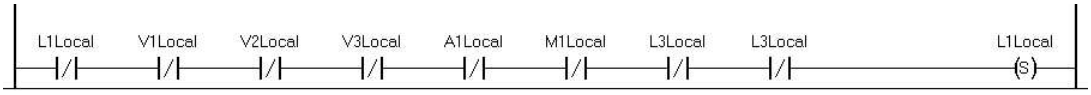


Figura 40 Codice LD quarto blocco

Partendo dal modello di PN di un sistema, tramite la tecnica di traduzione trattata, si è riusciti ad arrivare al codice equivalente per PLC.

CONCLUSIONE

In questo elaborato si è voluto fare una ricerca di metodi di traduzione da Rete di Petri a codice equivalente per PLC. L'attenzione, dopo una ricerca in letteratura, si è focalizzata su due metodi di traduzione, uno relativo al linguaggio LD e uno relativo al linguaggio SFC.

Prima di analizzare tali metodi si sono date delle specifiche di informazione, partendo dall'automazione industriale e analizzando i sistemi che la rappresenti: i DEDS. Come strumento di modellazione di tali sistemi si sono analizzare le PN poiché come si è visto nel capitolo 2 risultano uno dei migliori strumenti di analisi.

Per quanto riguarda i due metodi di traduzione analizzati nel capitolo 4 si può affermare che la traduzione da PN a codice equivalente per PLC risulta più semplice e compatta utilizzando il linguaggio l'SFC, essendo un linguaggio di alto livello e graficamente molto simile al modello di PN. La sua traduzione infatti è prettamente grafica: analizzando il grafo di raggiungibilità è possibile trovare infatti il corrispettivo codice equivalente.

Differentemente, la traduzione da PN a LD non è definita in modo grafico; il linguaggio LD non è visivamente intuitivo e compatto come l'SFC. Questo comporta una tecnica di traduzione più complessa e meno intuitiva.

BIBLIOGRAFIA E FIGURE

[1] Enciclopedia Italiana Treccani

[2] PLC e Automazione Industriale – Pasquale Chiacchio.

[3] Sistemi di automazione industriale: architetture e controllo – McGraw Hill – Bonivento, Gentili, Paoli

[4] Universal Robots – Automazione Industriale

[5] INDUSTRY4.BUSINESS – Automazione Industriale: cos'è e quali sono i vantaggi

[6] NETWORKDIGITAL360 – ECONOMY UP – Da Industria 4.0 a Industria 5.0: qual è la differenza?

[7] Rivista "Automazione e Strumentazione" – AutomazionePlus.it

[8] Automazione Industriale: CLS iMation

[9] Universal Robots – Quali sono i settori di applicazione dell'automazione industriale?

[10] Key4biz - Quotidiano online sulla digital economy e la cultura del futuro

[11] certificazione TUV - Wikipedia

[12] Automazione industriale: controllo logico con Reti di Petri – Luca Ferrarini

[13] Sistemi ad eventi discreti - Angela Di Febbraro Alessandro Giua - McGraw-Hill

[14] Transformation from Petri Nets Model to Programmable Logic Controller using One-to-One Mapping Technique - Devinder Thapa, Suraj Dangol, and Gi-Nam Wang

[15] A TRANSCRIPTION TOOL FROM PETRI NET TO CLP PROGRAMMING LANGUAGES - André Torres Ferraz de Mello, Marcelo Castro Barbosa, Diolino José dos Santos Filho, Paulo Eigi Miyagir, Fabrício Junqueira.

[16] Ladder Diagram and Petri-Net-Based Discrete-Event Control Design Methods - Shih Sen Peng Mengchu Zhou

[17] Research on Structural Analysis for the Petri-net representing Sequential Function Chart - Makoto Okuda, Tatsuaki Nagao, Toru Mizuya, Iko Miyazawa

[18] Implementation of Petri Nets Based Controller using SFC Abbas Dideban - Mohamed Kiani, Hassane Alla

[19] Sistemi di automazione industriale Architetture e controllo - Claudio Bonivento Luca Gentili Andrea Paoli - McGraw-Hill

[20] Algorithm to Convert Signal Interpreted Petri Net models to Programmable Logic Controller Ladder Logic Diagram Models - Z. Aspar, Nasir Shaikh-Husin, M. Khalil-Hani

[21] Petri Nets: Properties, Analysis and Applications – Tadao Murata

[22] ITSMAKER (itsmaker.it)

[22] I modelli di PN sono stati realizzati utilizzando il software al seguente link: https://petri.hp102.ru/pnet.html

[23] I codici PLC sono realizzati utilizzando il software TwinCAT

ELENCO FIGURE