Dipartimento di Elettronica e Informazione
POLITECNICO DI MILANO

Automazione industriale
dispense del corso
20. Implementazione SFC

Luigi Piroddi
piroddi@elet.polimi.it

Introduzione

Vantaggi della progettazione in SFC:

- l’SFC permette una agevole progettazione degli algoritmi di controllo
- con l’SFC è naturale fornire una rappresentazione diretta delle specifiche funzionali
- il programma SFC può essere utilizzato direttamente per la documentazione e la manutenzione del SW di controllo

Non tutti i dispositivi HW di controllo (PC, PLC, ecc.) prevedono l’SFC come linguaggio di programmazione, e alcuni ne implementano solo una versione fortemente ridotta (p.es. senza la possibilità di utilizzare SFC multipli).

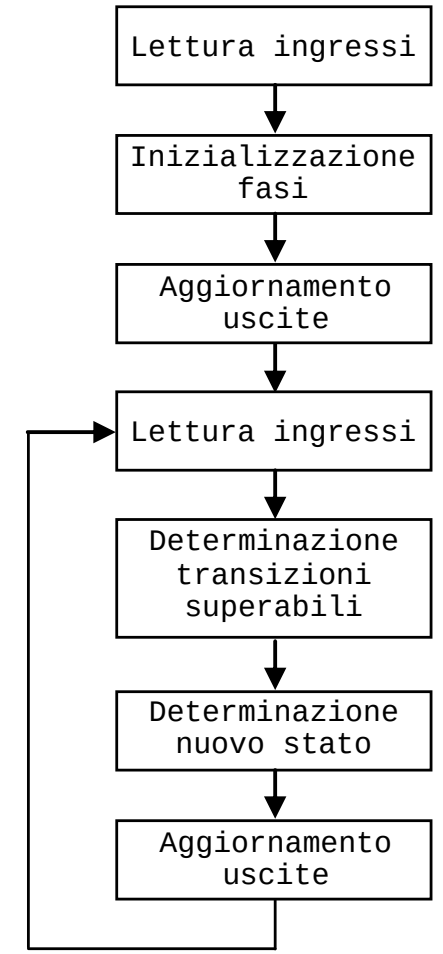
- potrebbe essere quindi necessario tradurre gli schemi SFC in codice implementabile su una qualsiasi macchina per l’elaborazione dell’informazione
- tale traduzione deve preservare la struttura dei programmi SFC, per consentire una mappatura biunivoca tra l’SFC e il corrispondente codice implementativo
- la traduzione può essere fatta mediante equazioni booleane equivalenti o *algoritmi di evoluzione*

Algoritmi di evoluzione

Gli *algoritmi di evoluzione* costituiscono la rappresentazione in forma algoritmica delle regole di evoluzione degli SFC.

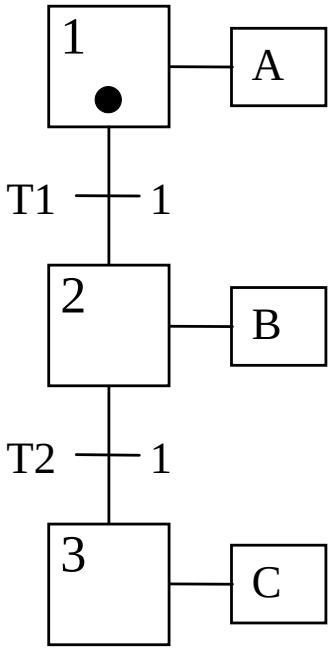
Un algoritmo di evoluzione prevede:

- una *fase di inizializzazione*
  - si attivano le fasi iniziali
  - si aggiornano le eventuali uscite ad esse associate
  - occorre leggere prima gli ingressi, perché le uscite potrebbero essere condizionate (v. azioni condizionate)
- un *ciclo iterativo*
  - lettura degli ingressi
  - determinazione delle transizioni superabili
  - determinazione del nuovo stato di attivazione dei passi dell’SFC
  - aggiornamento delle uscite



L’algoritmo di evoluzione illustrato è detto *senza ricerca di stabilità* perché nel caso in cui più transizioni in una sequenza siano superabili, non le supera tutte in una volta, ma rimane nelle fasi intermedie per almeno un ciclo eseguendo le azioni associate.

Ad esempio, si consideri lo schema SFC:

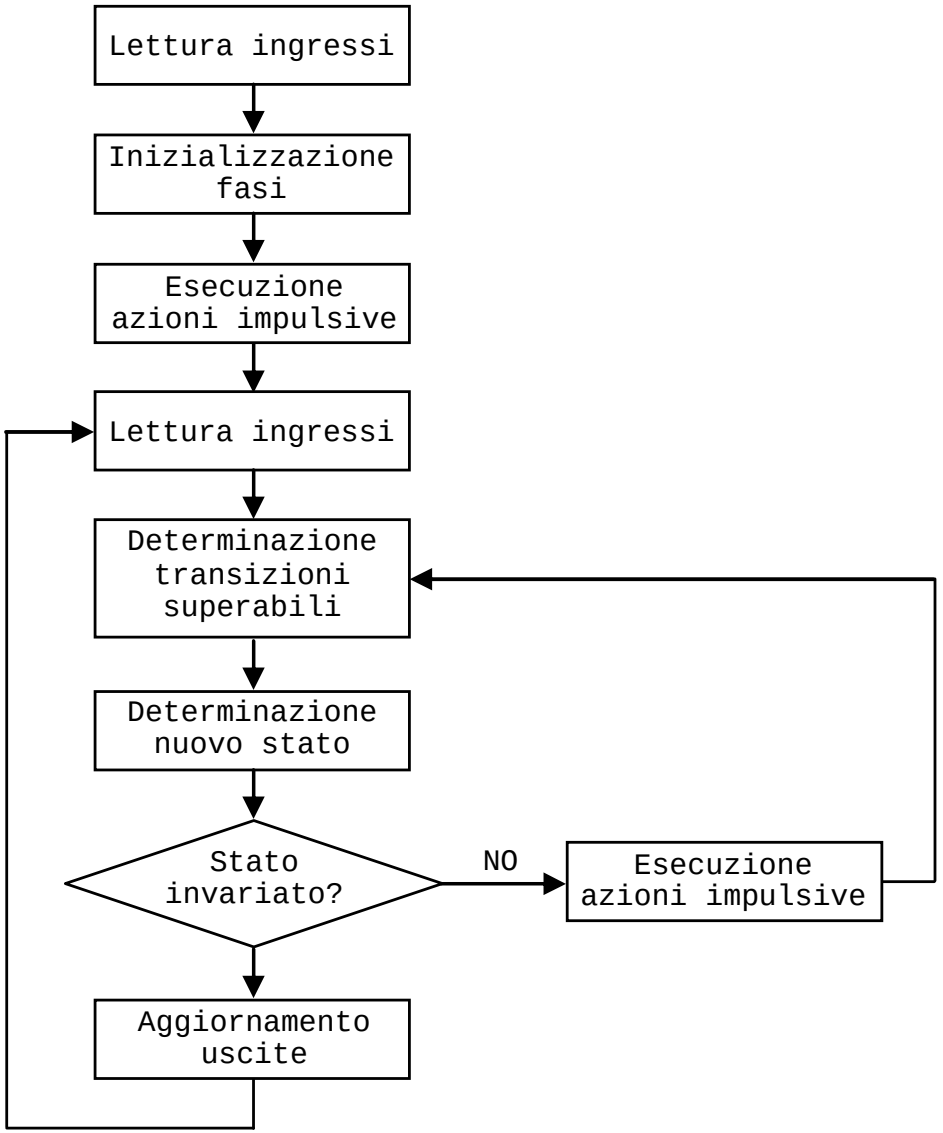


- la fase 2 è *instabile*, perché appena si attiva, la transizione immediatamente successiva (T2) è superabile
- secondo le regole di evoluzione, la permanenza nella fase 2 dovrebbe essere di durata nulla e quindi l’azione B non sarebbe eseguita
- l’algoritmo di evoluzione senza ricerca di stabilità, invece, si ferma nella fase 2 per un ciclo, attivando nel medesimo intervallo di tempo l’azione B

Normalmente questo non rappresenta un problema e si può utilizzare tranquillamente questo algoritmo di evoluzione.

Tuttavia, in alternativa si può implementare uno schema SFC utilizzando un *algoritmo di evoluzione con ricerca di stabilità*, in cui

- le uscite sono aggiornate solo se l’SFC è giunto in uno stato stabile (non varia più sotto l’azione degli stessi ingressi)
- nelle fasi instabili si eseguono solo le azioni impulsive



Codifica dell’algoritmo di evoluzione

La codifica degli algoritmi di evoluzione per l’SFC permette di tradurre un programma SFC in un qualsiasi linguaggio di programmazione.

Noi studiamo la sua traduzione in LD, facendo riferimento all’algoritmo senza ricerca di stabilità.

Operazioni preliminari:

- associare ad ogni fase un bit di memoria dell’area utente che ne indichi lo stato di attivazione (*marker di fase*)
  - i marker di fase servono per stabilire se le azioni associate alle fasi devono essere eseguite e valutare le condizioni associate alle transizioni
- associare ad ogni transizione un bit di memoria dell’area utente che indichi se la transizione è superabile (*marker di transizione*)
  - i marker di transizione dipendono dallo stato dei marker di fase e dalle condizioni associate alle transizioni
  - servono per determinare la nuova condizione dell’SFC aggiornando i marker di fase

Il ciclo dell’algoritmo richiede la lettura degli ingressi, la determinazione delle transizioni superabili e della nuova condizione e infine la scrittura delle uscite.

Il programma LD che realizza la codifica dell’algoritmo di evoluzione è composto dalle seguenti quattro sezioni in sequenza:

❶ sezione di *inizializzazione*

❷ sezione di *esecuzione delle azioni*
(solo scrittura delle variabili, non aggiornamento delle uscite effettive,
v. passo ❺)

❸ sezione di *valutazione delle transizioni*

❹ sezione di *aggiornamento dello stato*

Le fasi di lettura/scrittura sono già realizzate dal sistema operativo del PLC in maniera trasparente all’utente, limitando la necessità di codifica alla pura elaborazione dei dati:

❶ *lettura ingressi*

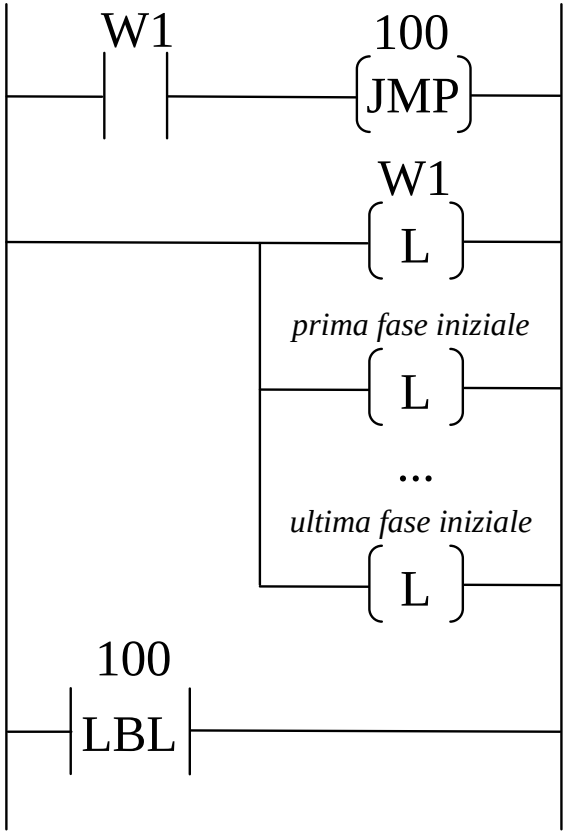
❺ *scrittura uscite* effettive

Osservazioni:

- l’algoritmo senza ricerca di stabilità prevede che ad ogni stato raggiunto, compreso quello iniziale, si eseguano le azioni associate alle fasi attive
- per questo motivo, la sezione di esecuzione delle azioni è inserita immediatamente dopo l’inizializzazione e non alla fine del ciclo di scansione
- così facendo, al primo giro le uscite effettive dipendono comunque soltanto dall’inizializzazione

Sezione di inizializzazione

La *sezione di inizializzazione* deve essere eseguita solo una volta, alla prima esecuzione del programma.



Essa deve porre a 1 i marker delle fasi iniziali (quelle con la doppia cornice) previste dall’SFC.

Ciò può essere realizzato utilizzando delle bobine a ritenzione associate alle fasi iniziali.

La procedura racchiusa tra il rung con il JMP e quello con LBL viene eseguita solo la prima volta.

Dal secondo ciclo di scansione in avanti W1 risulta vera e quindi l’esecuzione del programma salta al rung con LBL.

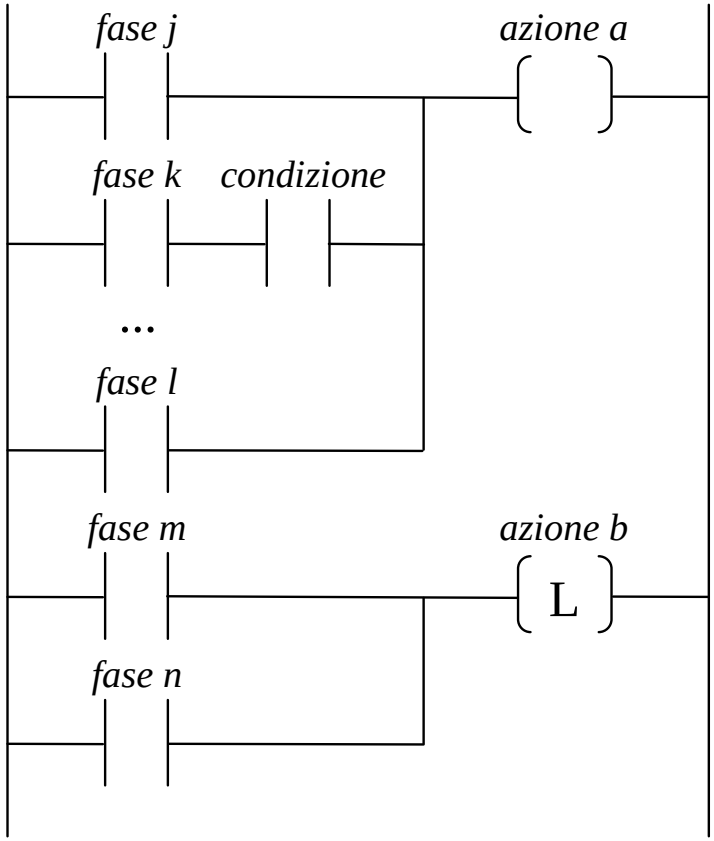
Sezione di esecuzione delle azioni

La *sezione di esecuzione delle azioni* deve aggiornare l’area di memoria riservata alle uscite, che verrà riversata sulle uscite fisiche alla fine del ciclo di scansione, mettendo a 1 le uscite corrispondenti a fasi attive.

La codifica sarà diversa a seconda del qualificatore associato ad ogni azione e a seconda della presenza di un’eventuale condizione sull’azione stessa.

- azione *continua*
  un rung in cui la condizione di abilitazione sia espressa come OR logico dei marker di *tutte* le fasi in cui l’azione è presente
- azione *condizionata*
  la condizione va posta in AND logico con il marker della fase a cui è associata
- azione *memorizzata*
  uso di bobine a ritenzione
- azione *impulsiva*
  o l’azione corrispondente in LD è già di tipo impulsivo (p.es. aggiornamento di un contatore), oppure l’azione deve essere opportunamente codificata usando un riconoscitore del fronte di salita del marker di fase

Esempio:



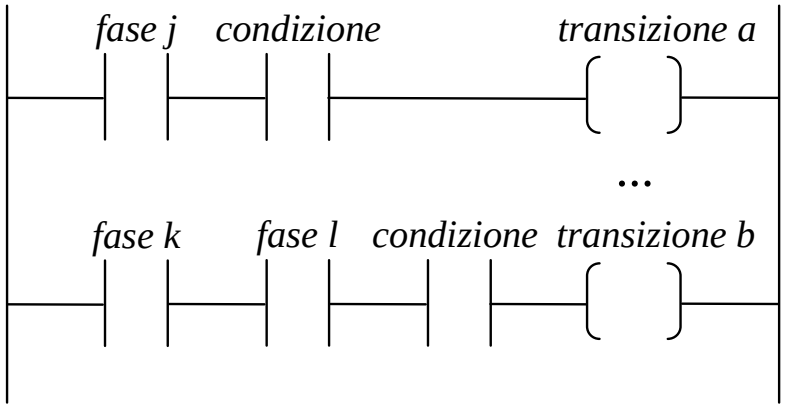
In figura è rappresentata un’azione continua (*azione a*) associata a più fasi.

Per una di queste l’azione risulta anche condizionata.

L’*azione b* è di tipo memorizzato (stored).

Sezione di valutazione delle transizioni

La *sezione di valutazione delle transizioni* deve verificare la superabilità delle transizioni, aggiornando lo stato dei relativi marker.



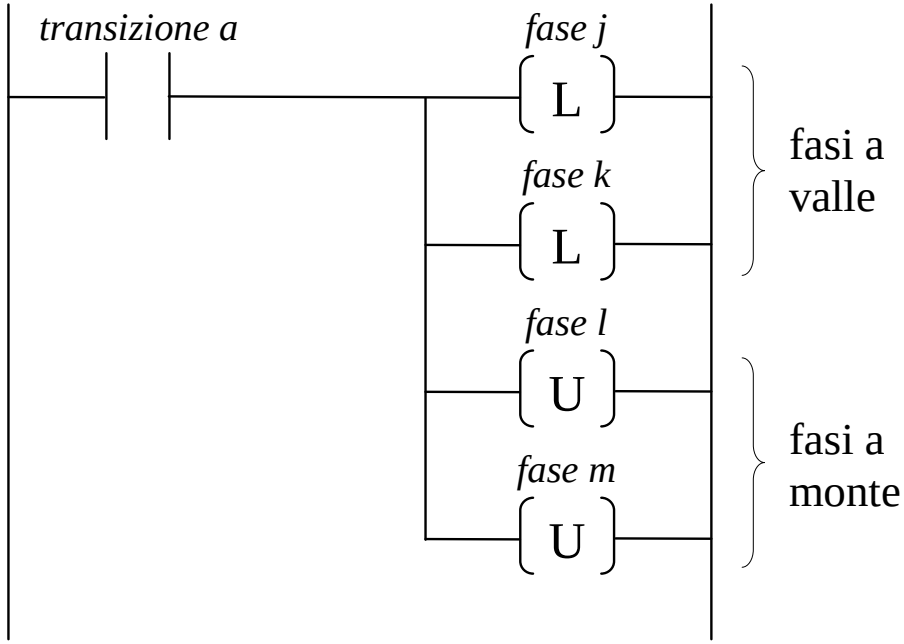
Si rappresenta un rung per ogni transizione, con le fasi a monte e la ricettività in AND logico.

Sezione di aggiornamento dello stato

La *sezione di aggiornamento dello stato* deve eseguire gli scatti delle transizioni valutate come superabili dalla sezione precedente, ovvero

- attivare i marker delle fasi a valle
- disattivare i marker delle fasi a monte

Richiede un rung per ogni transizione, rappresentata con un contatto, mentre le fasi da attivare o disattivare sono associate a bobine a ritenuta.

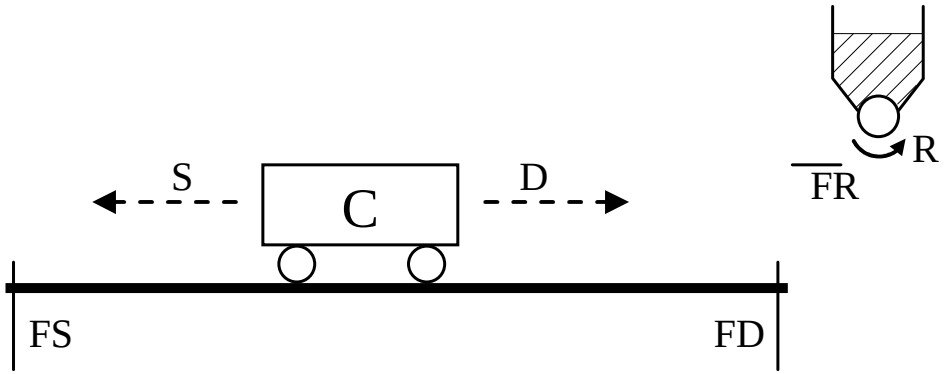


Le variabili temporali eventualmente utilizzate nell’SFC devono essere realizzate mediante dei timer che devono essere abilitati nelle fasi in cui devono contare il trascorrere del tempo, come se fossero delle azioni.

La loro condizione di fine conteggio, realizzabile in LD utilizzando l’indirizzo del timer come contatto, equivale alla variabile temporale dell’SFC.

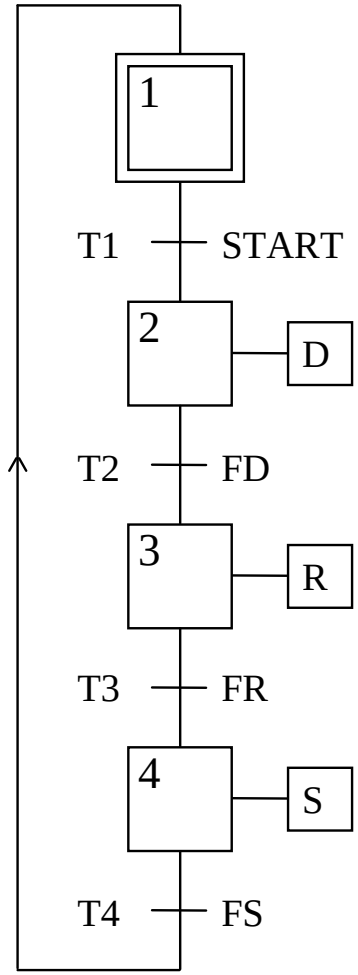
Esempio: carrello

Consideriamo un carrello trasportatore che si muove su una rotaia.

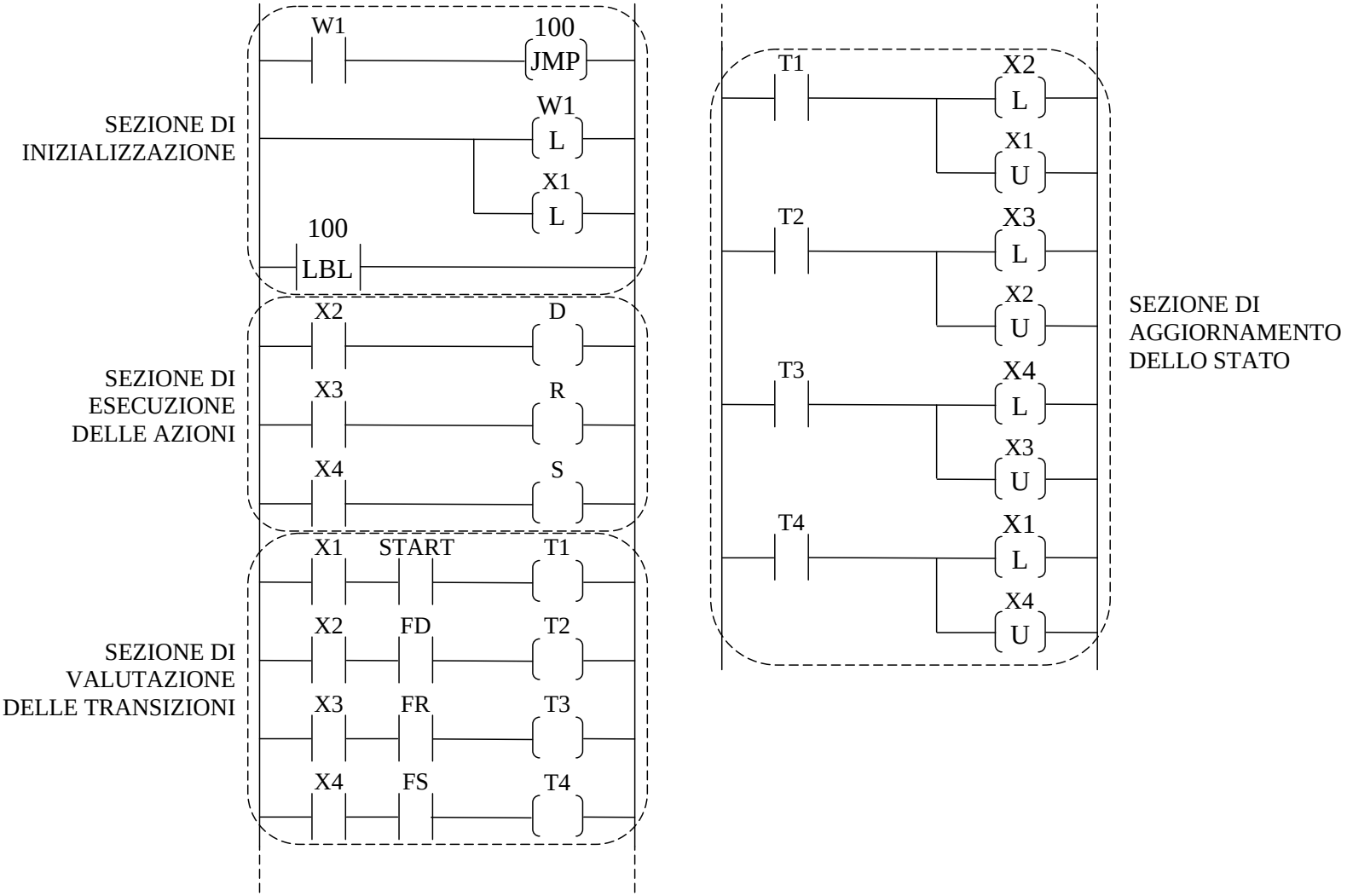


- il processo si attiva su richiesta dell’operatore (START)
- per prima cosa il carrello deve spostarsi al limite destro del binario (comando D, finecorsa FD)
- in tale posizione viene caricato ribaltando un serbatoio (comando R, finecorsa FR)
- infine, il carrello deve spostarsi al limite sinistro del binario (comando S, finecorsa FS)

Schema SFC:



Traduzione SFC in LD:



Traduzione di macroazioni (SFC gerarchici)

La procedura può essere adottata anche per tradurre SFC gerarchici (contenenti macroazioni).

Traduzione di una macroazione di sospensione:

- richiede un rung con un contatto associato alla fase dell’SFC di livello superiore che attiva la macroazione e tante bobine (a ritenzione) quante sono le fasi dell’SFC di livello inferiore, tutte unlatched
- quando si attiva la fase dell’SFC di livello superiore, tutte le fasi dell’SFC di livello inferiore vengono disattivate

Traduzione di una macroazione di forzatura:

- richiede, nel rung corrispondente, una bobina a ritenzione latched per la fase dell’SFC di livello inferiore da forzare
- quando si attiva la fase dell’SFC di livello superiore, tutte le fasi dell’SFC di livello inferiore vengono disattivate tranne quella forzata, che viene invece attivata

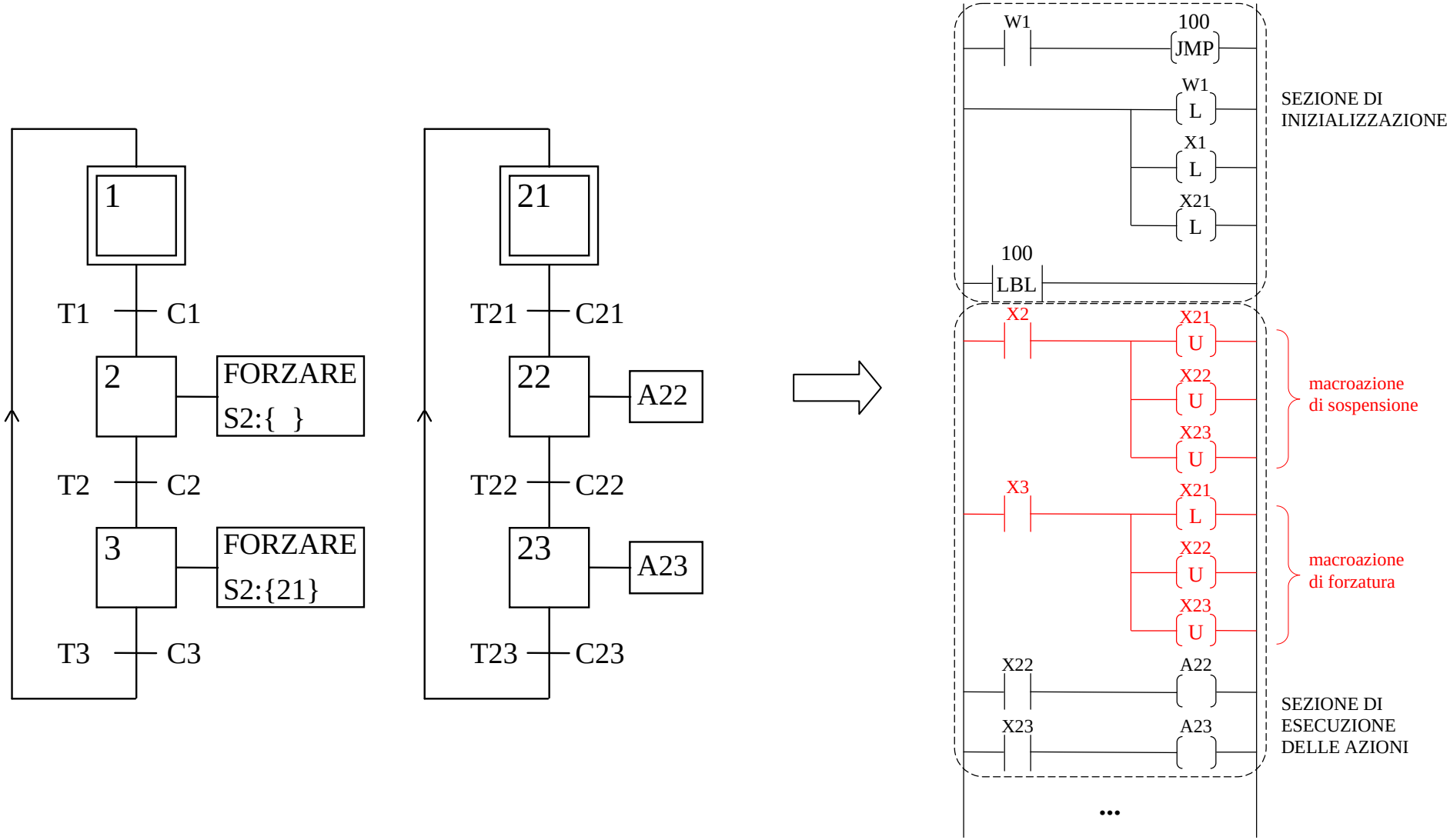
La traduzione della macroazione dovrà essere posta nella sezione di esecuzione delle azioni.

Si osservi che, avendo la macroazione nell’SFC di livello superiore effetto sulla condizione dell’SFC di livello inferiore, la codifica della macroazione deve precedere la codifica delle azioni dell’SFC di livello inferiore.

In questo modo, l’esecuzione della macroazione annulla l’evoluzione dell’SFC di livello inferiore avutasi nella sezione di aggiornamento della condizione.

Le sezioni di inizializzazione, di valutazione delle transizioni e di aggiornamento della condizione riportano le codifiche di ognuno degli SFC della gerarchia (in un qualunque ordine).

Esempio


1
T1
C1
2
FORZARE
S2:{ }
T2
C2
3
FORZARE
S2:{21}
T3
C3
21
T21
C21
22
A22
T22
C22
23
A23
T23
C23
W1
100
JMP
W1
L
X1
L
X21
L
100
LBL
SEZIONE DI
INIZIALIZZAZIONE
X2
X21
U
X22
U
X23
U
macroazione
di sospensione
X3
X21
L
X22
U
X23
U
macroazione
di forzatura
X22
A22
X23
A23
SEZIONE DI
ESECUZIONE
DELLE AZIONI
...

Considerazioni finali

La procedura di codifica adottata introduce relazioni biunivoche tra fasi, transizioni e azioni dell’SFC e rung del codice LD.

- ciò facilita la modifica, l’estensione e il debugging del codice di controllo
- il codice LD prodotto è sicuramente non ottimo in termini di dimensione (maggiore occupazione di memoria)
- programmando direttamente in LD si potrebbe ottenere un codice molto più sintetico

E’ anche vero che la maggiore lunghezza del codice non si paga eccessivamente in termini di tempo di esecuzione.

- infatti, i contatti corrispondenti a marker delle fasi e delle transizioni sono posizionati sempre all’estrema sinistra dei rispettivi rung
- poiché l’esecuzione di un rung viene immediatamente interrotta quando nella scansione da sinistra a destra si incontra una condizione falsa, di fatto verranno saltati tutti i rung corrispondenti a marker di fasi inattive e di transizioni non superabili