Ottimizzazione per FSM

▼ Creatore originale: @Samuele Gentile

Durante la realizzazione di una FSM è possibile avere degli stati equivalenti, ovvero che:

- per qualsiasi ingresso producono le stesse uscite;
- hanno transizioni verso stati uguali o equivalenti.

Quando esistono stati equivalenti, possiamo combinarli in uno solo. In questo modo possiamo avere un risparmio a livello di FF e di porte logiche.

Se, ad esempio, avessimo 5 stati, avremmo bisogno di 3 bit, ma se riuscissimo a condensare due stati in uno ne basterebbero 2, e quindi servirebbe un FF in meno.

L'obiettivo è, quindi, quello di utilizzare il minor numero di bit per coprire tutti gli stati.

Algoritmo "informale"

Per l'algoritmo "informale", bisogna:

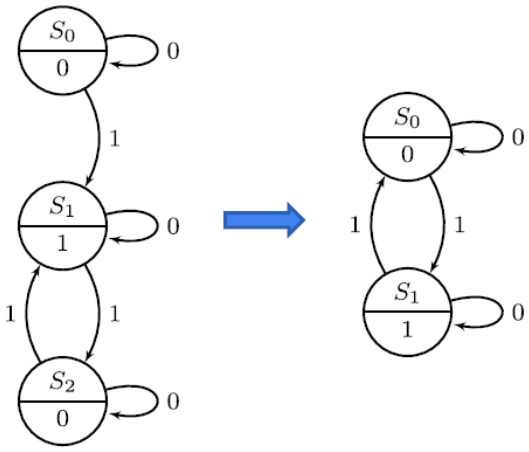
1. guardare la tabella delle transizioni tra gli stati;
2. cercare gli stati con stesso comportamento per le uscite;
3. se hanno gli stessi stati futuri sono equivalenti;
4. si combinano gli stati equivalenti trovati;
5. ripetere fino a che non si riesce più a compattare.

▼ Esempio - Controllore di parità

La FSM illustrata è un controllore di parità, ovvero conta i numeri di 1:

- se sono in numero dispari, restituisce 1;
- se sono in numero pari, restituisce 0.

Ad “occhio”, si può vedere che gli stati S1 e S2 sono equivalenti.



Semplificazione della FSM del controllore di parità

Costruendo la tabella degli stati, possiamo vedere come non possiamo applicare l’algoritmo sopra elencato, poiché sono presenti delle “etichette” diverse nella colonna degli stati futuri di S1 e S2.

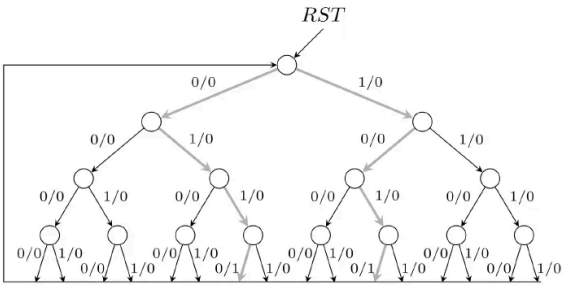
L’algoritmo non copre il comportamento del cappio che si è verificato in questo caso.

	Stato Futuro		
Stato Presente	$x = 0$	$x = 1$	Uscita
S_0	S_0	S_1	0
S_1	S_1	S_2	1
S_2	S_2	S_1	0

Tabella degli stati della FSM

▼ Esempio - Riconoscitore di sequenze

In questo caso, vogliamo costruire una FSM che riconosca solo le righe 1010 o 0110.



Visualizzazione dell'albero RST delle scelte per la sequenza di caratteri

Come visto dall'algoritmo informale, partiamo dalla tabella degli stati.

Suddividiamo le righe in base al numero di bit per ogni sequenza di ingresso.

		Stato Futuro		Uscita	
Sequenza Ingresso	Stato Presente	$x=0$	$x=1$	$x=0$	$x=1$
RST	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_7	S_8	0	0
01	S_4	S_9	S_{10}	0	0
10	S_5	S_{11}	S_{12}	0	0
11	S_6	S_{13}	S_{14}	0	0
000	S_7	S_0	S_0	0	0
001	S_8	S_0	S_0	0	0
010	S_9	S_0	S_0	0	0
011	S_{10}	S_0	S_0	1	0
100	S_{11}	S_0	S_0	0	0
101	S_{12}	S_0	S_0	1	0
110	S_{13}	S_0	S_0	0	0
111	S_{14}	S_0	S_0	0	0

Tabella degli stati iniziale

Notiamo che S10 e S12 hanno gli stessi stati futuri e le stesse uscite, e risultano, quindi, equivalenti.

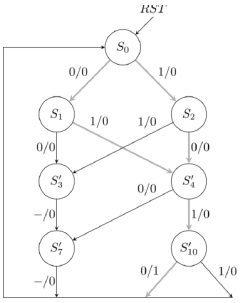
Notando ciò, S10 e S12 vengono accorpati in un unico stato S10′.

Continuando con questo ragionamento, tutti gli stati evidenziati in grigio hanno gli stessi stati futuri e le stesse uscite. Di fatto, essi posso essere riscritti come un unico stato S7′.

		Stato Futuro		Uscita	
Sequenza Ingresso	Stato Presente	$x=0$	$x=1$	$x=0$	$x=1$
RST	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_7	S_8	0	0
01	S_4	S_9	S'_{10}	0	0
10	S_5	S_{11}	S'_{10}	0	0
11	S_6	S_{13}	S_{14}	0	0
000	S_7	S_0	S_0	0	0
001	S_8	S_0	S_0	0	0
010	S_9	S_0	S_0	0	0
011 | 101	S'_{10}	S_0	S_0	1	0
100	S_{11}	S_0	S_0	0	0
110	S_{13}	S_0	S_0	0	0
111	S_{14}	S_0	S_0	0	0

Passaggio intermedio per l'ottimizzazione della tabella degli stati

Si procede in questo modo fino a raggiungere un punto in cui non ci sono più stati equivalenti. Notiamo come, dai 15 stati iniziali, siamo arrivati a solo 8.

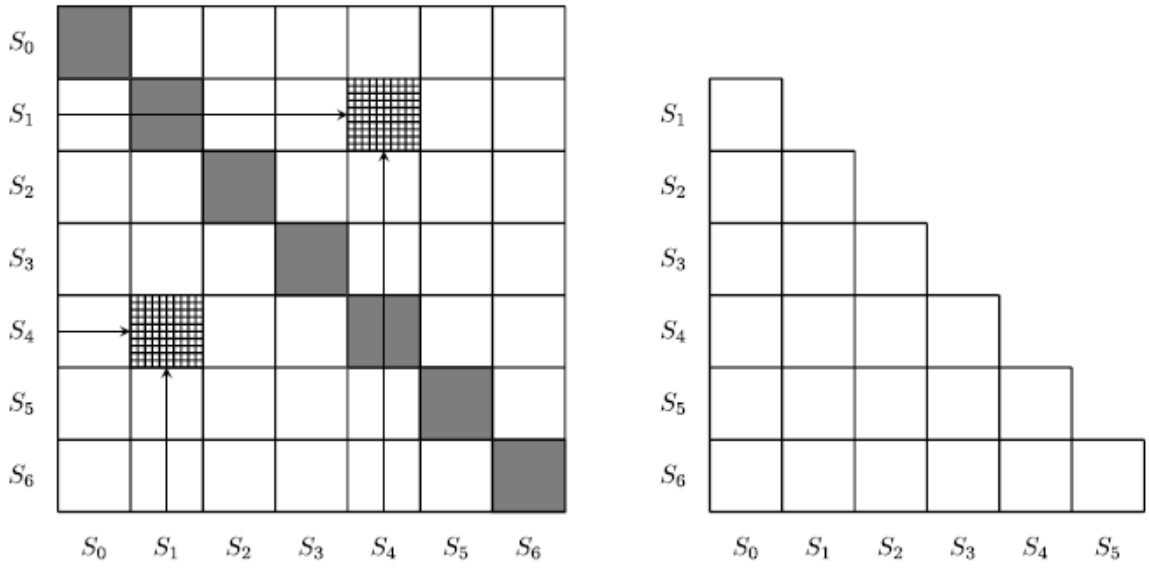


		Stato Futuro		Uscita	
Sequenza Ingresso	Stato Presente	$x=0$	$x=1$	$x=0$	$x=1$
RST	S_0	S_1	S_2	0	0
0	S_1	S'_3	S'_4	0	0
1	S_2	S'_4	S'_3	0	0
00 | 11	S'_3	S'_7	S'_7	0	0
01 | 10	S'_4	S'_7	S'_{10}	0	0
¬(011 | 101)	S'_7	S_0	S_0	0	0
011 | 101	S'_{10}	S_0	S_0	1	0

Tabella degli stati senza stati equivalenti

Metodo della tabella delle implicazioni

La tabella delle implicazioni è una tabella che definisce le relazioni tra tutti gli stati di una FSM. È una tabella con una struttura simmetrica ($S1 \equiv S4 \Leftrightarrow S4 \equiv S1$) e riflessiva ($S4 \equiv S4$) rispetto alla sua diagonale. Per queste proprietà, si semplifica la tabella rimuovendo la diagonale e tutte le celle al di sopra di essa.

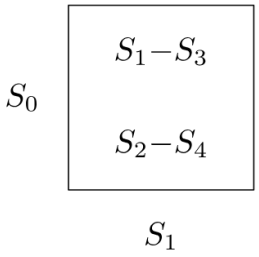


Esempio di struttura della tabella delle implicazioni

Le caselle sono riempite con due coppie di valori:

- la prima coppia di valori definisce gli stati a cui transiziona lo stato definito dall'indice della riga;
  - il primo valore della coppia è lo stato a cui transiziona con ingresso 0;
  - il secondo valore della coppia è lo stato a cui transiziona con ingresso 1.
- la seconda coppia di valori definisce gli stati a cui transiziona lo stato definito dall'indice della colonna.
  - il primo valore della coppia è lo stato a cui transiziona con ingresso 0;
  - il secondo valore della coppia è lo stato a cui transiziona con ingresso 1.

Per esempio, se S0 va a S1 con ingresso 0 e a S2 con ingresso 1, e S1 va a S3 con ingresso 0 e a S4 con ingresso 1, nella cella di indice <S0,S1> ci saranno le coppie S1-S3 e S2-S4.



Esempio di cella <S0, S1>

▼ Esempio - Riempimento e semplificazione di una tabella

Ipotizziamo di avere la seguente tabella degli stati.

		Stato Futuro		Uscita	
Sequenza Ingresso	Stato Presente	$x=0$	$x=1$	$x=0$	$x=1$
RST	S_0	S_1	S_2	0	0
0	S_1	S_3	S_4	0	0
1	S_2	S_5	S_6	0	0
00	S_3	S_0	S_0	0	0
01	S_4	S_0	S_0	1	0
10	S_5	S_0	S_0	0	0
11	S_6	S_0	S_0	1	0

Tabella degli stati iniziale

Possiamo utilizzare il seguente procedimento:

1. definire la tabella delle implicazioni e popolare le sue celle, come descritto precedentemente, eliminando le celle che hanno stati con uscite diverse.

   Ad esempio, S0 ha uscite 00, mentre S4 ha uscite 10, e quindi la cella definita da <S4,S0> viene eliminata, poiché è definita dall'incrocio di due stati non equivalenti;

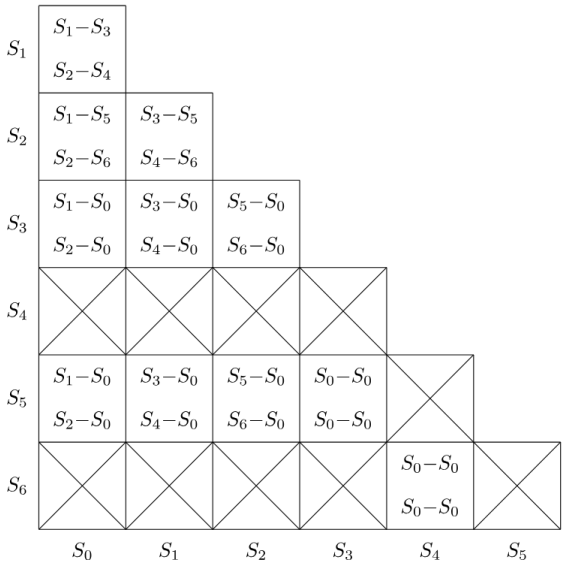


Tabella delle implicazioni iniziale

2. per semplificare ulteriormente la tabella, si passa su ogni casella non cancellata (dall'alto verso il basso e, una volta arrivato alla fine, si passa alla colonna successiva) e si controlla che entrambe le sue coppie di stati futuri siano equivalenti. Se anche solo una delle coppie non è equivalente, allora deve essere cancellata anche quella casella.

Per esempio, partiamo dalla prima casella <S1,S0>, che ha come stati futuri rispettivamente S1-S3 e S2-S4. Si nota come la casella <S1,S3> non definisce che gli stati siano non equivalenti, mentre è ovvio per la casella <S2,S4>, che risulta avere due stati equivalenti.

Dato che basta una coppia di stati futuri non equivalenti, è necessario eliminare anche la casella <S1,S0>;

3. dopo aver passato tutte le celle la prima volta, l'algoritmo deve essere reiterato fino a quando la tabella resta uguale dopo un'iterazione, raggiungendo il fix-point.

   Si raggiunge la conclusione che gli stati S3 e S5 sono equivalenti (S1'), e lo stesso vale per S4 e S6 (S4') e per S2 e S1 (S3').

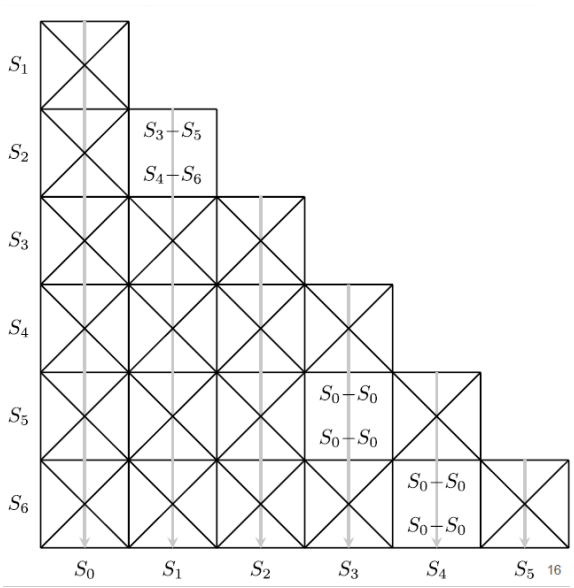


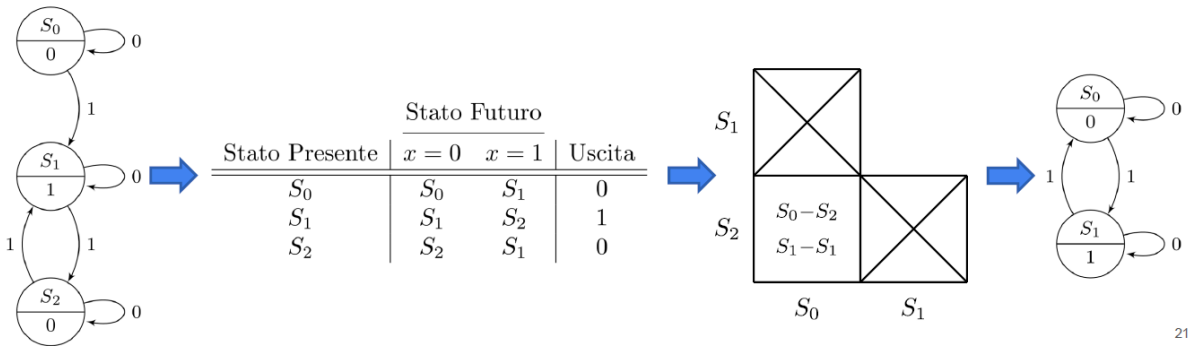
Tabella delle implicazioni al termine della semplificazione

		Stato Futuro		Uscita	
Sequenza Ingresso	Stato Presente	$x=0$	$x=1$	$x=0$	$x=1$
RST	S_0	S_1'	S_1'	0	0
$0 \mid 1$	S_1'	S_3'	S_4'	0	0
$00 \mid 10$	S_3'	S_0	S_0	0	0
$01 \mid 11$	S_4'	S_0	S_0	1	0

Tabella degli stati finale

▼ Esempio - Controllore di parità con tabella delle implicazioni

Applicando il metodo della tabella delle implicazioni, si nota come si arriva subito alla conclusione che S0 e S2 sono stati equivalenti.



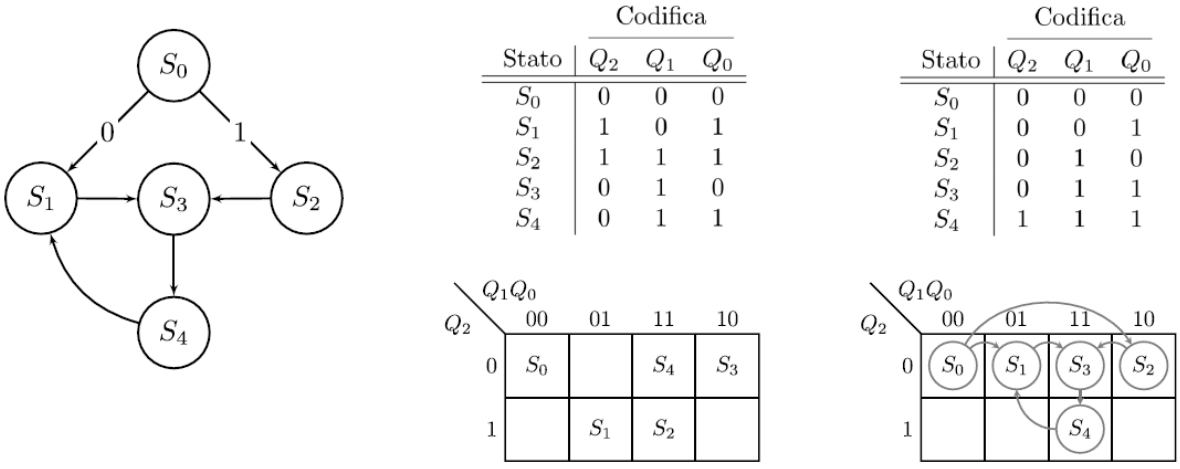
Procedimento di ottimizzazione della FSM con tabella della implicazioni

Codifica degli stati

Codifiche di stato e misura delle metriche con distanza di Hamming

Un ulteriore lavoro per l'ottimizzazione delle FSM può essere fatto scegliendo la codifica degli stati.

In questo caso, ad esempio, perché lo stato S0 deve essere codificato con 000 e non con 011 oppure qualsiasi altro codice?



Esempio di visualizzazione di una FSM

Esiste un criterio che si basa sul calcolo delle distanze di Hamming per definire la preferenza di una codifica degli stati rispetto ad un'altra.

Partendo dall'esempio, si vuole decidere quale delle due codifiche proposte sia migliore. Il calcolo delle distanze di Hamming si basa sulle variazioni di codice di ogni coppia si stati collegati da un arco nella FSM, attraverso i passaggi seguenti:

	Δ_1	Δ_2
$S_0 \rightarrow S_1$	2	1
$S_0 \rightarrow S_2$	3	1
$S_1 \rightarrow S_3$	3	1
$S_2 \rightarrow S_3$	2	1
$S_3 \rightarrow S_4$	1	1
$S_4 \rightarrow S_1$	2	2
Σ	13	7

Calcolo delle distanze di Hamming

1. partendo dalla prima codifica, la coppia S0 (`000`) e S1 (`101`) ha una variazione di codice di 2 bit, ovvero cambiano il primo e il terzo bit, mentre il secondo rimane invariato. Inseriamo nella colonna Δ_1 il valore 2 che appena trovato;
2. per la seconda codifica, la coppia S0 (`000`) e S1 (`001`) ha una sola variazione, ovvero il terzo bit. Aggiungiamo 1 nella colonna Δ_2.
3. svolgiamo lo stesso ragionamento per tutti gli altri stati collegati da un arco e riempiamo le due colonne;
4. la distanza di Hamming della codifica è la sommatoria di tutti i valori della colonna corrispondente. Per la prima codifica, la sommatoria dei valori di Δ_1 è 13, mentre per Δ_2 è 7.
5. la codifica migliore è quella con distanza di Hamming minore, che, nel nostro caso, risulta essere la seconda.

Approccio basato sulla mappa di Karnaugh:

Si può utilizzare una mappa di Karnaugh per disporre gli stati in modo da evidenziare le adiacenze logiche e per trovare una codifica ottimizzata, dove le

transizioni tra stati comportano variazioni minime nei bit. Si possono eseguire i seguenti passi:

1. rappresentare gli stati della FSM come celle di una mappa di Karnaugh;
2. posizionare gli stati in modo che quelli con transizioni frequenti, collegati da archi nella FSM, siano adiacenti nella mappa, come nella tabella di Karnaugh in basso a destra dell'esempio;
3. assegnare codici binari alle celle, sfruttando la proprietà della mappa di Karnaugh, per cui celle adiacenti differiscono di un solo bit;
4. dopo tutti questi passaggi, si ottiene una codifica che minimizza le variazioni di bit durante le transizioni tra stati frequentemente collegati.

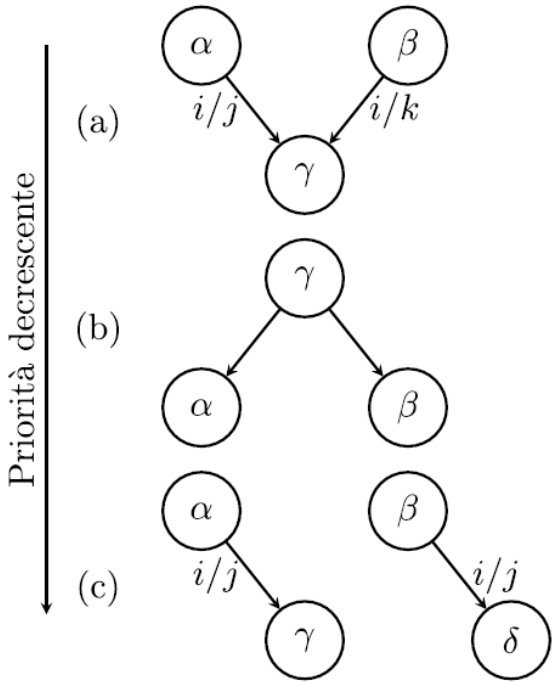
Metodi euristici

Quando le FSM diventano più complesse rispetto all'esempio precedente, si assegnano codici adiacenti privilegiando certe condizioni, con priorità differenti:

- stati con lo stesso stato futuro (a);
- stati con lo stesso stato precedente (b);
- stati con la stessa uscita (c).

La logica dietro questa scelta è che con (a) e (b) riusciamo a raggruppare gli 1 nella mappa dello stato futuro, mentre con (c) raggruppiamo gli 1 nella mappa delle uscite.

Si ricordi come avere tanti 1 vicini nelle mappe di Karnaugh ci permette di realizzare circuiti più semplici.

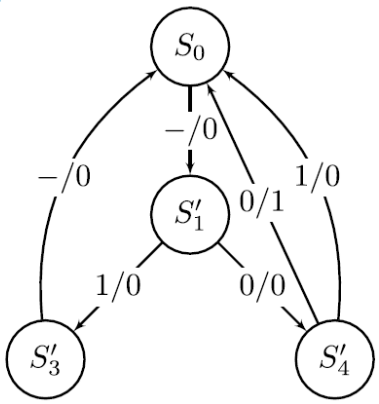


Condizioni di codici adiacenti

Esempio - Metodi euristici 1

Ipotizziamo di avere la seguente situazione:

- FSM di riferimento:
  - Identificatore di sequenza a 3 bit
- Priorità *massima* per lo *stesso stato futuro* $(S_3' - S_4')$
- Priorità *media* per lo *stesso stato precedente* $(S_3' - S_4')$
- Priorità *minima* per la *stessa uscita*
  $0/0$: (S_0, S_1', S_3')
  $1/0$: (S_0, S_1', S_3', S_4')



Q_1 \ Q_0	0	1
0	S_0	S_1'
1	S_3'	S_4'

Q_1 \ Q_0	0	1
0	S_0	S_3'
1	S_1'	S_4'

Descrizione della situazione

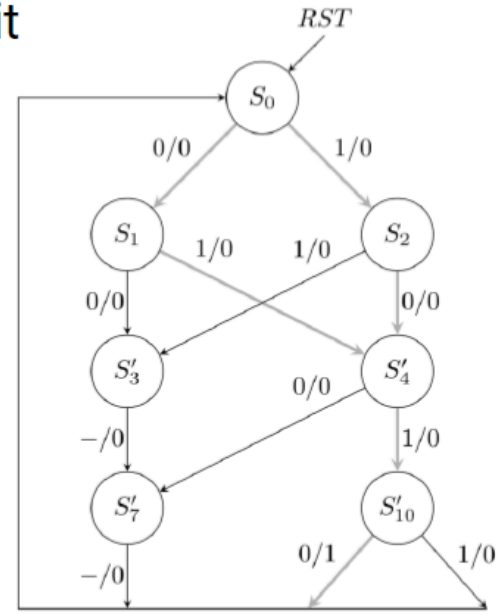
- S3′ e S4′ hanno entrambi lo stato futuro S0;
- S3′ e S4′ hanno entrambi lo stesso predecessore;
- 0/0 identifica la ricezione in ingresso di uno 0 restituendo in uscita 0, mentre 1/0 identifica la ricezione in ingresso di un 1 e restituendo in uscita 0.

Mettiamo S0 come 00, in modo da utilizzarlo come una sorta di reset, e ci rimangono due scelte per posizionare S3′ e S4′ vicini. Nell'esempio, le due possibili mappe di Karnaugh sono rappresentate nel lato destro.

Esempio - Metodi euristici 2

Ipotizziamo di avere la seguente situazione:

- FSM di riferimento: riconoscitore di stringa a 4 bit
- Priorità *massima* per lo *stesso stato futuro*
  - $(S'_3 - S'_4), (S'_7 - S'_{10})$
- Priorità *media* per lo *stesso stato precedente*
  - $(S_1 - S_2), (S'_3 - S'_4), (S'_7 - S'_{10})$
- Priorità *minima* per la *stessa uscita*
  - 0/0: (S_0, S_1, S_2, S'_3, S'_4, S'_7)
  - 1/0: (S_0, S_1, S_2, S'_3, S'_4, S'_7)

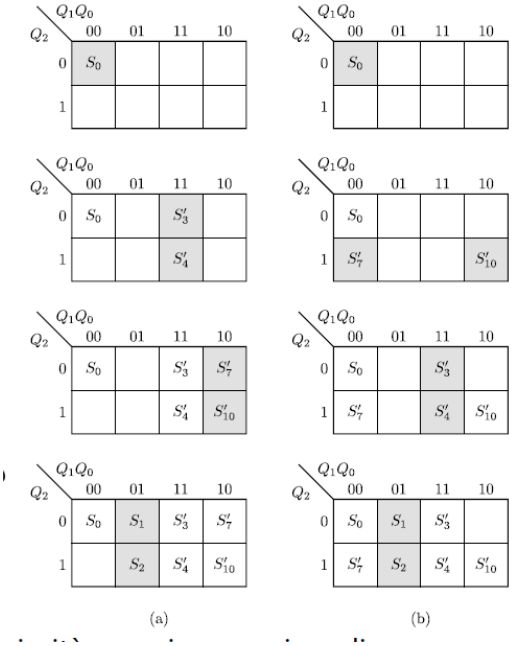


Descrizione della situazione

Si noti come abbiamo raggruppato le coppie di stati che dovremmo mettere vicine nella FSM con priorità decrescente.

Descriviamo la soluzione dell'esempio:

- utilizziamo S0 codificato con 000 come RST;
- nell'immagine a destra, sono presenti due esempi lato a lato di riempimento differenti, inserendo prima le coppie di stati con priorità massima e successivamente quelli con priorità minimi;
- una codifica rimane senza stato, perciò questa cella potrebbe essere usata in modo strategico, in modo da avere valori "don't care" nelle mappe di Karnaugh degli stati futuri e delle uscite, in modo da semplificare ulteriormente il circuito.

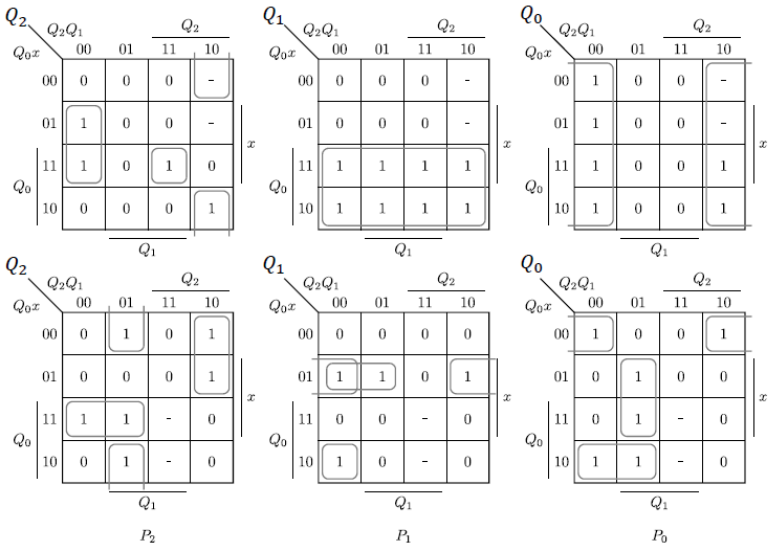


Esempi di riempimento delle mappe di Karnaugh lato a lato

Passando dalla nostra codifica degli stati alle relative mappe di Karnaugh dell'uscita e degli stati futuri, si nota che la prima codifica raggruppa meglio i valori 1 nella mappa dello stato futuro.

Stato Presente		Stato Futuro $x=0$	Stato Futuro $x=1$
(S_0)	000	001	101
(S_1)	001	011	111
(S_2)	101	111	011
(S'_3)	011	010	010
(S'_4)	111	010	110
(S'_7)	010	000	000
(S'_{10})	110	000	000

Stato Presente		Stato Futuro $x=0$	Stato Futuro $x=1$
(S_0)	000	001	010
(S_1)	001	011	100
(S_2)	010	100	011
(S'_3)	011	101	101
(S'_4)	100	101	110
(S'_7)	101	000	000
(S'_{10})	110	000	000



Confronto delle soluzioni ottenute dai due diversi riempimenti