Funzioni uscita e stato prossimo

- L’uscita e lo stato prossimo sono funzioni della sequenza di ingressi applicata a partire da uno “stato iniziale”:

$$u_k = \lambda(q_0, J_k)$$
$$q_{k+1} = \delta(q_0, J_k)$$

Macchine complete e incomplete

Applicabilità di una sequenza

- Una sequenza di ingressi J è *applicabile* a M in q - *si dice a M(q)* - se è definita *u=λ(q, J), la funzione che fornisce l'uscita u che si ottiene applicando alla macchina la sequenza di ingressi J a partire da uno "stato iniziale" q.*
- Se la funzione di uscita *λ* è definita per tutte le coppie (q,i), la macchina si dice **completa**, altrimenti si dice **incompleta**
- **Per le macchine incomplete esistono sequenze non applicabili**
  - Sequenza J_j *non applicabile* in q_0: *λ* non definita

Equivalenza fra stati

- Occorre formalizzare il fatto che due macchine possano avere lo stesso funzionamento

  → reagire nello stesso modo (con le stesse uscite) alle stesse sequenze di ingressi
- Definizione di ***stati equivalenti*** in macchine complete*:*
  - **Producono la stessa sequenza di uscite per qualsiasi sequenza di ingressi**

Equivalenza fra stati

- Un modo per riconoscere stati equivalenti (fondamentale negli algoritmi che vedremo) è usare la proprietà ricorsiva degli stati equivalenti:

  Due stati sono equivalenti se lo sono tutte le possibili coppie di stati successivi, e sono uguali tutte le possibili uscite successive.

Equivalenza fra stati: riassumendo

- Concetto di equivalenza: avere lo stesso funzionamento “esterno”
  - Reagire nello stesso modo (con le stesse uscite) alle stesse sequenze di ingressi
  - Due stati sono equivalenti se, per ciascun ingresso:
    - sono eguali le uscite
    - sono equivalenti gli stati successivi
  - Definizione di *stati equivalenti* in macchine **complete***:*
    - Producono la stessa sequenza di uscite per qualsiasi sequenza di ingressi

Equivalenza fra macchine complete

- I due stati possono appartenere anche non alla stessa macchina
- Due macchine *complete* M e M' sono **equivalenti** se per ciascuno stato q di M esiste almeno uno stato q’ di M’ ad esso equivalente e, viceversa

Equivalenza e macchine incomplete

- Per macchine incomplete la definizione precedente non può essere applicata così com'è

  → non tutte le possibili sequenze sono applicabili a tutti gli stati
- Si introducono i concetti di:
  - ***Compatibilità*** tra stati:
    - Due stati q e q' sono *incompatibili* se esiste almeno una sequenza di ingresso applicabile ad entrambi per cui le uscite siano differenti. Sono *compatibili* se rispondono a medesime sequenze di ingresso (definite per entrambi) con le medesime sequenze di uscita
  - ***Inclusione*** tra macchine:
    - Una macchina M' include la macchina M se, per ciascuno stato q di M, ne esiste uno q' di M' tale che qualsiasi sequenza applicabile a M in q sia applicabile anche a M' in q' con le stesse uscite finali

Compatibilità ed equivalenza

- Nel caso di macchine complete le due definizioni coincidono.
- Tra due macchine equivalenti, conviene scegliere quella con meno stati

  → problema di **minimizzazione**

  individuare la macchina con il minor numero di stati tra tutte le possibili macchine equivalenti

Classi di equivalenza/compatibilità

- Il procedimento di minimizzazione si basa sulla determinazione delle cosiddette classi di equivalenza (macchine complete) o di compatibilità (macchine incomplete), definite come gli insiemi di stati fra loro equivalenti/compatibili
  - Una classe di equivalenza/compatibilità è massima se non è contenuta in nessun' altra
- La famiglia di massima compatibilità F è data dall'insieme delle classi di compatibilità massime della macchina M
  - Ogni classe C di F rappresenta uno stato che ha come uscita l'uscita degli elementi di C, tutti eguali per essere questi equivalenti/compatibili fra loro
  - Lo stato seguente di ciascuna classe C di F sarà quello determinato da C stesso; essendo infatti gli stati di C equivalenti fra loro, devono avere tutti per seguenti stati equivalenti che appartengono ad un unico elemento C' della famiglia delle classi di equivalenza.
- Una macchina M' che abbia come insieme degli stati l'insieme di massima compatibilità F di M (completa), è equivalente a M ed ha un numero di stati in generale ridotto (al più uguale) rispetto a M

Minimizzazione degli stati in macchine incomplete

- Nel caso di macchine complete, le classi di equivalenza sono disgiunte, mentre nel caso di macchine incomplete potrebbero non esserlo
- La macchina M’ ottenuta ricercando la famiglia di massima compatibilità include M, non è equivalente ad essa.
- La soluzione potrebbe non essere minima, ma in generale presenta una macchina a “stati ridotti”
- Si possono avere più soluzioni, potendo uno “stato seguente” essere incluso in più elementi distinti della famiglia di compatibilità massima

Problema della Minimizzazione

- Partendo da una macchina $M(Q,I,U,\tau,\omega)$, ne vogliamo trovare a macchina $M'(Q',I,U,\tau',\omega')$ equivalente ad M e con il minor numero di stati
- Dobbiamo determinare la famiglia di massima compatibilità (insiemi di stati compatibili massimi) $F=(S_1,S_2,\ldots,S_n)$

Problema della Minimizzazione

- La F gode delle seguenti proprietà, essenziali nei metodi di minimizzazione:
  - Gli elementi S di F sono disgiunti (macchine complete)
  - Gli elementi S di F coprono l'insieme degli stati Q
  - Tutti gli stati di un elemento S di F portano alla stessa uscita (eventualmente non definita)
  - F è chiusa: da due stati di uno stesso elemento S di F si arriva a due stati che appartengono ad una stessa S' di F
- Ricerchiamo la $M'(F,I,U,\tau',\omega')$

  → M' ha un numero di stati non superiore a M

Ricerca della famiglia F

- Algoritmo del partizionamento
- Metodo tabellare di Paull-Unger

- Procedono per “eliminazione”

  Partono da una presunta F (inizialmente contenente un solo insieme coincidente con Q) e cercano di individuare incompatibilità fin quando è possibile

Algoritmo del partizionamento

- Si individuano gli stati incompatibili rispetto alle uscite per ciascun ingresso
- Le partizioni individuate si esaminano rispetto allo stato prossimo
- Si itera fintantoché tutte le partizioni non verificano la definizione di compatibilità

Algoritmo del partizionamento

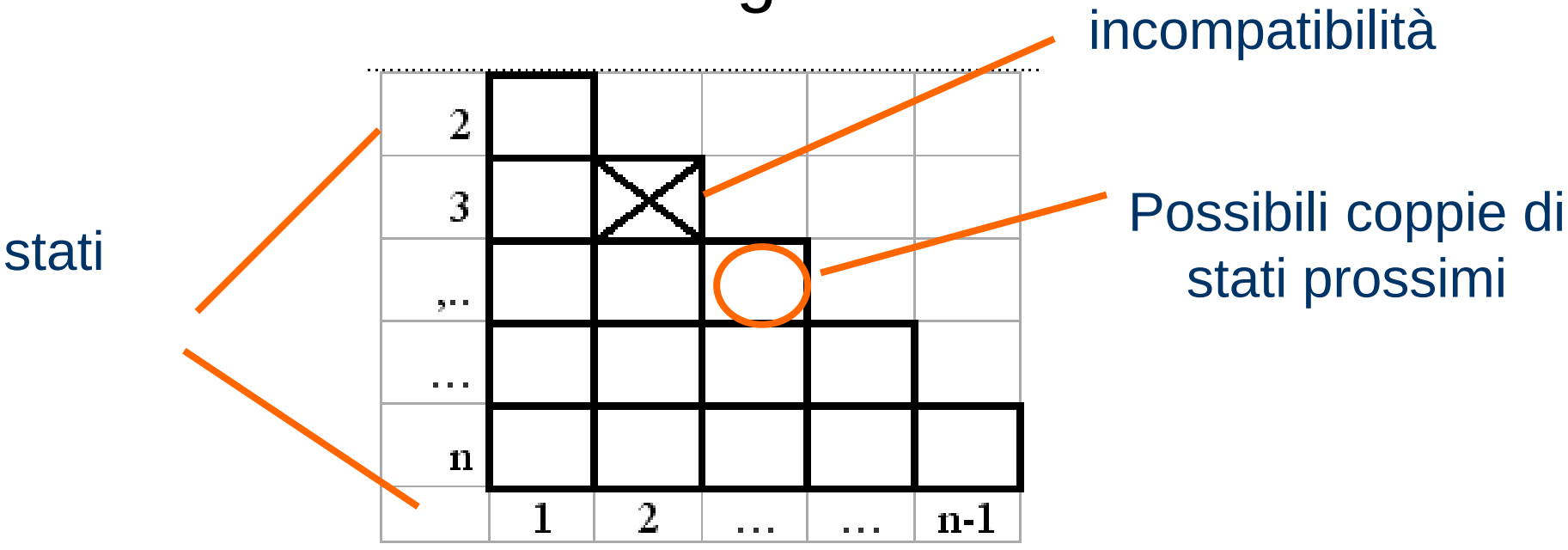
statii	i_1	i_2
q_1	q_2/u_1	q_7/u_2
q_2	q_4/u_2	q_7/u_1
q_3	q_1/u_1	q_5/u_2
q_4	q_2/u_2	q_3/u_1
q_5	q_4/u_1	q_3/u_2
q_6	q_1/u_2	q_2/u_2
q_7	q_5/u_1	q_5/u_2

i	Elemento in esame	Analisi uscite	Partizione elementi	Famiglia
				(1,2,3,4,5,6,7)
1	(1,2,3,4,5,6,7)	u_1: (1,3,5,7); u_2:(2,4,6)	(1,3,5,7) (2,4,6)	(1,3,5,7) (2,4,6)
2	(1,3,5,7)	u_2: (1,3,5,7)		
2	(2,4,6)	u_1:(2,4); u_2: (6)	(2,4) (6)	(1,3,5,7) (2,4,)(6)

i	Stati seguenti	Analisi stati seguenti	Partizione elemento	Famiglia
Passo 3				(1,3,5,7) (2,4) (6)
1	(1,3,5,7)→(2,1,4,5)	(2,4) (1,5)→(1,5) (3,7)	(1,5) (3,7)	(1,5) (3,7) (2,4) (6)
1	(2,4)→(4,2)			
2	(2,4)→(7,3)			

Metodo di Paull-Unger

- Riorganizza il procedimento visto prima in forma di matrice diagonale

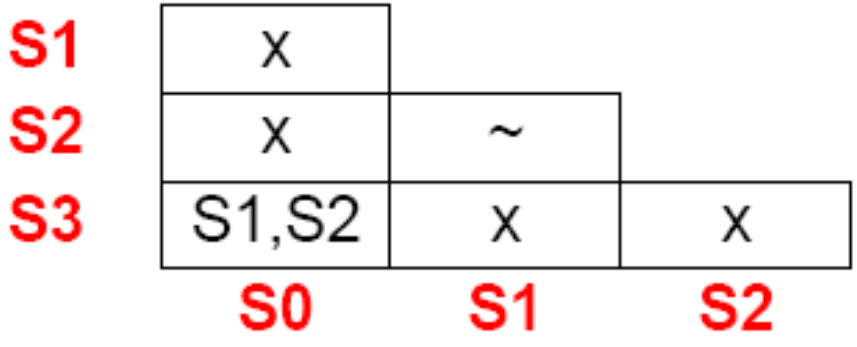


Metodo di Paull-Unger

- Si marcano come incompatibili le coppie di stati che portano ad uscite differenti per almeno un ingresso
- Si indicano le coppie di possibili stati prossimi in ogni casella
- Si continua iterativamente il procedimento partizionando rispetto agli stati

Metodo di Paull-Unger

- Ogni elemento della Tabella delle Implicazioni contiene:
  - Il simbolo di non equivalenza (X)
  - Il simbolo di equivalenza (~)
  - La coppia di stati *condizionanti* se non è possible stabilire immediatamente l'equivalenza (o non equivalenza)



Esempio – Paull-Unger

	0	1
a	g/0	e/1
b	c/0	a/1
c	e/1	g/0
d	b/0	e/1
e	g/0	a/1
f	d/1	f/0
g	a/1	g/0

	b	c	d	e	f
b	cg ae				
c	x	x			
d	bg	bc ae	x		
e	~	cg	x	bg ae	
f	x	x	de fg	x	x
g	x	x	ae	x	x	ad

"e" ed "a" hanno le stesse uscite per ogni ingresso

"g" ed "f" sono equivalenti se lo sono "a" e "d"

"g" e "b" hanno uscite differenti (con rif ad uno stesso ingresso)

Esempio - cont

- Procedendo iterativamente si giunge a determinare le classi di equivalenza

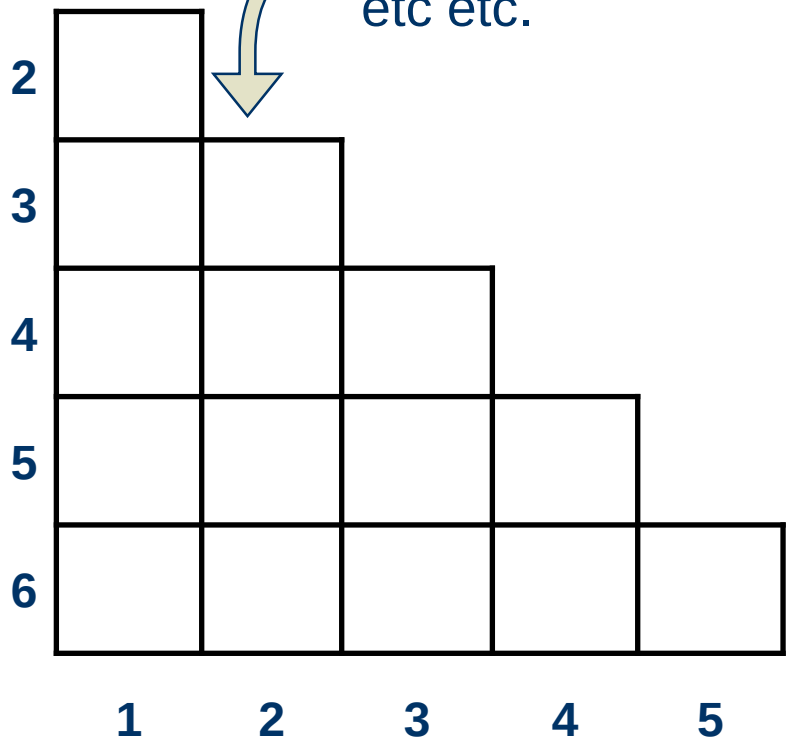
	a	b	c	d	e	f
b	~					
c	x	x				
d	x	x	x			
e	~	~	x	x		
f	x	x	x	x	x	
g	x	x	~	x	x	x

α = {a,b,e}
β = {c,g}
γ = {d}
δ = {f}

Metodo di Paull-Unger

	i_1	i_2
1	2/A	4/B
2	3/B	5/A
3	5/A	4/B
4	2/B	2/B
5	2/B	5/A
6	3/A	5/B

Ogni stato è confrontato con tutti gli altri, ma solo una volta:
1 è confrontato con 2…6
2 è confrontato con 3…6
etc etc.



Macchine Incompletamente Specificate

Stati compatibili

- Due stati sono **compatibili** se *per ogni* sequenza di ingressi applicabili ad *entrambi,* le uscite prodotte sono identiche
- A differenza della relazione vista prima per macchine complete, la compatibilità **NON** è una relazione di equivalenza:

  Gode delle proprietà
  - Riflessiva
  - Simmetrica
  - Ma NON di quella transitiva
    - p.e.: q1~q2, q2~q3, λ(q1,J)=a, , λ(q2,J)=-, λ(q3,J)=b

- Ciò complica la ricerca di macchine equivalenti minime

Compatibilità

- Così come l'equivalenza, anche la compatibilità può essere definita ricorsivamente
- **Due stati sono compatibili se tutti i possibili stati prossimi sono compatibili, e tutte le possibili uscite prossime sono uguali**
- Non essendo una relazione di equivalenza, non è possibile utilizzare le proprietà delle classi di equivalenza.
- Si generalizza con il concetto di *famiglia di insiemi di stati compatibili massimi*

Compatibilità

- Per le macchine incomplete, non si parla quindi di equivalenza, ma di inclusione:
  - Una macchina M' ne include una M, *in una coppia di stati q* e *q' ,* se tutte le sequenze di ingressi applicabili ad M a partire da q lo sono anche per M' a partire da q' producendo la stessa uscita
  - Se è possibile trovare per ciascuno stato di M uno q che soddisfa la precedente definizione, allora M' include M → è possibile usare M' in luogo della M
  - M ed M' possono includersi l'un l'altra

    → diremo in questo caso che le due macchine sono equivalenti

Compatibilità: formalmente

Inclusione fra macchine

- **Concettualmente:** una macchina include un'altra se "fa qualcosa in più"

- **Formalmente:**
  - M'(q') include M(q):

$$M'(q') \supseteq M(q) \Leftrightarrow \forall \left(J \text{ applicabile a } M(q)\right) : \lambda(q', J) = \lambda(q, J)$$

  - M' include M

$$M' \supseteq M \Leftrightarrow \forall (q \in M)\, \exists (q' \in M') : M'(q') \supseteq M(q)$$

Macchina incompletamente specificata

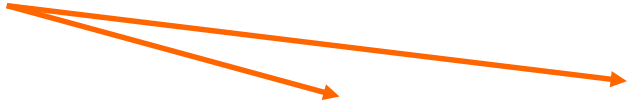
- Compatibilità: si ricorda che…
  - Due stati **si** e **sj** si dicono compatibili se partendo da essi ed applicando ogni possibile sequenza di ingresso applicabile **Iα** si ottengono le stesse sequenze d'uscita, ovunque queste siano specificate.
  - La relazione di compatibilità non è transitiva!!!
  - Le classi di compatibilità NON sono disgiunte
  - Non è possibile applicare il metodo della fusione delle righe (row merging)

Macchina incompletamente specificata

- Costruzione della macchina minima o a stati ridotti
  - E’ più complessa che nel caso delle macchine completamente specificate.
  - Gli insiemi di F non sono disgiunti!
  - Un insieme T può essere incluso in più di un insieme di F
    - la macchina che si ottiene associando uno stato ad ogni distinto insieme della famiglia F non è necessariamente una macchina a numero minimo di stati.

Algoritmo della suddivisione degli elementi della famiglia di presunta compatibilità

- Analogo al metodo del partizionamento visto per reti complete
- Data una famiglia, si esaminano singolarmente gli insiemi presumibilmente compatibili
  - se in uno di questi, S_k, si identificano due stati incompatibili, qi e qi, Sk viene sostituito da due nuovi insiemi, l'uno contenente tutti gli elementi tranne q_i e l'altro tutti tranne q_j, dando così luogo ad una nuova famiglia
- Ad es.
  - $S_k = (q_l, q_2, q_3, q_4)$; se successivamente si verifica che q_1 e q_2 sono incompatibili



$$S_{k1} = (q_l, q_3, q_4) \; ; S_{k2} = (q_2, q_3, q_4)$$

Algoritmo della suddivisione degli elementi della famiglia di presunta compatibilità

- TRE PASSI:
  - *Passo 1*
    - Si eliminano le incompatibilità immediatamente evidenti, (per tutti gli ingressi i_k).

      $$\omega(q_i, i_k) \neq \omega(q_j, i_k)$$
  - *Passo 2 (aggiunto rispetto al partizionamento)*
    - Si individua una famiglia di insiemi "presumibilmente compatibili",

      $$F = (S_1, S_2, \ldots, S_m),$$

      eliminando gli elementi inclusi in altri
  - *Passo 3*
    - Si eliminano le incompatibilità dovute ad incompatibilità negli stati seguenti. Per ciascun ingresso ik e ciascun insieme $Si = (q_{il}, q_{i2}, \ldots, q_{in})$, si forma l'insieme degli stati seguenti

      $$T(S_i, i_k) = (\tau(q_{i1}, i_k), \tau(q_{i2}, i_k), \ldots, \tau(q_{in}, i_k))$$
    - Se $T(S_i, i_k) \notin S_j$, si individua l'incompatibilità, si procede alla suddivisione di *Si* e si ritorna al passo 2.
    - Se $T(S_j, i_k) \in S$ per tutti gli elementi di F e per tutti gli ingressi, il procedimento ha termine.

Macchina incompletamente specificata

- Paull Unger esteso
  - Le relazioni di compatibilità possono essere ricavate dalla **Tabella delle implicazioni**
    - viene compilata come avviene nel caso delle macchine completamente specificate
  - Ogni elemento della tabella contiene:
    - Il simbolo di compatibilità se gli stati corrispondenti sono compatibili
    - Il simbolo di non compatibilità se gli stati corrispondenti non sono compatibili
    - Le coppie di stati a cui si rimanda la verifica se non è possibile stabilire la compatibilità

Macchina incompletamente specificata

- Passi:
  1. Si barrano le caselle i cui stati seguenti sono incompatibili (piuttosto che disequivalenti)
  2. Si determinano le coppie compatibili come per le macchine complete
  3. ***Si costruisce la famiglia degli insiemi massimi (F)***
     1. Si prendono in esame ordinatamente tutti gli stati della macchina e, per ciascuno di essi (q):
        - si aggiungono ad F tutte le coppie di compatibilità di q;
        - q viene confrontato con tutti gli stati di ciascun elemento E di F e se compatibile con tutti, E si accresce di q, se compatibile solo con alcuni, si genera un nuovo elemento di F
        - si eliminano da F gli elementi inclusi in altri

Esempi di minimizzazioni

Macchine completamente specificate

Macchine incompletamente specificate

Esempio 1: Riconoscitore di codice 8-4-2-1

- Costruire una rete nella quale entrano serialmente i bit di un codice decimale 8-4-2-1 a partire dal bit meno significativo e dalla quale esce un segnale impulsivo che individua se i quattro bit costituiscono o meno una delle 10 parole-codice previste

cifra	8-4-2-1
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Riferimento: “Reti logiche- Complementi ed Esercizi” CAP 5, es n.6

Esempio 1

- Procediamo per elencazione di tutti le possibili sequenze
  - Individuiamo tutti i possibili stati
  - Partizioniamo rispetto alle uscite

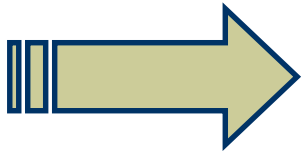
	1	0
0	1/0	2/0
1	3/0	4/0
2	5/0	6/0
3	7/0	8/0
4	9/0	10/0
5	11/0	12/0
6	13/0	14/0
7	0/0	0/1
8	0/0	0/1
9	0/0	0/1
11	0/0	0/1
12	0/0	0/1
13	0/0	0/1
10	0/1	0/1
14	0/1	0/1

	1	0
0	1/0	2/0
1	3/0	4/0
2	5/0	6/0
3	7/0	8/0
4	9/0	10/0
5	11/0	12/0
6	13/0	14/0
7	0/0	0/1
8	0/0	0/1
9	0/0	0/1
10	0/1	0/1
11	0/0	0/1
12	0/0	0/1
13	0/0	0/1
14	0/1	0/1

Esempio 1

- Eliminiamo le righe uguali
  - ne resta soltanto una e lo stato non eliminato (ad es. 7) viene sostituito a tutti quelli eliminati nella colonna degli stati futuri

	1	0
0	1/0	2/0
1	3/0	4/0
2	5/0	6/0
3	7/0	8/0
4	9/0	10/0
5	11/0	12/0
6	13/0	14/0
7	0/0	0/1
~~8~~	~~0/0~~	~~0/1~~
~~9~~	~~0/0~~	~~0/1~~
~~11~~	~~0/0~~	~~0/1~~
~~12~~	~~0/0~~	~~0/1~~
~~13~~	~~0/0~~	~~0/1~~
10	0/1	0/1
~~14~~	~~0/1~~	~~0/1~~



	1	0
0	1/0	2/0
1	3/0	4/0
2	5/0	6/0
3	7/0	7/0
4	9/0	10/0
5	11/0	7/0
6	13/0	10/0
7	0/0	0/1
10	0/1	0/1

non 8!!*

non 14!!

* Lo stesso vale per 9, 11, 12 e 13

Esempio 1

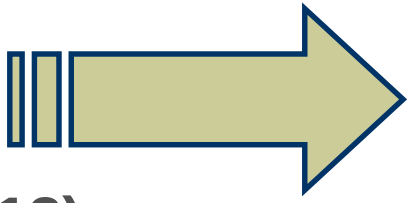
- Tracciamo la tabella delle implicazioni

1	1-3;2-4							
2	1-5;2-6	3-5;4-6						
3	1-7;2-7	3-7;4-7	5-7;6-7					
4	1-7;2-10	3-7;4-10	5-7;6-10	7-10				
5	1-7;2-7	3-7;4-7	5-7;6-7	~	7-10			
6	1-7;2-10	3-7;4-10	5-7;6-10	7-10	~	7-10		
7	X	X	X	X	X	X	X	
10	X	X	X	X	X	X	X	X
	0	1	2	3	4	5	6	7

Esempio 1

- Classi di equivalenza
  - Ricordando anche gli stati “fusi” in precedenza

S0=(0),
S1=(1,2),
S2=(3,5),
S3=(4,6),
S4= **(7,8,9,11,12,13),**
S5= (10,14).



	1	0
S0	S1/0	S1/0
S1	S2/0	S3/0
S2	S4/0	S4/0
S3	S4/0	S5/0
S4	S0/0	S0/1
S5	S0/1	S0/1

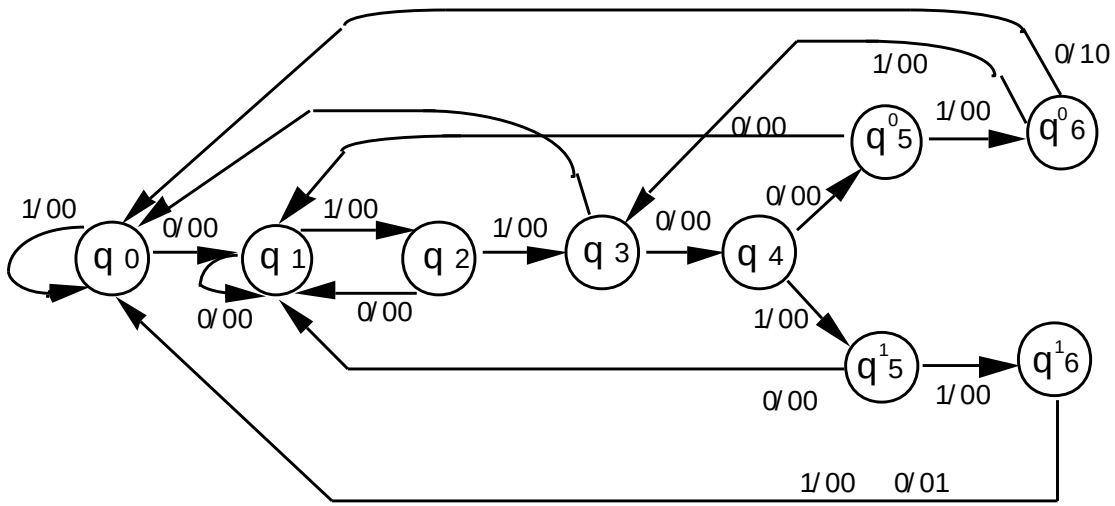
Tabella degli stati ridotti

Esempio 2: Riconoscitore di due sequenze

- Rete sincrona con un solo ingresso X e in grado di riconoscere le due sequenze: 0110010 e 0110110
- Le uscite devono essere codificate in modo da distinguere i tre casi di possibile riconoscimento di sequenze:
  - 1) 0110010; 2) 0110110; 3) tutte le altre
    - - Y=0, Z=0: non è stata trovata nessuna sequenza;
    - - Y=1: Z=0 riconosciuta la sequenza 0110010;
    - - Z=1: Y=0 riconosciuta la sequenza 0110110;
    - - Y=Z=1: uscita non utilizzata

Esempio 2: Riconoscitore di due sequenze

- q0 stato iniziale;
- qi (i < 5): riconosciuto un numero *i* di caratteri della seq. `0110---`
- q^0i (i ≥ 5): riconosciuto un numero *i* di caratteri della seq. `0110010`
- q^1i (i ≥ 5): riconosciuto un numero *i* di caratteri della seq. `0110110`

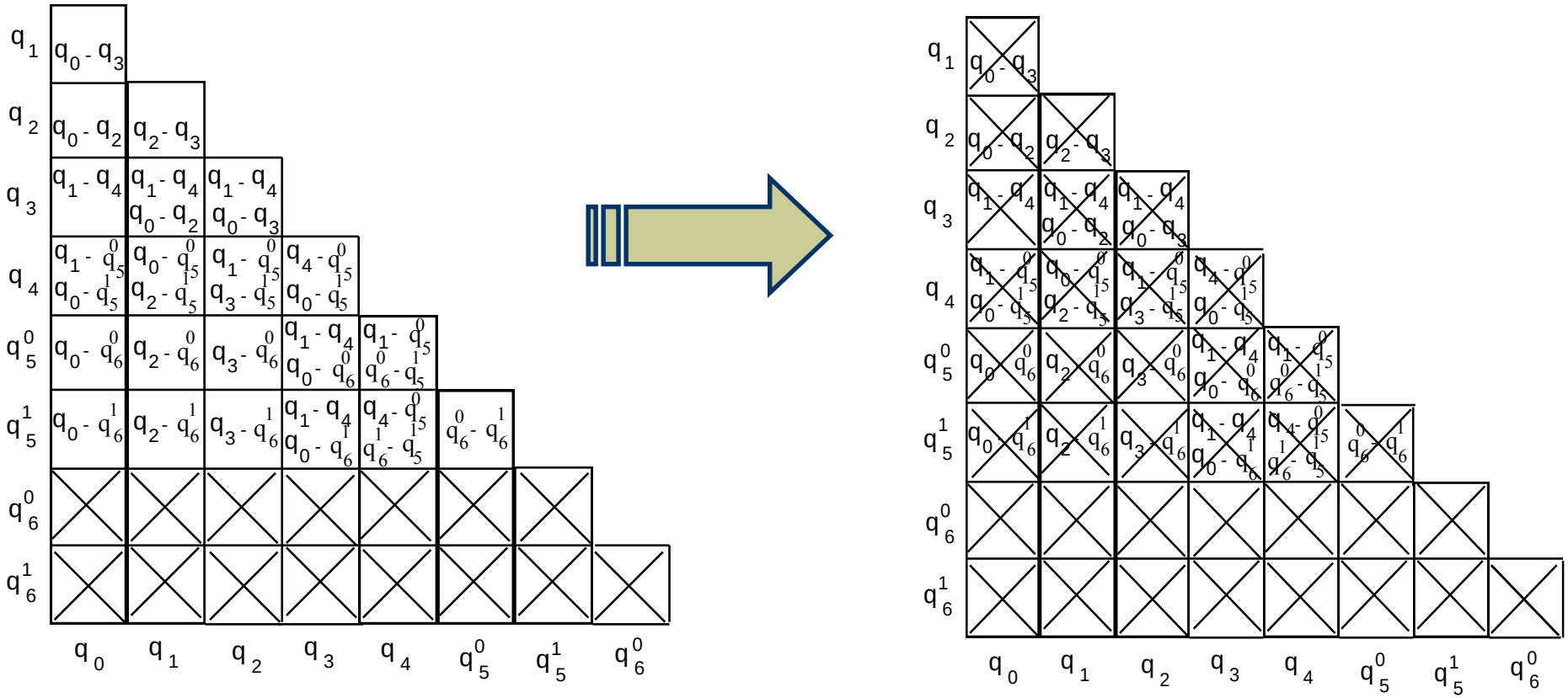


Esempio 2: Riconoscitore di due sequenze

stati	X = 0	X = 1
q_0	q_1 / 00	q_0/ 00
q_1	q_1/ 00	q_2/ 00
q_2	q_1/ 00	q_3/ 00
q_3	q_4/ 00	q_0/ 00
q_4	q_5^0/ 00	q_5^1/ 00
q_5^0	q_1/ 00	q_6^0/ 00
q_5^1	q_1/ 00	q_6^1/ 00
q_6^0	q_0/ 10	q_3/ 00
q_6^1	q_0/ 01	q_0/ 00

stati	X = 0	X = 1
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_4	q_0
q_4	q_5^0	q_5^1
q_5^0	q_1	q_6^0
q_5^1	q_1	q_6^1
q_6^0	q_0	q_3
q_6^1	q_0	q_0

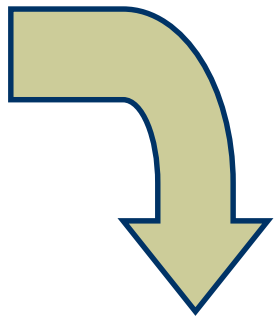
Esempio 2: Riconoscitore di due sequenze



Non esistono stati compatibili! La tabella di flusso è già minima

Esempio 3: Algoritmo della suddivisione degli elementi della famiglia di presunta compatibilità

	i_1	i_2	i_3	i_4
q_1	q_1/u_1	$q_2/-$	-/-	$q_6/-$
q_2	$q_1/-$	q_2/u_2	$q_5/-$	-/-
q_3	q_3/u_1	$q_4/-$	-/-	$q_6/-$
q_4	$q_1/-$	q_4/u_2	$q_5/-$	-/-
q_5	-/-	$q_7/-$	q_5/u_3	$q_6/-$
q_6	$q_3/-$	-/-	$q_5/-$	q_6/ u_4
q_7	$q_3/-$	q_7/u_4	$q_5/-$	-/-



PASSI 1 e 2

i	Elemento in esame	Analisi uscite	Suddivisione elemento	Famiglia
Passo 1				**(1,2,3,4,5,6,7)**
2	**(1,2,3,4,5,6,7)**	**$u_2 \neq u_4 \rightarrow 2 \not\sim 7$**	**(1,2,3,4,5,6)(1,3,4,5,6,7)**	**(1,2,3,4,5,6) (1,3,4,5,6,7)**
2	**(1,3,4,5,6,7)**	**$u_2 \neq u_4 \rightarrow 4 \not\sim 7$**	**(1,3,5,6,7) (1,3,4,5,6)**	**(1,2,3,4,5,6)(1,3,5,6,7)(1,3,4,5,6)**
Non essendovi altre incompatibilità a causa delle uscite, il passo 1 è terminato				
Passo 2		**(1,3,4,5,6) $\subset$ (1,2,3,4,5,6)**		**(1,2,3,4,5,6) (1,3,5,6,7)**

Esempio 3: Algoritmo della suddivisione degli elementi della famiglia di presunta compatibilità

PASSO 3 – da ripetere fino a che non siano più presenti incompatibilità

i	Stati seguenti	Stati incompatibili	Suddivisione elemento	Famiglia
Passo 3				(1,2,3,4,5,6) (1,3,5,6,7)
1	(1,2,3,4,5,6)→(1,1,3,1,-,3)	nessuna		
1	(1,3,5,6,7)→(1,3,-,3,3)	nessuna		
2	(1,2,3,4,5,6)→(2,2,4,4,7,-)	2 $\not\gamma$ 7→1,2 $\not\gamma$ 5 4 $\not\gamma$ 7→3,4 $\not\gamma$ 5	((1,2,3,4,6) (5,6)	
Passo2	(5,6) ⊂ (1,3,5,6,7)			(1,2,3,4,6) (1,3,5,6,7)
Passo 3				
3	(1,2,3,4,6)→ (-,5,-,5,5)	nessuna		
4	(1,2,3,4,6)→ (6,-,6,-,6)	nessuna		
2	(1,3,5,6,7)→(2,4,7,-,7)	2 $\not\gamma$ 7→5,7 $\not\gamma$ 1 4 $\not\gamma$ 7→5,7 $\not\gamma$ 3	(5,6,7) (1,3,6)	
Passo2	(1,3,6) ⊂ (1,2,3,4,6)			(1,2,3,4,6) (5,6,7)
Passo 3				
3	(5,6,7)→(5,5,5)	nessuna		
4	(5,6,7)→(6,6,-)	nessuna		
3	(1,3,6→(-,-,5)	nessuna		
4	(1,3,6)→ (6,6,6)	nessuna		

Esempio 3: Algoritmo della suddivisione degli elementi della famiglia di presunta compatibilità –

- Famiglia degli insiemi compatibili massima

(1,2,3,4,6) (5,6,7)

- Verifica

i	Stato seguente	Incluso in	uscita
1	(1,2,3,4,6)→(1,1,3,1,3)	(1,2,3,4,6)	1
2	(1,2,3,4,6)→(2,2,4,4,-)	(1,2,3,4,6)	2
3	(1,2,3,4,6)→ (-,5,-,5,5)	(5,6,7)	4
4	(1,2,3,4,6)→ (6,-,6,-,6)	(5,6,7) e (1,2,3,4,6)	-
1	(5,6,7)→(-,3,3)	(1,2,3,4,6)	-
2	(5,6,7)→(7,-,7)	(5,6,7)	4
3	(5,6,7)→(5,5,5)	(5,6,7)	3
4	(5,6,7)→(6,6,-)	(5,6,7) e (1,2,3,4,6)	4

- Sarebbe stato possibile procedere anche in altro ordine

Esempio 3

- Esempio:

	i_1	i_2	i_3	i_4
q_1	q_1/u_1	$q_2/-$	-/-	$q_6/-$
q_2	$q_1/-$	q_2/u_2	$q_5/-$	-/-
q_3	q_3/u_1	$q_4/-$	-/-	$q_6/-$
q_4	$q_1/-$	q_4/u_2	$q_5/-$	-/-
q_5	-/-	$q_7/-$	q_5/u_3	$q_6/-$
q_6	$q_3/-$	-/-	$q_5/-$	q_6/ u_4
q_7	$q_3/-$	q_7/u_4	$q_5/-$	-/-

i	Stato seguente	Incluso in	uscita
1	(1,2,3,4,6)→(1,1,3,1,3)	(1,2,3,4,6)	1
2	(1,2,3,4,6)→(2,2,4,4,-)	(1,2,3,4,6)	2
3	(1,2,3,4,6)→ (-,5,-,5,5)	(5,6,7)	4
4	(1,2,3,4,6)→ 6,-,6,-,6)	(5,6,7) e (1,2,3,4,6)	-
1	(5,6,7)→(-,3,3)	(1,2,3,4,6)	-
2	(5,6,7)→(7,-,7)	(5,6,7)	4
3	(5,6,7)→(5,5,5)	(5,6,7)	3
4	(5,6,7)→(6,6,-)	(5,6,7) e (1,2,3,4,6)	4

$$q_6 \subseteq S_1 \quad q_6 \subseteq S_2$$

F = (S1, S2);

S1 = (q1, q2, q3, q4, q6); S2 = (q5, q6, q7)

Esempio 3

- Esempio (continua)
  - Indeterminazioni in corrispondenza di τ'(S1, i4) e τ'(S2, i4) ciascuna con 2 alternative e quindi 4 possibili soluzioni.

	i_1	i_2	i_3	i_4
S_1	S_1/u_1	S_1/u_2	$S_2/-$	S_1 o S_2 $/u_4$
S_2	$S_1/-$	S_2/u_4	$S_2/\ u_3$	S_1 o S_2 $/u_4$

  - La tecnica di perseguire insiemi disgiunti è impropria per macchine incomplete
    - In particolare può accadere che la macchina cui si perviene non mantenga la proprietà di chiusura