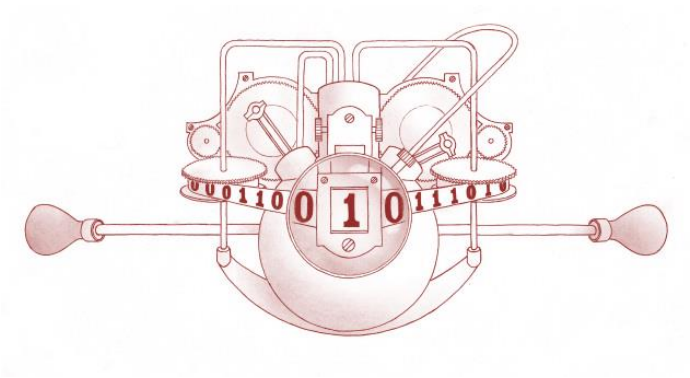
Automi a Stati Finiti



Materia: Architetture degli Elaboratori – 6 CFU

Facoltà di Scienze Matematiche Fisiche Naturali – Corso di Laurea in Informatica

Anno Accademico 2013/2014 - Università degli Studi di Palermo

Realizzato dalle lezioni della Professoressa

Simona Rombo

Scritto da

Valerio Bondì e Alessio Lombardo

01 Luglio 2014 – Revisione 3

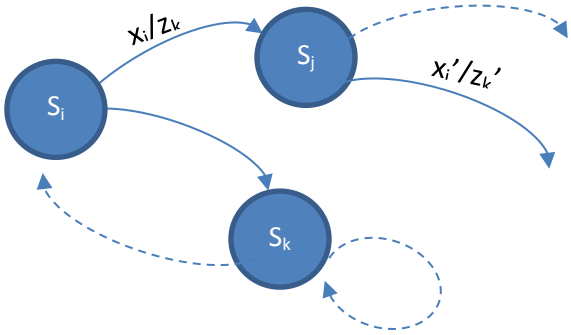
Introduzione

L'automa può essere definito come un sistema dinamico, ovvero quell'insieme di elementi interconnessi fra loro secondo delle regole, (dove il concetto di dinamico è dato dalle transizioni di stato che può subire il sistema a seconda degli input) che gode di due proprietà:

- È discreto: ovvero si considera il suo comportamento in un determinato tempo e stato;
- È invariante: il comportamento del sistema non dipende dall'istante di tempo in cui agisce.

L'automa, in generale, viene utilizzato per descrivere un linguaggio formale o per rappresentare un determinato circuito che compie un determinato lavoro.

Ci occuperemo di analizzare gli automi e di trovare una legge matematica che descriva tutti i possibili stati dell'automa per tutti i possibili input. Tratteremo gli automi a stati finiti, ovvero quegli automi che descrivono un evento tramite un numero finito di stati. Ecco la rappresentazione generica:



Legenda:

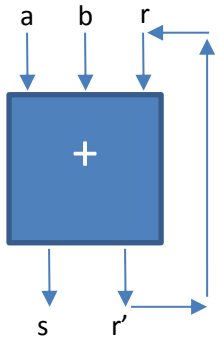
S_i : Nodo di rete di nome S_i; Descrive lo stato S_i

: Arco di rete, descrive la transizione da un nodo ad un altro

x_i: Generico input

z_k: generico output

Come esempio, riportiamo il caso di un circuito sommatore seriale (o sequenziale):



Poiché sequenziale, il circuito elabora una sola coppia di bit per volta. La macchina, deve essere in grado di memorizzare il riporto che eventualmente andrà sommato alla prossima coppia di bit.

Per esempio, dati i seguenti numeri binari, inseriti cifra per cifra a partire dalla meno significativa (cifra più a destra):

A: 10010 B: 01011

Si deve ottenere in output , sempre a partire dalla cifra meno significativa, il numero:

S: 11101

ATTENZIONE: Se i due numeri in input vengono inseriti a partire dalla cifra più significativa (quella più a sinistra) la somma sarà errata poiché la "propagazione" del riporto sarà invertita!

Analisi di un Automa – Gli Steps

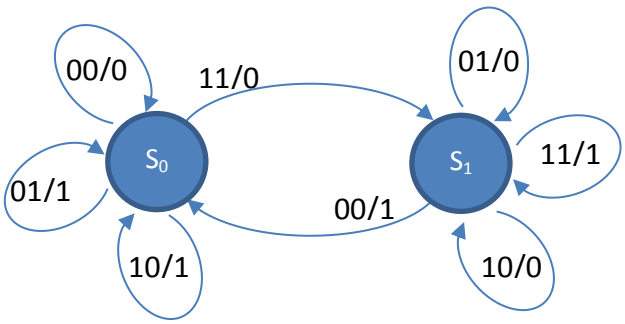
Vi sono 6 passaggi fondamentali per analizzare e trovare correttamente la legge che descrive l'automa:

1. Progettazione Grafica;
2. Tabella degli Stati;
3. Minimizzazione;
4. Tabella delle Transizioni;
5. Mappe di Karnaugh;
6. Schema Elettrico.

Attraverso l'esempio del Sommatore Seriale, si illustreranno tutti i passi necessari. Il passo relativo alla minimizzazione (che non è obbligatorio) verrà affrontato in seguito con un esempio più opportuno.

Step 1 – Progettazione Grafica:

Procediamo a disegnare l'automa. Siccome dobbiamo rappresentare gli stati di memoria (ovvero ciò che la macchina riesce a memorizzare) rappresentiamo l'automa secondo i riporti. Chiameremo S_0 lo stato senza riporto ed S_1 lo stato con riporto. Si avrà quindi un dispositivo di "Classe 2" (Riporto – Non Riporto):



Come si può ben notare, per alcune configurazioni, si ritorna allo stesso stato di partenza. Lo stato cambia solo se:

- ci troviamo in S_0 e diamo 11 in input;
- ci troviamo in S_1 e diamo 00 in input;

Step 2 – Tabella degli Stati:

Consiste semplicemente nel riportare per ogni stato la configurazione di input/output e lo stato successivo ad esso. Costruiremo una tabella nella quale:

- Le righe rappresentano i vari stati;
- Le colonne rappresentano i vari input dell'automa (la prima cifra è "a" mentre la seconda è "b");
- Le celle rappresentano l'output e lo stato successivo nel quale si deve transitare;

	00	01	10	11
S_0	0 / S_0	1 / S_0	1 / S_0	0 / S_1
S_1	1 / S_0	0 / S_1	0 / S_1	1 / S_1

Due esempi di lettura della tabella sono i seguenti:

Allo stato S_0, con input 00, si ottiene 0 e si ritorna allo stato S_0;

Allo stato S_1, con input 00, si ottiene 1 e si transita allo stato S_0.

Step 3 – Minimizzazione:

Passaggio non obbligatorio. Si veda il paragrafo successivo.

Step 4 – Tabella delle Transizioni:

Relativamente più semplice, basta modificare la tabella degli stati, assegnando un codice binario ai relativi stati.

Nell'esempio, dato che abbiamo solo S_0 ed S_1, basterà un bit per rappresentare i due stati (0 e 1). Se si hanno n stati, si considera l'esponente della potenza del 2 più vicina per eccesso per poter rappresentare ogni stato. Ad esempio, se si hanno gli stati S_0,S_1…S_{10}, saranno necessari 4 bit per rappresentarli tutti (poiché gli stati sono 11 e 2^4 è la potenza del 2 più vicina ad 11 per eccesso). Quindi, la tabella delle transizioni sarà:

	00	01	10	11
0	0 / 0	1 / 0	1 / 0	0 / 1
1	1 / 0	0 / 1	0 / 1	1 / 1

La trasformazione è avvenuta ponendo S_0=0 ed S_1=1.

Sia nella tabella degli stati sia in quella delle transizioni, è possibile inserire nelle celle prima lo stato successivo e poi l'output. Questo non deve però generare confusione tra progettista e realizzatore.

Step 5 – Mappe di Karnaugh:

Per costruire le mappe, gli identificativi di righe e colonne saranno gli input mentre il contenuto delle celle saranno gli "implicanti" (ovvero il contenuto) delle mappe. Saranno necessarie due mappe. La prima rappresenta l'output (cioè il bit di somma) mentre la seconda rappresenta la variabile "di anello" ovvero il bit necessario per la transizione da uno stato all'altro (ovvero il bit di riporto).

Output (bit di Somma):

r \ ab	00	01	11	10
0		1		1
1	1		1	

$s = a\bar{b}\bar{r} + \bar{a}\bar{b}r + abr + \bar{a}b\bar{r}$

Variabile di anello (bit di Riporto):

r \ ab	00	01	11	10
0			1	
1		1	1	1

$r' = ar + br + ab$

ATTENZIONE: Nella tabella di stato la numerazione degli input è crescente (secondo la classica numerazione BCD) mentre nelle mappe di Karnaugh bisogna rispettare la contiguità logica degli input (e quindi la numerazione procede secondo il codice Gray).

Si nota che le due mappe di Karnaugh sono identiche a quelle del Sommatore Parallelo (a logica combinatoria). Tuttavia bisogna sottolineare le differenze fra i due tipi di sommatori:

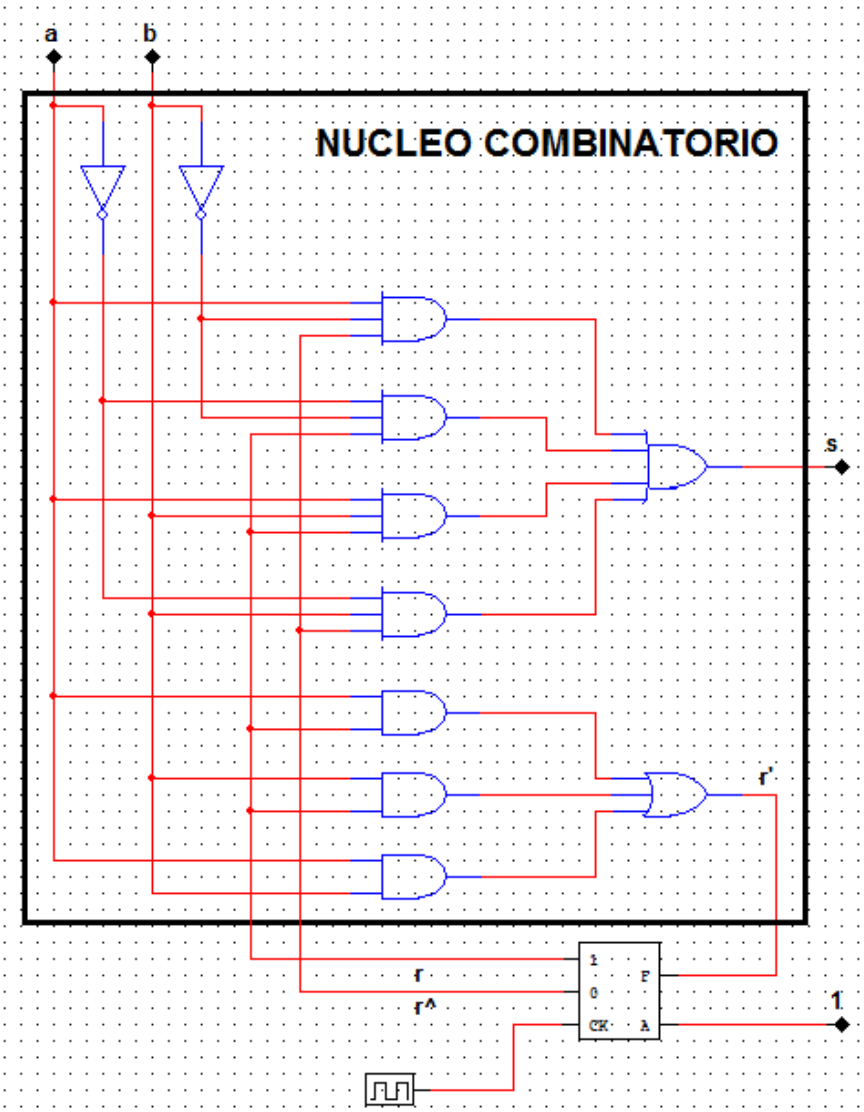
- Il sommatore seriale è composto da un solo "blocco" e può teoricamente sommare numeri ad infinite cifre. Il sommatore parallelo è invece composto da "n" blocchi e ovviamente non potrà sommare un numero infinito di cifre perché altrimenti sarebbero necessari infiniti blocchi.
- Il sommatore seriale è molto più lento perché può sommare una sola cifra per ciclo di clock. Il sommatore parallelo invece effettua la somma di tutte le cifre istantaneamente (trascurando i ritardi dovuti alla propagazione dei riporti).
- Il "nucleo" combinatorio del sommatore seriale è uguale al circuito del sommatore parallelo. Il sommatore seriale ha però in più un Flip-Flop che deve memorizzare il riporto.
- Il sommatore seriale necessita di un segnale di sincronismo (il clock) che deve essere fornito al Flip-Flop. Ognuna delle due cifre degli addendi deve essere fornita rispettando il clock altrimenti potrebbero verificarsi "sfasamenti" con conseguenti errori nella somma. Il sommatore parallelo non ha invece nessun clock (il clock può eventualmente essere inserito agli ingressi o all'uscita ma non all'interno del sommatore).

Step 6 – Schema Elettrico:

Basta leggere le mappe o la forma algebrica ottenuta per disegnare il circuito.

Si noti che la "variabile di anello" (in questo caso "r") viene convogliata in un Flip-Flop FA sincrono che si comporta da registro. Nei Flip-Flop FA il pin A è sempre connesso ad un segnale 1 poiché deve essere sempre abilitata la scrittura (si dice in gergo elettronico che il Flip-Flop deve agire in modalità "trasparente" ovvero deve permettere sia la scrittura che la lettura del dato mentre il clock è attivo, come se in quell'istante il registro non ci fosse).

Nello schema, "r^" rappresenta il resto negato. La variabile di anello quindi è una sola (diretta e negata) a differenza di quanto potrebbe sembrare.



Minimizzazione di un Automa

Questa parte è molto importante quando abbiamo a che fare con automi più complessi, ovvero quando la tabella degli stati è composta da un numero di stati troppo alto (tipicamente maggiore di 6, che è il massimo numero di variabili che è possibile trattare con le mappe di Karnaugh). Solitamente, se si riesce ad effettuare una progettazione grafica già ottimale, è superfluo procedere con questa fase, poiché risulterà che non è possibile minimizzare l'automa.

Per la minimizzazione, utilizzeremo l'**Algoritmo della tabella a scala**.

Parte 1 – Analisi della Tabella degli Stati:

Si abbia come esempio una tabella degli stati che deriva da una progettazione grafica non ottimizzata, dove quasi sicuramente saranno presenti stati equivalenti fra loro.

	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

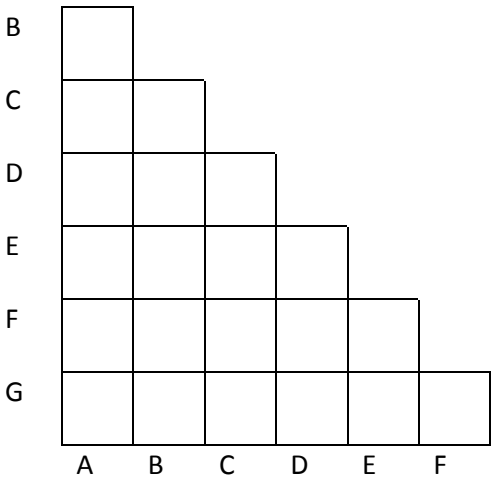
Dobbiamo appunto andare alla ricerca degli stati equivalenti ovvero quegli stati che godono delle seguenti condizioni:

1. A parità di un generico input, si ha un identico output;
2. A parità di un generico input, si ha un identico stato successivo;

Per capire meglio questi concetti, si svolgerà direttamente l'esercizio a partire dalla tabella soprastante.

Parte 2 – Costruzione della Tabella a Scala:

Costruiamo una tabella a scala che rappresenta tutte le possibili coppie di stati non uguali fra loro. Quindi si escludono totalmente le coppie riflessive ("A,A" , "B,B" , ecc.) e non si considerano le coppie simmetriche a quelle già considerate (si considera "A,B" e non "B,A" , "B,C" e non "C,B", ecc.). Si otterrà quindi una tabella a forma di scala nella quale nelle righe a sinistra si inseriscono ordinatamente tutti gli stati tranne il primo e nelle colonne in basso si inseriscono tutti gli stati tranne l'ultimo (non è possibile invertire i casi).



Questo algoritmo è di tipo ricorsivo, ovvero con una sola analisi cella per cella non si troveranno tutti gli stati equivalenti (alcune coppie di stati rimarranno non definite). Tuttavia, al secondo o terzo passo solitamente si risolvono tutti i casi incogniti.

Si procede cella per cella utilizzando le condizioni di equivalenza:

- Cella AB:
    - Uguali Output? SI.

- Uguali Stati successivi? Incognito, poiché le celle GC ed EA non sono ancora state analizzate. In questo caso si indicano all'interno della cella le 2 celle a cui riferirsi successivamente ovvero GC ed EA.

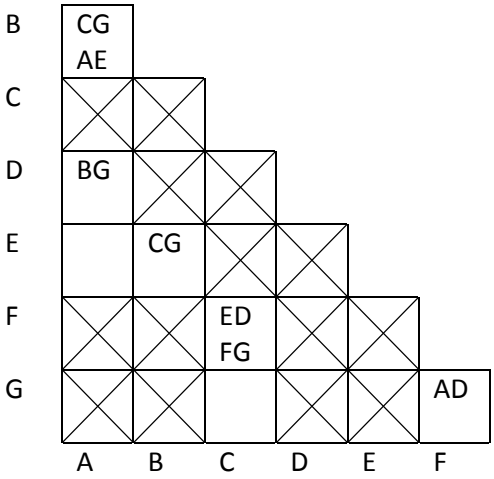
- Cella AC:
  - Uguali Output? NO. Stati sicuramente non equivalenti. La cella viene sbarrata.
- Cella AD:
  - Uguali Output? SI.
  - Uguali Stati successivi? Incognito. La cella EE, per la proprietà riflessiva, rappresenta stati equivalenti (infatti sono lo stesso stato). La cella GB è invece da analizzare. All'interno della cella si indica semplicemente quest'ultima condizione da verificare.
- Cella AE:
  - Uguali Output? SI.
  - Uguali Stati successivi? SI. La cella GG è riflessiva mentre la cella EA altro non è che la cella AE per simmetricità. Quindi, si avrà che gli stati A ed E sono equivalenti. La cella viene lasciata vuota.
- Celle AF, AG, BC:
  - Uguali Output? NO.

  Stati non equivalenti fra loro, celle sbarrate.

- Cella BD:
  - Uguali Output? SI.
  - Uguali Stati successivi? NO. La coppia EA rappresenta stati equivalenti ma la coppia BC rappresenta stati non equivalenti. Di conseguenza, gli stati B e D non sono equivalenti (prevale sempre la condizione di non equivalenza).

Si procede poi con le celle successive.

Ecco la tabella a scala risultante dopo il primo passaggio:



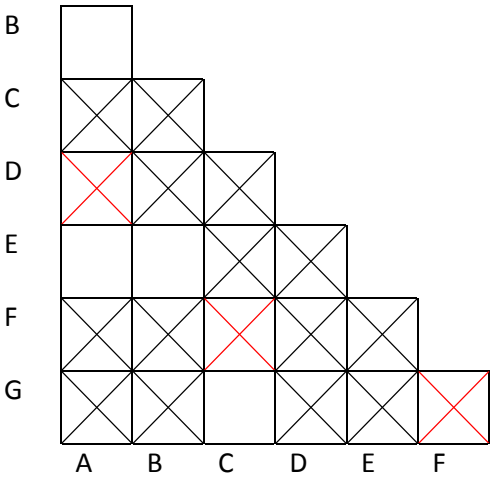
A questo punto si riapplica il metodo alla tabella già riempita per rimuovere le coppie di stati non definiti e vedere se sono equivalenti o meno. Le altre coppie si ignorano e si sbarrano o si lasciano vuote le celle corrispondenti nella nuova tabella:

- Cella AB:
  - CG Equivalenti? SI.
  - AE Equivalenti? SI.

  Stati Equivalenti. La cella nella nuova tabella sarà lasciata vuota.

- Cella AD:
  - BG Equivalenti? NO. Stati non equivalenti. Si sbarra la cella.
- Cella BE:
  - CG Equivalenti? SI. Stati equivalenti. La cella rimarrà vuota.
- Cella CF:
  - ED Equivalenti? NO
  - FG Equivalenti? Incognito.
    Poiché è già noto che gli stati E e D non sono equivalenti gli stati C ed F non sono equivalenti. La cella sarà sbarrata.
- Cella FG:
  - AD Equivalenti? NO. Stati non equivalenti. La cella si sbarra.

Ecco la tabella a scala risultante dopo il secondo passaggio (le cella sbarrate in rosso sono quelle considerate in quest'ultimo passaggio):



Il procedimento è concluso poiché non ci sono più stati non definiti.

Ricapitolando, questi sono tutti i possibili casi:

1. Stati non definiti: si scrive nella cella la coppia o le coppie da ricontrollare al passaggio successivo. Le coppie riflessive o simmetriche rispetto ai due stati considerati non vanno riportate;
2. Stati non equivalenti a causa di output diversi: si sbarra la cella;
3. Stati equivalenti per riflessività o simmetricità: si lascia la cella vuota (ATTENZIONE: tutte le coppie devono risultare riflessive o quantomeno simmetriche agli stati considerati, viceversa si agisce secondo il caso 1);

Questi ultimi 2 casi si verificano dal secondo passaggio in poi:

4. Stati equivalenti poiché collegati a coppie di stati TUTTI verificatisi equivalenti fra loro;
5. Stati non equivalenti poiché collegati ad ALMENO UNA coppia di stati verificatisi non equivalenti fra loro.

Parte 3 – Costruzione della nuova Tabella degli Stati:

Riguardando le celle vuote dell'ultima tabella, abbiamo trovato le seguenti equivalenze:

A≡B, A≡E, B≡E, C≡G.

Si può notare che le prime tre coppie sono equivalenti per la proprietà transitiva. Si può dunque dire che l'equivalenza fra stati diversi dal punto di vista matematico è una relazione d'equivalenza poiché valgono le proprietà riflessiva, simmetrica e transitiva.

L'insieme degli stati si può quindi partizionare in "classi di equivalenza". Ovvero, gli stati equivalenti fra loro vengono associati insieme in un'unica partizione (gli stati indipendenti avranno una propria partizione a sé):

A={ {A,B,E} , {C,G} , {D} , {F} }

Ad ognuna delle classi si associano dei nomi (in questo caso diversi dai nomi degli stati originari ma è possibile anche "riciclare" uno dei nomi già assegnati). Gli stati ottimizzati saranno quindi solamente 4:

α = A = B = E

β = C = G

γ = D

δ = F

Infine, riscriveremo la tabella degli stati, sostituendo agli stati non ottimizzati le nuove lettere assegnate ai nuovi stati ottimizzati:

	0	1
α	β / 0	α / 1
β	α / 0	β / 1
γ	α / 1	α / 0
δ	γ / 0	δ / 1

Nel caso in cui uno stato ottimizzato corrisponde a più stati originari (come nel caso di α e β), si prenderà come riferimento una sola delle righe della tabella originaria, a scelta; le righe degli altri stati possono essere ignorate.

Dopo la minimizzazione dell'automa e la creazione della nuova tabella degli stati, si può procedere dal punto 4 ovvero si crea la tabella delle transizioni, si sintetizza la rete combinatoria con le mappe di Karnaugh e infine si disegna il circuito.

Minimizzazione Rapida™ di un Automa

[Variante ideata da Alessio Lombardo. Tutti i diritti sono riservati.]

Nel precedente paragrafo, abbiamo visto come utilizzare l'algoritmo a scala per la minimizzazione di un automa. Come si è potuto notare, si tratta di un procedimento piuttosto lento, soprattutto quando ci sono molti stati che al primo passaggio risultano incogniti.

Il metodo qui proposto, sfrutta sempre la tabella a scala ma il controllo delle celle avviene in modo diverso.

Si abbia come esempio la stessa tabella degli stati considerata in precedenza:

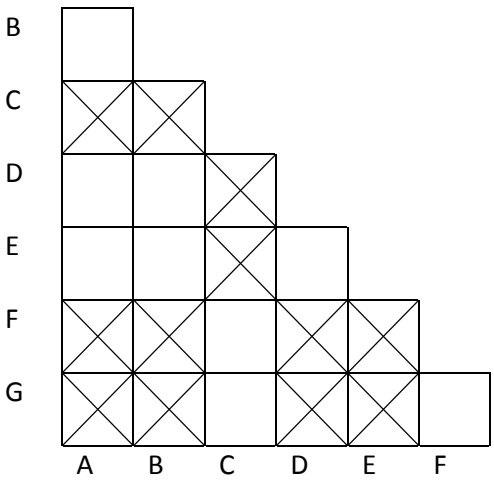
	0	1
A	G / 0	E / 1
B	C / 0	A / 1
C	E / 1	G / 0
D	B / 0	E / 1
E	G / 0	A / 1
F	D / 1	F / 0
G	A / 1	G / 0

Si considerano adesso esclusivamente gli output degli stati: A=01, B=01, C=10, D=01, E=01, F=10 e G=10.

Gli stati che hanno output diversi sono sicuramente non equivalenti. Quindi, guardando la tabella degli stati dall'alto verso il basso (o anche al contrario se si preferisce), si sbarrano le celle contrassegnate da stati con output diversi.

Per esempio, la cella AB non sarà sbarrata (gli output sono uguali) mentre la cella AC sarà sbarrata (output diversi).

Rapidamente, si arriverà alla situazione seguente:



A questo punto, si passa alla seconda fase. Sempre riferendosi alla tabella degli stati, si ignorano completamente gli output e si considerano solo gli stati successivi di ogni stato.

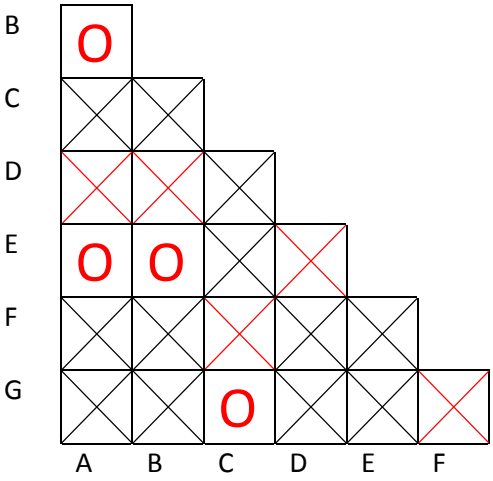
Ovviamente, si considerano solo le coppie rappresentate da celle non sbarrate:

- Cella AB: Stati GC ed EA incogniti. Al momento la cella si lascia vuota.
- Cella AD: Stati GC non equivalenti. Stati sicuramente non equivalenti e si sbarra la cella.
- Cella AE: Coppia GG riflessiva e coppia EA simmetrica. Stati equivalenti. La cella viene indicata con un cerchietto o con un altro simbolo (NON lasciarla vuota altrimenti potrebbe sembrare che la cella sia ancora incognita).
- Cella BD: Stati CB non equivalenti. Stati non equivalenti. Si sbarra la cella.
- Cella BE: Stati CG incogniti. La cella si lascia vuota al momento.
- Cella CF: Stati ED e GF incogniti. La cella si lascia vuota.
- Cella CG: Coppia GG riflessiva e stati EA equivalenti. Stati equivalenti. La cella viene cerchiata.
- Cella DE: Stati BG non equivalenti. Stati non equivalenti. La cella si sbarra.
- Cella FG: Stati AD non equivalenti. Stati non equivalenti. Si sbarra la cella.

A questo punto, si ricontrollano le celle ancora incognite:

- Cella AB: Stati GC ed EA equivalenti. La cella si cerchia poiché gli stati sono equivalenti.
- Cella BE: Stati GC equivalenti. Stati equivalenti. Si cerchia la cella.
- Cella CF: Stati ED e GF non equivalenti. Stati non equivalenti. La cella si sbarra.

La tabella a scala definitiva sarà dunque la seguente (le celle in rosso sono quelle contrassegnate in quest'ultimo passaggio):



Le celle cerchiate rappresentano stati equivalenti. Si può quindi procedere nello stesso modo indicato nella Parte 3 del paragrafo precedente.

Ecco i vantaggi del metodo rapido rispetto al metodo classico:

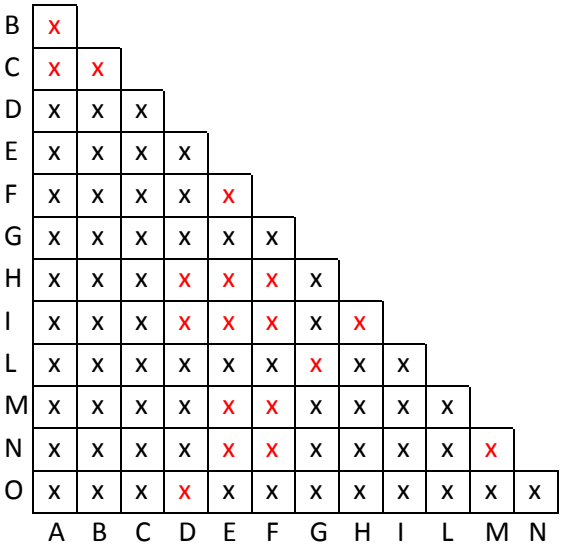
- Con il metodo classico, per ogni coppia di stati bisognava controllare contemporaneamente output e stati successivi, con il rischio di dimenticare uno dei due controlli. Con il metodo rapido si controllano prima tutti gli output e dopo tutti gli stati successivi diminuendo la probabilità di sbagliare.
- Con il metodo classico, le celle incognite dovevano essere "macchiate" indicando al loro interno le coppie da ricontrollare successivamente. Questo non solo è visivamente più confusionario ma è anche più lento e difficoltoso poiché bisogna fare 2 o più tabelle (o riscrivere sopra la precedente,

rendendo difficoltosa la correzione all'insegnante). Con il metodo rapido le celle incognite sono lasciate bianche, pronte per essere sbarrate o cerchiate nei passaggi successivi.

- Con il metodo classico, le celle della tabella dovevano essere abbastanza grandi poiché al loro interno era necessario annotare le celle da ricontrollare successivamente. Con il metodo rapido, le celle possono essere anche molto piccole poiché al loro interno va inserita una semplice "X" o un cerchietto.

Si riporta adesso un rapido esempio riguardante una tabella degli stati più complessa:

	0	1
A	B / 0	H / 0
B	B / 0	C / 0
C	D / 0	H / 0
D	E / 1	L / 0
E	E / 1	F / 1
F	G/ 1	M / 1
G	B / 0	O / 1
H	I / 0	H / 0
I	B / 0	L / 0
L	D / 0	M / 1
M	N / 1	M / 1
N	E / 1	M / 1
O	G / 1	H / 0



Le celle in nero sono quelle riguardanti stati che avevano output fra loro diversi e quindi sono state contrassegnate al primo passaggio.

Le celle contrassegnate in rosso sono quelle che al primo passaggio erano incognite (poiché avevano output uguali) e sono state contrassegnate nei passaggi successivi controllando gli stati successivi.

L'automa era già minimo poiché non esistono stati equivalenti.

Appunti

Questo foglio è lasciato libero per l'inserimento degli appunti del lettore.

Nota Bene: Questo documento riguardante argomenti di Architetture degli Elaboratori è totalmente artigianale e pertanto non si assicura che tutte le informazioni riportate siano corrette. Preghiamo il lettore di segnalare eventuali errori ed inesattezze all'indirizzo mail **alessiox.94@hotmail.it** . Grazie.

Alessio Lombardo