Macchine a Stati Finiti

Tutto ciò che c'è da sapere su FSA per l'esame di Reti Logiche

Prima di vedere gli esercizi, ecco un veloce recap sulle macchine a stati finiti:

Le **macchine a stati finiti (FSM - Finite State Machines)** sono **modelli matematici** utilizzati per descrivere il comportamento di un sistema che può trovarsi in un numero finito di stati e cambia stato in base a degli input.

- Il sistema si trova in **uno stato alla volta**.
- Un **evento (input)** può causare una **transizione** verso un altro stato.
- Il comportamento del sistema è definito da una **tabella di transizione** o un **diagramma a stati**.

Ecco i modelli di FSM che si usano:

Macchina di Moore

- L'output dipende **solo dallo stato corrente**.
- Più semplice da progettare.
- Esempio: Un distributore automatico che mostra il credito disponibile indipendentemente dal tasto premuto.

Macchina di Mealy

- L'output dipende sia dallo **stato corrente** che dall'**input attuale**.
- Più reattiva ai cambiamenti di input.
- Esempio: Un semaforo che passa subito al rosso se rileva un pedone che preme il pulsante.

Una **FSM (macchina a stati finiti)** è quindi un **modello astratto** usato per descrivere il comportamento di un sistema sequenziale. Quando implementiamo una FSM in **hardware**, questa diventa una **rete logica sequenziale**, cioè un circuito digitale che usa **flip-flop** per memorizzare lo stato e **porte logiche** per gestire transizioni e output.

Un circuito digitale può essere:

1. **Combinatorio** → L'output dipende solo dagli input attuali (es. sommatore, moltiplicatore).
2. **Sequenziale** → L'output dipende sia dagli input attuali che dallo **stato precedente** (es. contatore, registro a scorrimento, processore).Cioè un circuito che ha una memoria.

Le **FSM sono reti sequenziali**, perché devono **memorizzare lo stato corrente** per determinare il comportamento futuro. Questa memoria è realizzata con **flip-flop**, che memorizzano lo stato attuale e lo aggiornano a ogni ciclo di clock.

Quindi diciamo che questa è una nuova categoria di circuiti, perchè sin ora abbiamo visto solo circuiti combinatori: davo un input e subito potevo sapere quale era l'output.Adesso invece introduciamo gli elementi di memoria.

detta δ la **funzione di transizione**, cioè quella che ci dice qual'è lo stato prossimo dato input e stato corrente e λ la **funzione di uscita,** cioè quella che ci da l'uscita dato l'input e lo stato corrente, S l'insieme degli stati e I l'insieme degli input...

- **Mealy:** δ: S x I→ S* λ: S x I → O
- **Moore:** δ: S x I→ S* λ: S → O (out non dipende da input)
- **Contatori:** δ: S → S* (non c'è un input) λ: S = O (l'uscita è lo stato stesso)

(nb: quando scriviamo S*,Q* ecc... con * intendiamo uno stato prossimo, cioè lo stato successivo a quello corrente, negli esercizi sarà più chiaro)

Un tipo particolare di FSM che vediamo sono i **contatori**, cioè una macchina di Moore che non ha nessun input. Lo scopo di un contatore, come dice la parola stessa, è questo: dato uno stato di partenza, a ogni ciclo di clock si aggiorna e incrementa il conteggio, passando allo stato successivo, senza input quindi e l'uscita è proprio lo stato corrente. quando è arrivato all'ultima sequenza ricomincia da capo, ad esempio se un contatore conta fino a 4, i suoi stati sono

000,001,010,011,100. Quindi la sequenza è questa: parte da 000 → stato prossimo: 001→010→011→100→000(ricomincia da capo)→...

FLIP FLOP

Un flip flop è un elemento che memorizza un bit. Ce ne sono diversi tipi.Ogni tipo di flip flop si distingue dagli altri per:

- **Tabella delle Transizioni** = Tabella che mi dice Stato futuro in base all'attuale e agli input.
- **Tabella delle Eccitazioni** = Tabella che mi dice gli Input necessari per arrivare ad un determinato stato.Ogni flip flop ha certi input, ad esempio quello SR ne ha due, uno che serve a memorizzare uno(S, sta infatti per "set", quando vale 1 memorizza 1 nel flip flop) e uno per memorizzare zero(R, "reset", quando vale 1 riporta lo stato memorizzato dal FF a zero.R ed S non devono mai essere contemporaneamente 1 nei FF-SR, perchè non ha senso logico, poi vedremo cosa vuol dire)

ad esempio un FF-D(flip-flop tipo D) è semplice perchè ha un solo ingresso(chiamato D) e l'uscita, quindi lo stato memorizzato coincide proprio con il bit passato (D). Quindi Q*=D.

TABELLE DELLE TRANSIZIONI

S	R	Q*
0	0	Q
0	1	0
1	0	1
1	1	-

nei FF-SR quindi dare in ingresso S=R=1 non ha senso, non posso fare set e reset contemporaneamente

J	K	Q*
0	0	Q
0	1	0
1	0	1
1	1	Q'

FF-JS sono simili a FF-SR ma quando trovano S=R=1 invertono lo stato

D	Q*
0	0
1	1

FF-D sono tra i più semplici, ciò che diamo in ingresso memorizzano

T	Q*
0	Q
1	Q'

FF-T, "toggle", invertono lo stato memorizzato se trovano 1 nel loro unico ingresso T, altrimenti T=0 lascia lo stato memorizzato invariato

TABELLE DELLE ECCITAZIONI

Q	Q*	S	R
0	0	0	-
0	1	1	0
1	0	0	1
1	1	-	0

FF-SR

Q	Q*	J	K
0	0	0	-
0	1	1	-
1	0	-	1
1	1	-	0

FF-JS

Q	Q*	D
0	0	0
0	1	1
1	0	-
1	1	-

FF-D

Q	Q*	T
0	0	0
0	1	1
1	0	1
1	1	0

FF-T

Esercizio macchine completamente specificate

Data la tabella degli stati di una macchina compl. spec. con ingresso x e uscita y...

a. effettuare l'analisi di raggiungibilità
b. analisi di equivalenza
c. tabella delle transizioni usando una codifica degli stati a numero minimo di FF

d. tabella delle eccitazioni della macchina minima usando FF di tipo SR

tabella degli stati della macchina:

STATO	X=0	X=1
A(RESET)	G/0	G/0
B	D/0	C/1
C	E/0	F/1
D	D/0	D/1
E	G/0	H/0
F	D/0	D/1
G	E/0	F/0
H	D/0	D/1

la prima richiesta è l'analisi di raggiungibilità: dobbiamo capire quali stati si possono raggiungere dallo stato iniziale(RESET o RST) e eliminare quelli che non sono raggiungibili, cosi che la macchina diventi minima, rappresentandola infatti con un grafo(come si faceva ad API), alla fine della nostra analisi e della minimizzazione(che sarà lo step successivo) vedremo un grafo più piccolo.

Per fare l'analisi di raggiungibilità partiamo dallo stato iniziale, cioè A e scriviamo subito i nodi in cui si entra da A. Nell'iterazione successiva dobbiamo entrare in questi nodi a cui si arriva da A appunto. Qui si va solo in G quindi entro soltanto in G. A questo punto però, in G devo entrare anche in E ed F, quindi procedo in parallelo e vedo quali nodi si raggiungono da E ed F

$$\{A, G\} => \{A, G, E, F\}$$

A questo punto quindi entro in E e vedo che da li posso andare o in G o in H. G già l'ho visitato, quindi non ci rientro, H no, quindi lo aggiungo e nello step(iterazione) successivo ci entro. Poi entro in F e vedo che da F posso andare soltanto in D, che ancora non visito, per cui lo aggiungo e ci entro nello step successivo insieme ad H.

$$\{A, G\} => \{A, G, E, F\} => \{A, G, E, F, H, D\}$$

Quindi mi manca solo entrare in D e vedere dove si può andare da li: in D si rimane in D, quindi non si aggiunge niente. **Ci si ferma quando l'insieme non cresce più e in una iterazione rimane invariato.**

$$\{A, G\} => \{A, G, E, F\} => \{A, G, E, F, H, D\} => \{A, G, E, F, H, D\}$$

Quindi adesso ho scoperto quali nodi non sono raggiungibili: quelli iniziali erano {ABCDEFGH} e quelli dell'analisi di raggiungibilità sono soltanto {ADEFGH}, in questo ultimo insieme mancano B e C, quindi li cancello perchè sono irrangiungibili.

Per comodità riporto qui la **tabella degli stati** senza stati irragiungibili:

STATO	X=0	X=1
A(RESET)	G/0	G/0
D	D/0	D/1
E	G/0	H/0
F	D/0	D/1
G	E/0	F/0
H	D/0	D/1

ricordiamoci che in ogni cella c'è STATO_PROSSIMO/USCITA. Quindi il significato di una riga è questo: prendiamo ad esempio la riga E, ci dice che se siamo nello stato E e si riceve in input 0, il prossimo stato sarà G e l'uscita 0, se invece si riceve in input 1, si va nello stato H e l'output è 0.

analisi di equivalenza: adesso devo iniziare a vedere quali stati sono equivalenti, cosi da creare le cosiddette **classi di equivalenza (cioè gli insiemi degli stati che sono equivalenti, ma poi vedremo meglio come prenderli)** e ridurre quindi il numero degli stati.

> ⚠️ due stati sono equivalenti solo se hanno le stesse uscite. Se gli stati che raggiungono sono diversi, sono potenzialmente equivalenti.

si costruisce a tal scopo una tabella mettendo in verticale tutti gli stati(tra quelli raggiungibili ovviamente) tranne il primo e in orizzontale tutti gli stati tranne l'ultimo:

D	X	————	————	————	————
E	GH	X	————	————	————
F	X	~	X	————	————
G	GE GF	X	HF	X	————
H	X	~	X	~	X
	A	**D**	**E**	**F**	**G**

La struttura deve essere triangolare, infatti dove ci sono le linee potete non disegnare direttamente le celle.

Come si compila questa tabella ? partiamo dalla cella in alto a sinistra, quindi la cella AD, e vado a vedere la tabella degli stati ridotta e controllo la riga A: per X = 0, da A andiamo in G e l'uscita è 0, per X = 1 idem, poi confronto con la riga D e vedo che le uscite sono diverse, quindi A e D NON sono equivalenti. Segno X.

Poi scendo sotto e confronto A con E e vedo che effettivamente le uscite sono uguali, ma per X=1 gli stati sono diversi, quindi i due stati A ed E sono equivalenti solo se lo sono anche GH, quindi scrivo GH.

Quando invece trovo due stati che hanno stessi stati prossimi e stesse uscite, ho trovato stati equivalenti e segno quindi una tilde.

nb: quando confronto GE ,vedo che per X=0 avrei GE, ma è lo stesso confronto, quindi non lo includo.

A tabella finita adesso dobbiamo propagare le non equivalenze e le equivalenze in modo da togliere le celle che contengono ancora stati da confrontare. Inizio dalla cella AE: devo vedere se GH sono due stati equivalenti, allora vado alla cella GH e vedo che c'è una X, quindi propago la non equivalenza:

D	X	————	————	————	————
E	X	X	————	————	————
F	X	~	X	————	————
G	GE GF	X	GE	X	————
H	X	~	X	~	X
	A	**D**	**E**	**F**	**G**

Poi continuo in AG: devo controllare GF e GE, partiamo da GE. Andiamo allora nella cella GE e vediamo che c'è una X, allora la propaghiamo anche su GA:

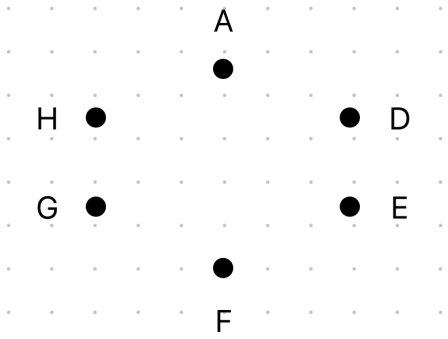
D	X	————	————	————	————
E	X	X	————	————	————
F	X	~	X	————	————
G	X	X	HF	X	————
H	X	~	X	~	X
	A	**D**	**E**	**F**	**G**

infine manca GE, che dipende da HF. Allora vado sulla cella HF e vedo che c'è una equivalenza. Allora la propago:

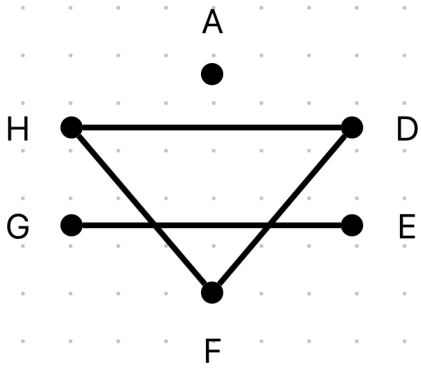
D	X	———	———	———	———
E	X	X	———	———	———
F	X	~	X	———	———
G	X	X	~	X	———
H	X	~	X	~	X
	A	**D**	**E**	**F**	**G**

fine!

Adesso dobbiamo vedere quali sono le classi di equivalenza: dobbiamo fare un grafo in questo modo: prendiamo tutti gli stati raggiungibili e questi saranno i punti(nodi) e disponiamo questi nodi in modo da fare un poligono:



Adesso dobbiamo collegare tutti quelli che sono equivalenti, quindi dove c'è la tilde: GF,HF,HD,FD:



Ora posso finalmente capire quali sono le classi di equivalenza e scrivere la tabella degli stati semplificata.

Le classi di equivalenza sono gli stati che non sono collegati a nulla, quindi A. Chiamo questa classe che contiene solo A, α ={A}. Poi le classi di equivalenza sono tutti i collegamenti, quindi β = {FDH}, ɣ ={GE}.

A questo punto posso scrivere la tabella degli stati semplificata. Per farlo mi basta iniziare da una delle classi di equivalenza, ad esempio β, prendo uno stato a caso al suo interno, ad esempio F. Vado nella tabella iniziale e vedo, dando in input X=0 e X=1 in che stato vado a finire e vedo a quale classe di equivalenza appartiene, ad esempio se finisco in G, scriverò ɣ.

tabella degli stati ridotta:

	X=0	X=1
α	ɣ / 0	ɣ / 0
β	β/0	β/1
ɣ	ɣ / 0	β/0

quindi adesso posso passare a costruire la tabella delle transizioni e poi posso inserire i flip flop. Per farlo però devo codificare gli stati, cioè invece di usare α, β, γ. Per farlo bastano 2 bit: 00 per il primo, 01 per il secondo, 10 per il terzo. 2 bit di codifica per lo stato vuol dire due flip flop. Il primo bit lo chiamo Q0 e il secondo Q1 e sono i bit salvati nel FF.

	X=0	X=1
00	10/ 0	10 / 0
01	01/0	01/1
10	10 / 0	01/0
11	-	-

ovviamente lo stato 11 è inutilizzato. Mettiamo quindi gli stati prossimi nella tabella delle transizioni:

		X=0			**X=1**		
Q0	***Q1***	***Q0****	***Q1****	***Y***	***Q0****	***Q1****	***Y***
0	0	1	0	0	1	0	0
0	1	0	1	0	0	1	1
1	0	1	0	0	0	1	0

adesso devo mettere i FF. Per farlo mi basta vedere stato corrente e stato prossimo della stessa riga e mi chiedo quale input devo dare al FF scelto(qui scegliamo gli SR), ad esempio prendiamo la prima riga sull'input x=0, che passa da Q0 = 0 Q1=0 a Q0* = 1 Q1*=0.Per fare questa transizione il primo SR ha bisogno di S0= 0 R0 = -, il secondo S1 = 1 R1=0.

Per brevità analizziamo solo il caso x= 0. **tabella delle eccitazioni:**

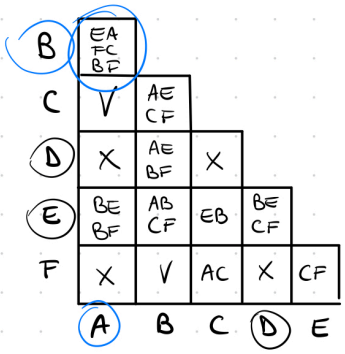
		X=0					X=1
Q0	***Q1***	***S0***	***R0***	**S1**	***R0***	**Y**	…
0	0	0	-	1	0	0	…
0	1	0	-	-	0	0	…
1	0	—	0	0	-	0	..

Macchine non completamente specificate

Qui cambia leggermente il procedimento perchè quando facciamo l'analisi di equivalenza, non possiamo dire che due stati sono equivalenti, ma potenzialmente compatibili(si dice compatibilità nelle n.c.s), quindi invece della tilde si usa una spunta. Inoltre non si propagano le equivalenze, ma soltanto le non compatibilità. Quando confronto le righe della tabella degli stati e trovo ad esempio un don't care e uno stato A, è come se su entrambe le righe ci fosse A, idem per le uscite, se su una riga c'è don't care e sull'altra 0 o 1 è come se su entrambe ci fosse 0 o 1. Vediamo un esercizio:

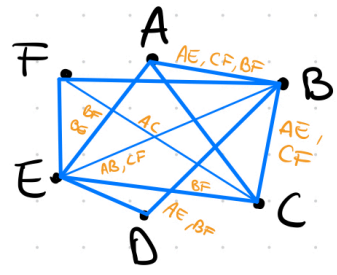
	X=00	X=01	X=11	X=10
A	E/0	F/1	A/0	B/-
B	A/-	C/-	-/-	F/1
C	E/-	F/1	A/-	-/-
D(RST)	E/1	C/0	D/-	B/1
E	B/-	F/-	-/-	F/-
F	-/0	C/-	C/1	F/-

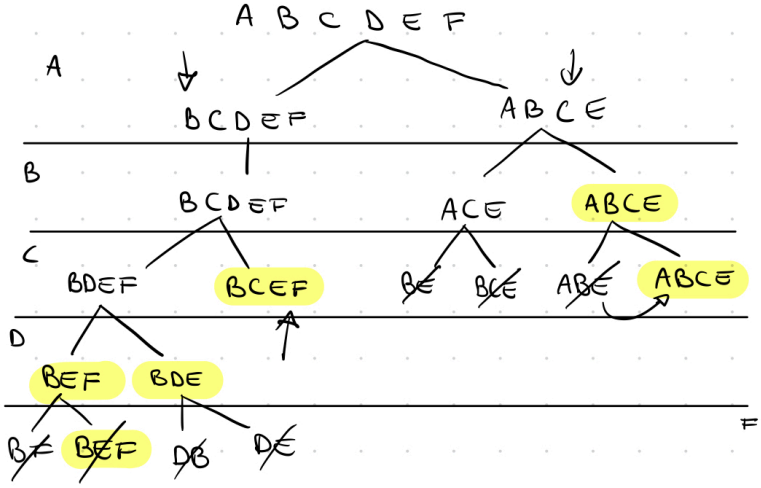
analisi di compatibilità



quelli che non hanno nè V nè X sono POTENZIALMENTE COMPATIBILI

qui non c'è nessuna non compatibilità da propagare. Si lascia cosi la tabella, perchè adesso bisogna fare un albero che ci permetterà di capire quali sono le classi di massima compatibilità(come le classi di equivalenza).Supponendo di aver già fatto l'analisi di raggiungibilità, che rimane uguale a prima, partiamo e iniziamo con il primo nodo, che conterrà tutti gli stati raggiungibili:





$\alpha = \{ABCE\} = \{AEB, FC, BF\}$

$\beta = \{BCEF\} = \{ABE, FC, BF\}$

$\gamma = \{BDE\} = \{ABE, FC, BF\}$

ora, livello per livello esploriamo ogni nodo, partendo dal primo. la regola è questa:

> Partendo dal primo livello si esplora A, poi il secondo, il terzo stato ecc... Si fanno due sottoalberi: uno sinistro e uno destro. In quello sinistro riscriviamo semplicemente quello che c'è a 1 livello sopra ma togliamo lo stato che stiamo correntemente esplorando, semplicemente questo. Poi invece nel sottoalbero destro dobbiamo entrare nel nodo che stiamo esplorando(l'ho riportato in alto a sinistra, ma tanto ogni livello che scendiamo è il nodo successivo che si sta analizzando) e vediamo quali sono compatibili dalla tabella fatta prima e consideriamo compatibili anche quelli che lo sono potenzialmente(quelli che nelle celle hanno ancora le lettere, per capirci).Riscriviamo quindi lo stato corrente e tutti quelli compatibili e potenzialmente compatibili, ma con una regola: ogni stato che riscrivo sotto deve essere presente nel livello di sopra, altrimenti lo ignoro. Mi fermo quando trovo la stessa sequenza 2 volte.

qui a volte abbiamo riscritto soltanto un sotto albero anzichè 2, ma è soltanto una semplificazione perchè quando vedo che ottengo due sequenze in cui una è contenuta nell'altra, posso eliminarla già da li.

alla fine elimino tutte quelle che sono contenute nelle altre.

Da qui è tutto uguale alle completamente specificate.