Modellare con le Finite State Machines

Angelo Gargantini

Testing e verifica del software

2026

Notazione della specifica

Si possono usare diverse notazioni per specificare il sistema
Esempi

- UML: macchini di stati/diagrammi di interazione
- ASM: ...
- Simulink: ...
- VHDL
- Itemis Create (vedremo dop)

Qui vediamo le FSM (di Mealy/Moore)

- **Macchine a stati finiti (FSM)**

Macchine a stati finiti con output

Una FSM (S, I, O, δ, λ) con output è:

- S : insieme finito di stati
- I : insieme finito di eventi di input
- O : insieme finito di eventi di output
- δ : funzione di transizione
- λ : funzione di output
- una macchina di Mealy se è una FSM che produce un output per **ciascuna transizione**
- una macchina di Moore se è una FSM che produce un output per **ciascun stato**

Macchina di Mealy

Una macchina di Mealy è una tupla

(S, I, O, δ, λ)

- S : insieme finito di stati
- I : insieme finito di eventi di input
- O : insieme finito di eventi di output
- δ : S x I ->S : funzione di transizione
- λ : S x I ->O : funzione di output

Spesso è anche fissato lo stato iniziale $s_0 \in S$

semantica:

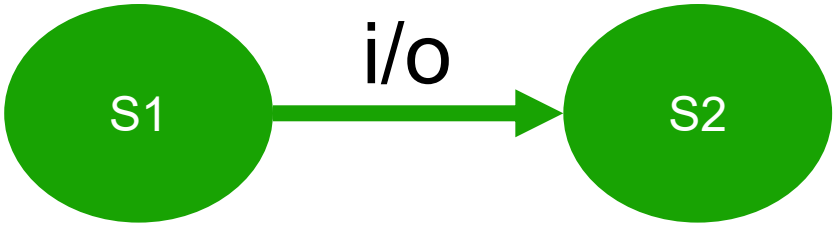
$\delta(s,i)$ = s' vuol dire che se mi trovo in s e ricevo i, vado in s'

$\lambda(s,i)$ = o vuol dire che se mi trovo in s e ricevo i, produco o

FSM di Mealy: rappresentazione grafica

Nei diagrammi che rappresentano una FSM di Mealy, ogni arco è etichettato da i/o

- **i** denota un simbolo di input ed è anche noto come evento di input
- **o** denota un simbolo di output ed è anche noto come evento di output



$\delta(s1,i) = s2$

$\lambda(s1,i) = o$

FSM di Mealy: rappresentazione grafica

Esempio:

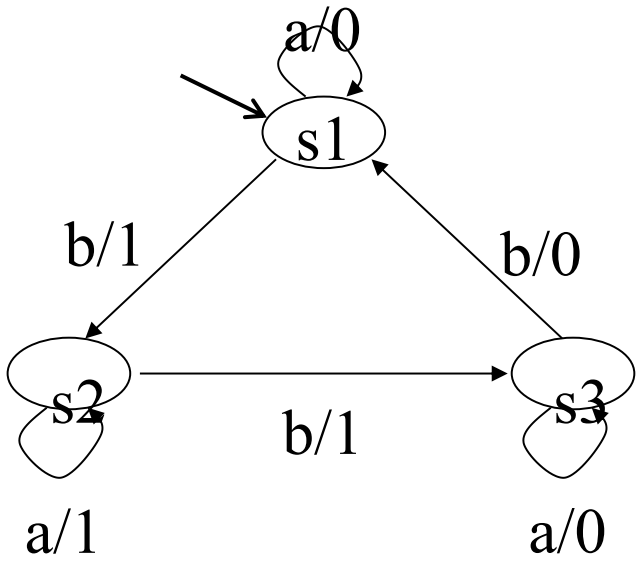
S = {s1, s2, s3}

I = {a, b}

O = {0, 1}

δ = (s1,a) = s1, (s1,b)=s2,
(s2,a) = s2, (s2,b)=s3,
(s3,a) = s3, (s3,b)=s1

λ = (s1,a) = 0, (s1,b) = 1,
(s2,a) = 1, (s2,b) = 1,
(s3,a) = 0, (s3,b) = 0}

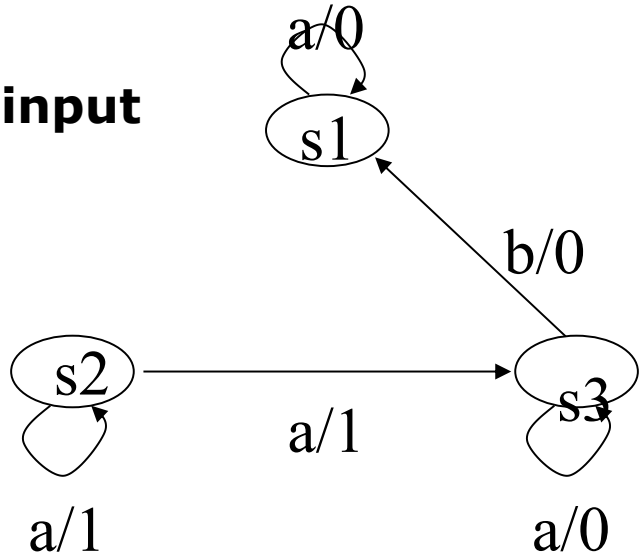


Errori nella definizione dalla mia macchina

1. non definisco cosa fare quando ho un certo input

- Arriva input b in S1

2. ho diverse transizioni con lo stesso input

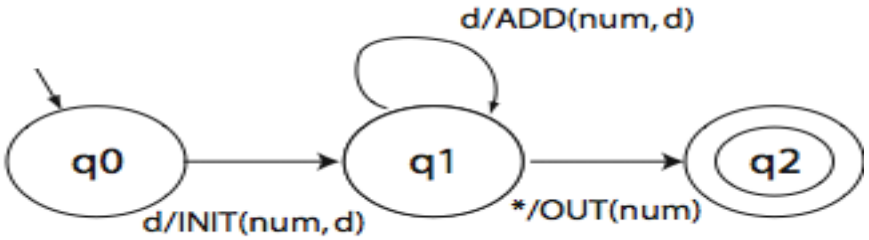


FSM: azioni sulle transizioni

Un evento di output può anche essere una azione della macchina

Esempio: macchina per convertire una sequenza di cifre decimali (d) in un intero (num)

- Azioni:
  - INIT: inizializza "num"
  - ADD: aggiunge la cifra "d" al valore corrente di "num"
  - OUT: output del numero



FSM di Mealy

È utile vedere una FSM di Mealy come la tupla (S, I, O, T)

- S : insieme finito di stati
- I : insieme finito di eventi di input
- O : insieme finito di eventi di output
- T : insieme finito di transizioni
  - Una transizione è una tupla (s, i, o, s’)
    - s : stato sorgente
    - i : evento di input
    - o : evento di output
    - s’ : stato target

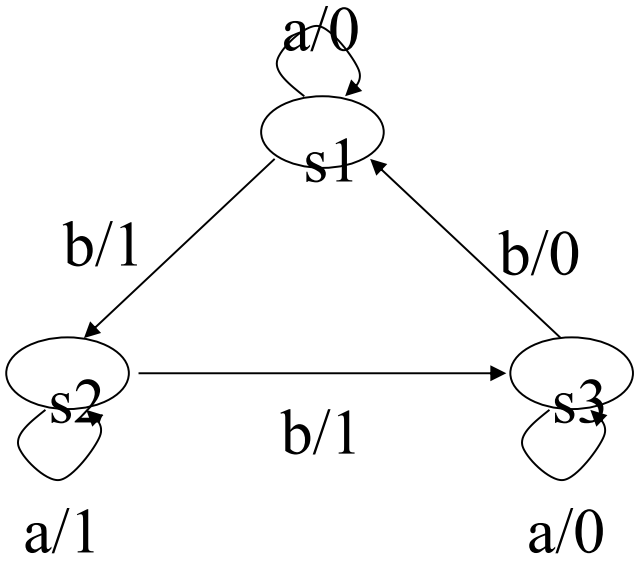
FSM di Mealy: esempio

S = {s1, s2, s3}

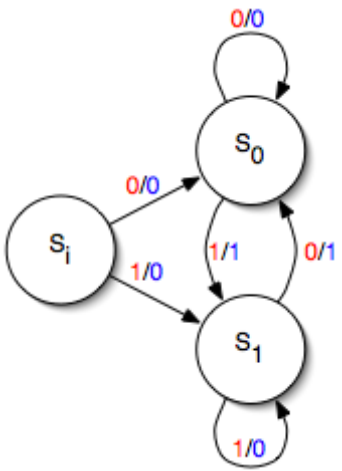
I = {a, b}

O = {0, 1}

T = {(s1,a,0,s1),
(s1,b,1,s2),
(s2,a,1,s2),
(s2,b,1,s3),
(s3,a,0,s3),
(s3,b,0,s1)}



Esempio di macchina di Mealy



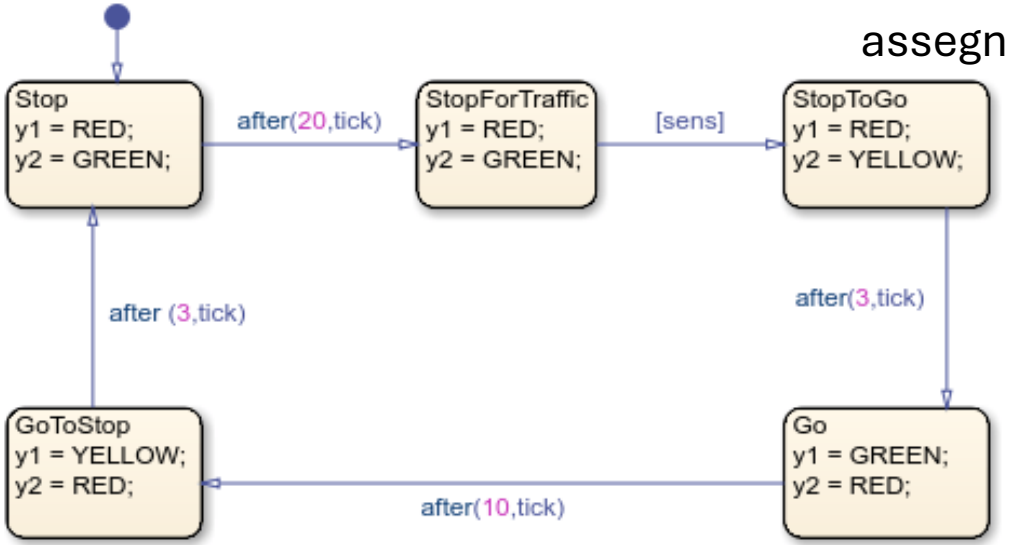
- M ha un ingresso (che può essere o 0 o 1) un'uscita (che può essere o 0 o 1).
- Ogni transizione è etichettata con il valore dell'input (mostrato in rosso) e il valore dell'output (mostrato in blu).
- La macchina si avvia nello stato *Si*.,
- l'output è l'OR ESCLUSIVO (XOR) dei due valori di input più recenti; quindi, la macchina implementa un rilevatore di "rampa", emettendo un 1 ogni volta che l'input si capovolge e uno 0 in caso contrario.
- Ad esempio, se in Si riceve uno 0, va in S_0 e produce 0.
- Prova a scrivere qualche esecuzione

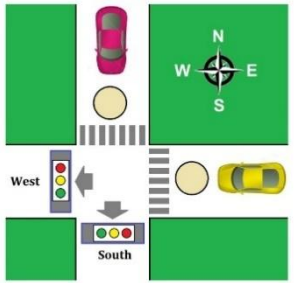
Macchine di Moore

- **Definizione:** l'output dipende **solo dallo stato corrente**.
- Ogni stato ha associato un valore di output fisso.
- **Transizioni:** determinate solo dall'input, ma l'output cambia solo quando cambia lo stato.
  - S : insieme finito di stati
  - I : insieme finito di eventi di input
  - O : insieme finito di eventi di output
  - δ : S x I ->S : funzione di transizione
  - λ : S ->O : funzione di output

Esempio macchine di MOORE Light controller

Gli **output** della macchina sono y1 e y2 per le luci rispettivamente Nord-Sud e Est-Ovest (l'output è espresso con un assegnamento).





Traffic Lights Intersection

Il Light_Controller si comporta come una macchina di Moore perché aggiorna i suoi output in base allo stato corrente prima di passare a un nuovo stato.

Esempio - spiegazione

- L'operatore **after** implementa un timer per il conto alla rovescia, che viene inizializzato quando si entra nello stato. Per impostazione predefinita, il timer fornisce un semaforo verde più lungo in direzione est-ovest rispetto alla direzione nord-sud perché il volume del traffico è maggiore sulla strada est-ovest. La luce verde in direzione Est-Ovest rimane accesa per almeno 20 ticchettii, ma può rimanere verde finché non arriva traffico in direzione Nord-Sud. Un sensore rileva se le auto sono in attesa al semaforo rosso in direzione nord-sud. In tal caso, il semaforo diventa verde in direzione nord-sud per mantenere il traffico in movimento. Quindi gli input sono il timer rappresentato dall'operator **after** e il sensore rappresentato da **[sens]**.

Traffic Light 2

- A esempio nello stato Stop, il semaforo è rosso per Nord-Sud, verde per Est-Ovest. Dopo 20 tick di clock, lo stato attivo diventa StopForTraffic in cui controlla il sensore – ma la luci non cambiano. Se il sensore indica che le auto sono in attesa ([sens] è true) in direzione nord-sud, lo stato attivo diventa StopToGo e la luce y2 diventa YELLOW.

Limiti delle FSM

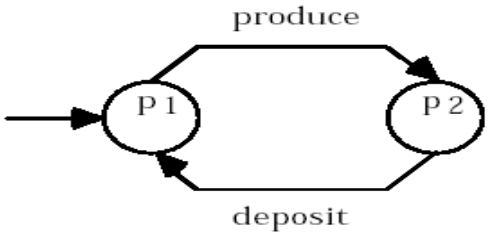
E’ possibile rappresentare solo un numero finito di stati

Esplosione del numero di stati:

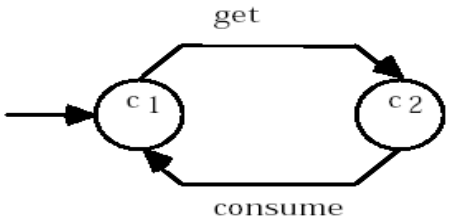
- dato un numero di FSM con k1, k2, … kn stati ciascuna, la loro composizione è una FSM con k1 * k2 *… * kn stati
- tale crescita è esponenziale nel numero di FSM
- ci piacerebbe una crescita lineare, ossia k1 + k2 +… + kn stati

Esplosione degli stati: esempio

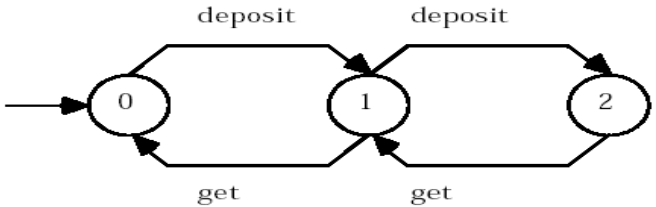
Produttore



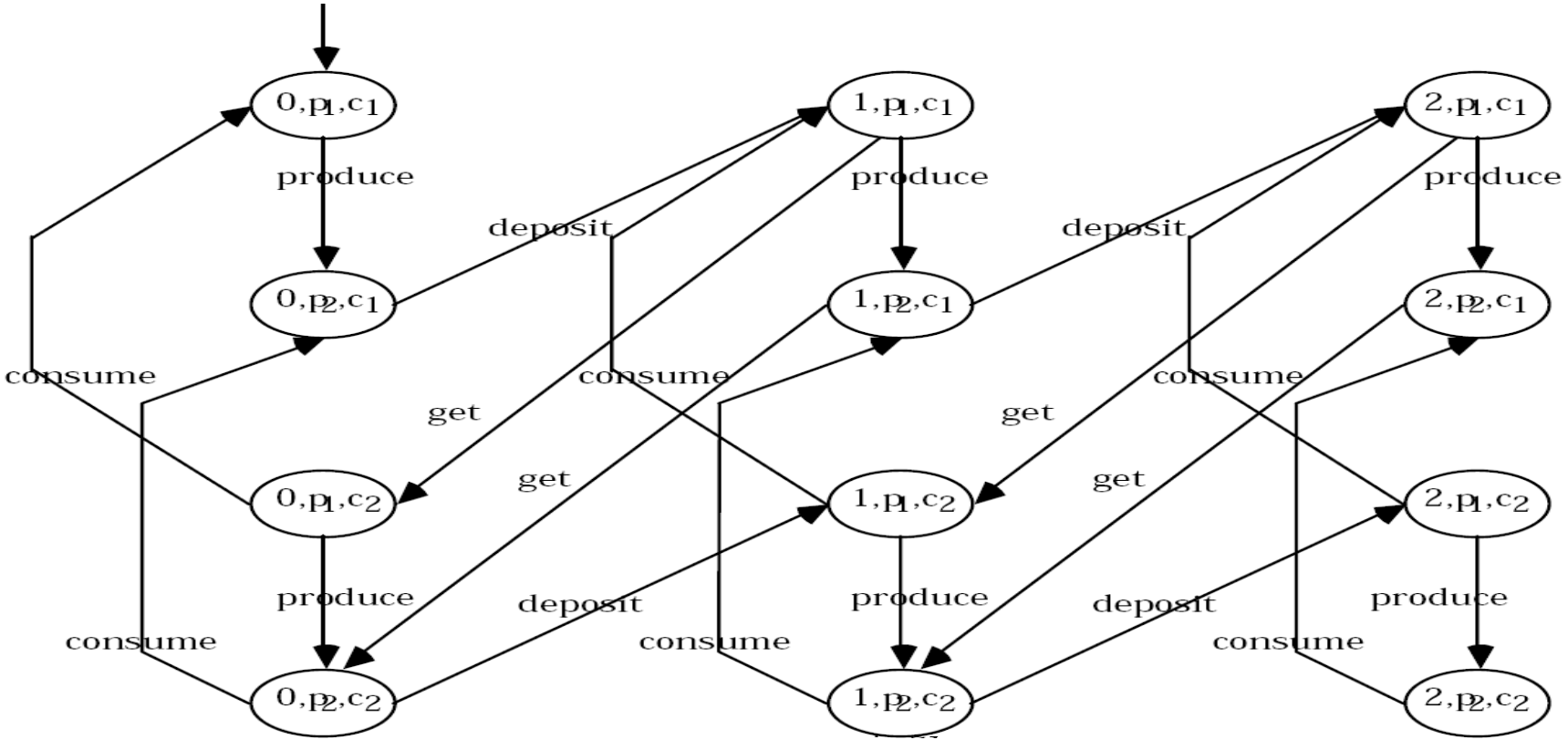
Consumatore



Magazzino



Combinando le FSM



Limiti delle FSM

Per superare i limiti di composizionalità delle FSM, sono state definite opportune estensioni (tra cui le Statecharts di UML) che sono dotate dei concetti di sottomacchina e permettono

- Composizione sequenziale
- Composizione parallela
- Modularità

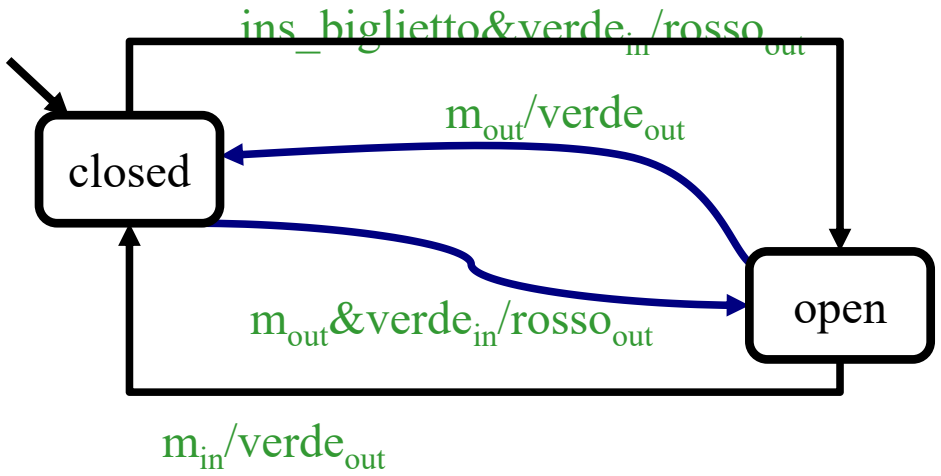
Soluzione esercizio

$verde_{in}$, $verde_{out}$, $rosso_{in}$, $rosso_{out}$: colori semaforo

m_{in} : segnale di presenza dell'auto nel parcheggio

m_{out}: segnale di presenza dell'auto fuori il parcheggio

ins_biglietto: segnale di inserimento biglietto



Esercizio

Modellare con una FSM il comportamento di una sbarra che consente l'accesso/l'uscita di un parcheggio.

Per accedere al parcheggio, il semaforo deve essere verde. Mentre la sbarra è aperta, il semaforo è rosso.

Quando l'auto è entrata, la sbarra si chiude ed il semaforo ritorna verde.

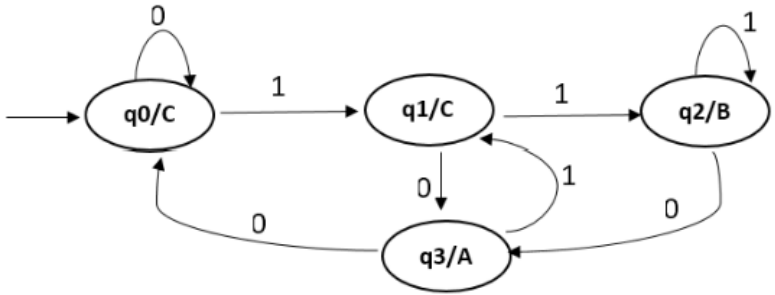
Per uscire dal parcheggio, l'autista deve inserire il biglietto pre-pagato nel dispositivo che comanda la sbarra.

Quando l'auto è uscita la sbarra si chiude.

Esercizio 1

- Costruisci una macchina di Moore che prende l'insieme di tutte le stringhe con alfabeto {0, 1} come input e produce 'A' come output se l'input ha come ultime due cifre 10 o produce 'B' come output se l'input ha come ultime due cifre 11 altrimenti produce 'C' .
- Input alphabet: I = {0, 1}
- Output alphabet: O = {A, B, C}
- Esempio:
- input: 1 0 1 0 → produce C A C A
- input: 1 0 1 1 → produce C A C B

Soluzione 1



Esercizio 2

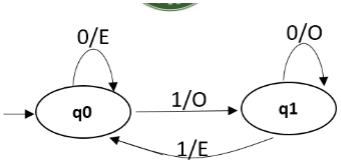
- Progetta una macchina di mealy che dia come output "a" ogni volta che viene rilevata la sequenza "01" in qualsiasi stringa binaria di input. Altrimenti darà “b” come output
- Esempio:
- input: 1 0 1 0 1 → produce b b a b a
- input: 0 1 1 0 0 → produce b a b b b

Esercizio 1 e 2

- **Problema 1 – Macchina di Moore per bit stuffing**
- Costruisci una macchina di Moore che prende un numero binario (una cifra alla volta) e sostituisce il primo 1 con uno 0 da ogni sottostringa inizia con 1. Ad esempio, 0001001110 diventa 0000000110. Questo tipo di Il "bit stuffing" può essere utilizzato nella trasmissione di dati e nelle telecomunicazioni per la codifica di lunghezza di esecuzione per limitare il numero di bit consecutivi dello stesso valore. Viene inserito un bit del valore opposto dopo il numero massimo consentito di bit consecutivi.
- **Problema 2 – Macchina di Mealy per bit stuffing**
- Proviamo a scrivere una macchina di Mealy che faccia la stessa cosa

- **Problema 3 – Macchina di Mealy per conta 1 PARI o DISPARI**
- Costruisci una macchina di Mealy che prende l'insieme di tutte le stringhe con alfabeto {0, 1} come input e produce 'P' (pari) se ha preso in input fino ad ora un numero pari di 1, oppure produce ‘D’ se ha letto un numero dispari di 1.
- **Problema 4 – Macchina di Moore per conta 1 PARI o DISPARI**
- Proviamo a scrivere una macchina di Moore che faccia la stessa cosa

Soluzione



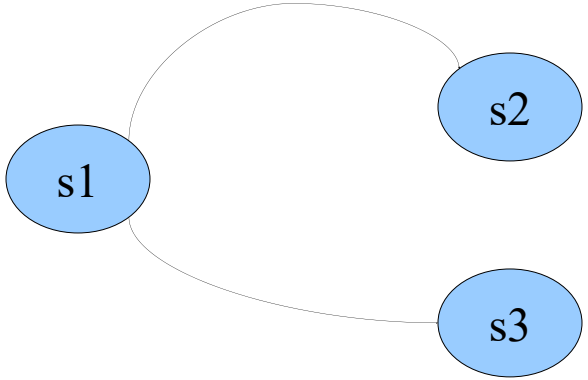
In sintesi

- Abbiamo visto:
  - le Macchine a Stati finiti sono estese con il concetto di output
  - la differenza tra macchine di Mealy e di Moore
  - i limiti delle macchine a stati finiti dovuti alla non-composizionalità dei modelli
- Ricordate che:
  - l'output può anche essere rappresentato da un'azione della macchina
- Infine:
  - da ora in avanti per FSM intenderemo una macchina di Mealy

- **Alcune estensioni**

Non determinismo

In caso la macchina sia non determinisca che fare?



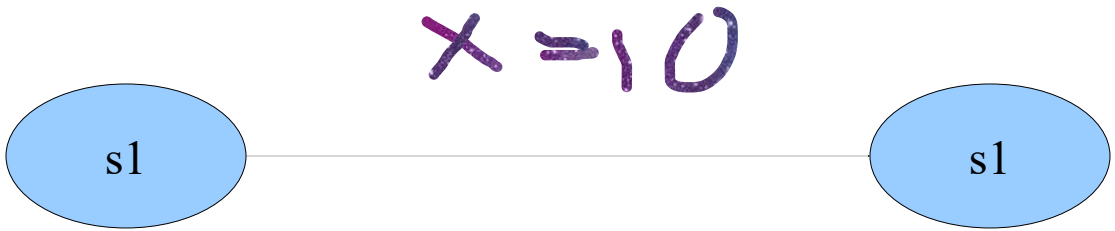
Alcune soluzioni possibili:

- Uso di runtime testing: genero mentre eseguo i casi di test
- Test non come sequenze ma come alberi

Aggiungo variabili

Spesso ho macchine con variabili

Agiungo variabili agli stati
+ guardie e assegnamenti
EFSM



Altri esempi: UML/ Abstract State machines/NuSMV

Esercizio FSM

Scrivi una macchina a stati finiti con almeno 4 stati e due input e due output e trovane un transition tuor (euleriano). Introduci un difetto e scopri se il tuo test è in grado di scoprirlo.

set of transitions $R \subseteq S \times S$

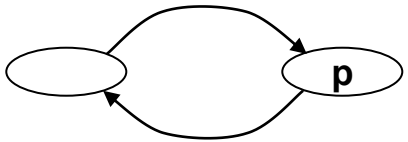
labeling $L : S \to 2^{set(AP)}$

Kripke structures

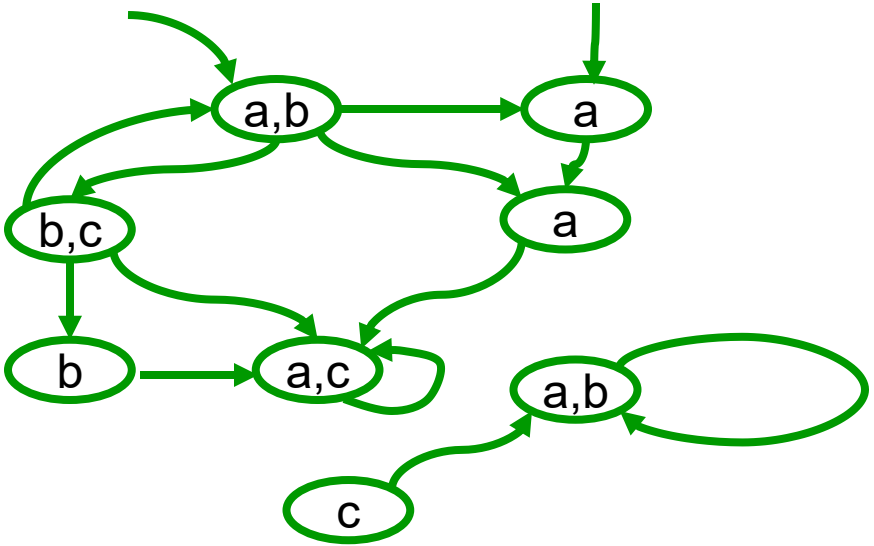
NON HA INPUT NE OUTPUT

Example: Kripke model of a program

```
repeat
    p := true;
    p := false;
end
```



Kripke structure / transition system



Vantaggi Kripke vs FSMs

- **Separazione tra stati e proprietà (etichette logiche)**
- Nelle FSM:
- Le transizioni sono etichettate con input/output.
- Gli stati spesso non rappresentano direttamente proprietà logiche.
- Nelle strutture di Kripke:
- Ogni stato è etichettato con un insieme di **proposizioni atomiche vere in quello stato**.
- Questo permette di esprimere facilmente proprietà logiche sul sistema.

- Supporto naturale al nondeterminismo
- Nelle strutture di Kripke:
- Le transizioni sono semplicemente relazioni.
- Il nondeterminismo è naturale e non richiede meccanismi speciali.
- Nelle FSM:
- Il nondeterminismo può complicare l'analisi.

Esecuzione di una Kripke structure

- π=s0s1s2… is a run in M from s iff
- s = s0 and for every i$\geq$0: (si,si+1)$\in$R

- Ci torneremo