Corso di
Metodi Formali per il Software e i Servizi

(Hands-on)
Il Model Checker NuSMV

Fabio Patrizi
patrizi@dis.uniroma1.it
www.dis.uniroma1.it/~patrizi

NuSMV

- Scaricabile all'indirizzo:
  http://nusmv.irst.itc.it/
- Ultima versione (al 13/5/2009): **2.4.3**
- Per utenti Mac: **2.4.2** (2.4.3 dà problemi)
- Scaricare i sorgenti e compilare seguendo le istruzioni contenuti nei file:
  - nusmv/README e
  - nusmv/README_PLATFORMS
- Oppure scaricare uno dei pacchetti precompilati, se disponibile per la piattaforma desiderata
- Allo stesso indirizzo, sono disponibili anche:
  - Il manuale Utente (riferimento)
  - Un tutorial (Capp. 1,2,4,5 usati per questa lezione)

Generalità

- È un programma per effettuare MC di Transition System (o strutture di Kripke)
- Prende in input:
  - Un TS $\mathcal{M}$
  - Una formula temporale ϕ
- Verifica se $\mathcal{M} \models \phi$

Linguaggio di Specifica

- Un linguaggio per la specifica del TS
- Due linguaggi (molto simili) per la specifica di proprietà temporali:
  - uno per LTL
  - uno per CTL

Specifica del TS

- Modello: Mealy Machine
  - FSM con transizioni non etichettate
- stato: valore delle *variabili*
- transizioni: definite tramite *vincoli*

NOTA: ogni stato deve avere sempre almeno un successore

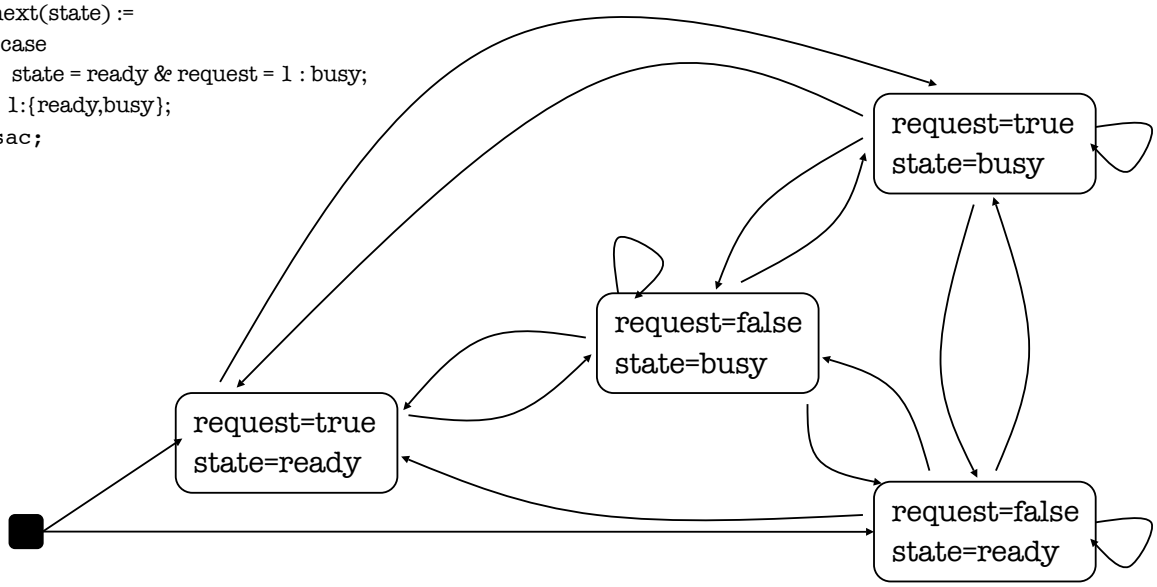
Specifica del TS (2)

Esempio:

```
MODULE main
    VAR
        request: boolean;
        state : {ready,busy};
    ASSIGN
        init(state)=ready;
        next(state) :=
                case
                        state = ready & request = 1 : busy;
                        1:{ready,busy};
                esac;
```

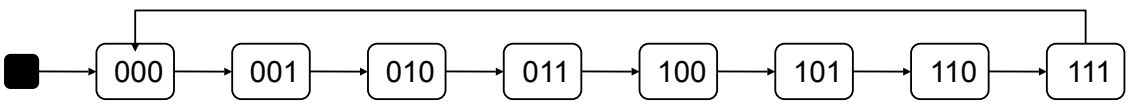
Specifica del TS (3)

```
init(state)=ready;
 next(state) :=
  case
   state = ready & request = 1 : busy;
   1:{ready,busy};
esac;
```



Esercizio 1

- Specificare il contatore modulo 8 rappresentato in figura.



Esercizio 1

(Possibile) Soluzione:

```
MODULE main
    VAR
            x1 : boolean;
            x2 : boolean;
            x3 : boolean;
    ASSIGN
            init(x1) := 0;
            init(x2) := 0;
            init(x3) := 0;
            next(x1) := !x1;
            next(x2) := x1 xor x2;
            next(x3) := (!x3 & x2 & x1) | (x3&!x2) | (x3 & x2 & !x1);
```


Specifica di Proprietà in LTL

Operatori temporali:

- F (eventually)
- G (always)
- U (until)
- X (next)

Es.: il bit x1 dell'esempio precedente alterna sempre il proprio valore tra uno stato ed il successivo:

$$G\ F\ !(x1\ \&\ X\ x1)$$

Specifica di Proprietà in LTL (2)

Altri esempi:

- `G x1 = 1` (in ogni run, il valore del bit x1 è sempre pari ad 1)
- `F x3 = 1` (in ogni run, il valore del bit x3 sarà prima o poi pari ad 1)
- `G F x2 = 0` (in ogni run, il valore del bit x2 sarà pari a 0 infinite volte)
- `!x3 U (x2 & x1)` (in ogni run, il valore del bit x3 è false finché `x2` ed `x1` non diventano entrambi true.

Specifica di Proprietà in LTL (3)

Per verificare una proprietà LTL aggiungiamo, in fondo al modulo `main`, la parola chiave

`LTLSPEC`

Seguita dalla formula che vogliamo verificare

Specifica di Proprietà in CTL

Operatori path-temporali:

AG, AF, A [p U q], AX:

- A: for all paths
- G, F, U, X: analoghi ad LTL

EG, EF, E [p U q], EX:

- E: there exists a path
- G, F, U, X: come sopra

Specifica di Proprietà in CTL (2)

- Esempi:
  - AF x3 = 0
  - EF x3 = 1
  - AG (x3 = 0 -> AF x2 = 0)

Specifica di Proprietà in CTL (3)

Per verificare una proprietà CTL aggiungiamo, in fondo al modulo `main`, la parola chiave

`SPEC`

Seguita dalla formula che vogliamo verificare

Esercizio 2

- Con riferimento all'esempio del contatore modulo 8, verificare, tramite l'uso di NuSMV, la verità o falsità delle formule LTL d'esempio mostrate nelle slide precedenti
- Successivamente, trasformare, dove possibile, le formule LTL in altre equivalenti in CTL e verificarne la verità o falsità con l'ausilio di NuSMV
- Nel caso del contatore modulo 8, esistono proprietà LTL non esprimibili in CTL? E il viceversa?

TS Modulari

- Tipicamente, ci si trova a modellare sottosistemi interagenti
- La specifica della loro interazione è notevolmente semplificata se:
  - Specifichiamo ciascun sottosistema separatamente
  - Successivamente, integriamo i sottosistemi (nel modulo `main`)
  - … piuttosto che combinare tutto in un unico modulo main
- In tal modo, inoltre, favoriamo il *riuso*

Specifiche Modulari - Esempio

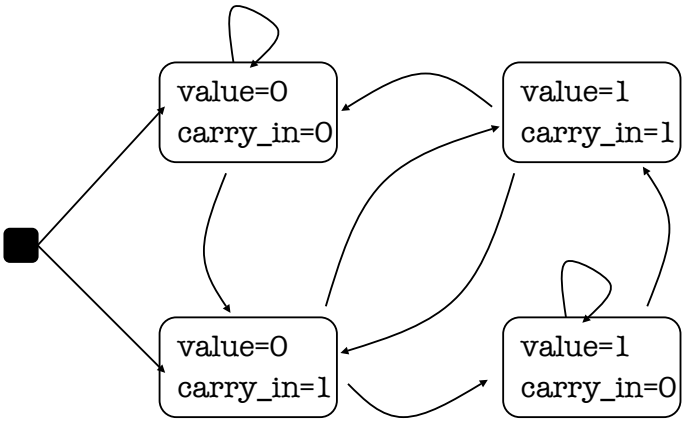
- Vogliamo definire il contatore modulo 8
  1. Costruiamo un singolo contatore binario
  2. Connettiamo 3 contatori binari

- Contatore Binario Singolo

```
MODULE counter_cell(carry_in)
     -- carry_in è controllato esternamente
     VAR
               value : boolean;
     ASSIGN
               init(value) := 0;
               next(value) := (value + carry_in) mod 2;
     DEFINE
     -- sezione per la definizione di espressioni
               carry_out := value & carry_in;
```

Specifiche Modulari - Esempio (2)

```
init(value) := 0;
              next(value) := (value + carry_in) mod 2;
```



Specifiche Modulari - Esempio (3)

- Connettiamo 3 contatori binari nel modulo main:

```
MODULE main
    VAR
              bit0 : counter_cell(1);
              bit1 : counter_cell(bit0.carry_out);
              bit2 : counter_cell(bit1.carry_out);
              -- carry_out è l'espressione definita in DEFINE
```

Esercizio 3

- Sfruttando la composizione di moduli come nell'esempio del contatore modulo 8, realizzare un contatore modulo 16.

Composizione Asincrona

- Nell'esempio precedente i sottosistemi (cioè i contatori binari) evolvono in maniera ***sincrona***:
  - ad ogni passo, ciascuno di essi effettua una transizione di stato
- NuSMV permette **anche la composizione *asincrona***:
  - Ad ogni passo, solo un modulo, **scelto nondeterministicamente**, esegue una transizione

Composizione Asincrona (2)

- Per comporre in maniera **asincrona** i moduli, si usa la parola chiave `process`

```
MODULE main
    VAR
            bit0 : process counter_cell(1);
            bit1 : process counter_cell(bit0.carry_out);
            bit2 : process counter_cell(bit1.carry_out);
```

  - Tutte le variabili non presenti nella specifica del modulo selezionato, *come variabili o parametri*, **restano immutate**
  - Non c'è garanzia che ciascun processo venga**, prima o poi, selezionato**

Composizione Asincrona (3)

- Per garantire che, a partire da uno stato qualsiasi, ciascun processo sia prima o poi selezionato, aggiungiamo, in fondo alla specifica del processo l'espressione :

  `FAIRNESS running`

NOTA: È un vincolo di *fairness incondizionata* (o semplice) del tipo □♢ p

Esercizio 4

Due processi asincroni condividono una risorsa ad *accesso esclusivo, regolato da un semaforo*.

Ciascun processo può *richiedere* l'accesso alla risorsa, averlo *acquisito*, *rilasciare* la risorsa oppure essere *nonInteressato*.

1. Specificare un protocollo d'accesso che garantisca ai processi di non accedere mai contemporaneamente alla risorsa, garantendo comunque a tutti la possibilità di accesso.
2. Verificare che, con esecuzione asincrona, valgano le seguenti proprietà:
   - I due processi non possono mai acquisire contemporaneamente accesso alla risorsa;
   - Se un processo richiede accesso alla risorsa, prima o poi lo acquisisce;
   - Se un processo richiede la risorsa infinite volte, allora vi acquisice l'accesso infinite volte.

Quando possibile, se necessario, modificare il protocollo in maniera tale da garantire tali proprietà.

Esercizio 5

L'incrocio tra via Rossi e via Verdi, mostrato in figura, è controllato da un singolo semaforo intelligente, dotato di fotocellule per il rilevamento del flusso di traffico. Quando è verde per via Verdi, il semaforo è rosso per via Rossi e viceversa. Le linee numerate rappresentano la posizione delle fotocellule e vengono usate per la gestione del flusso.

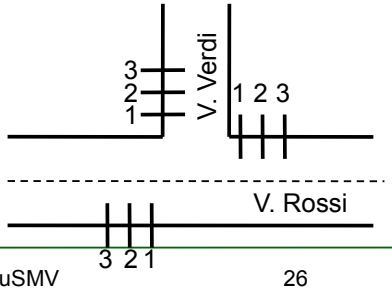
Ad ogni ramo dell'incrocio viene associato un **indice di flusso**, ovvero l'intero massimo che contrassegna una delle linee raggiunte dalla coda di autovetture in attesa in quel ramo.

Il semaforo diventa verde per la strada contenente uno dei rami con indice di flusso massimo e rimane tale finché tutti i rami della strada non si sono svuotati (indice di flusso = 0).

Se esistono diversi rami con indice di flusso massimo, viene effettuata una scelta nondeterministica.

Quando il semaforo diventa verde per una via, gli indici di flusso dei suoi rami diminuiscono progressivamente fino a raggiungere il valore 0. Solo a quel punto, possono tornare ad aumentare (progressivamente).

Fornire la specifica del sistema sopra descritto per NuSMV



Esercizio 5 (2)

- Verificare, con l'ausilio di NuSMV, le seguenti proprietà espresse in LTL o in CTL (entrambe, quando possibile):
  - Non è possibile che il semaforo rimanga rosso indefinitamente per una stessa strada
  - Esiste una particolare sequenza di arrivo delle autovetture tale che, ad un certo punto, il semaforo in via Verdi rimanga rosso per sempre.
  - Se ci sono autovetture in attesa in un ramo (indice di flusso >= 1), prima o poi il semaforo diventerà verde per quel ramo
  - Se il semaforo diventa rosso infinite volte per via Verdi, allora diventa verde infinite volte per via Rossi
  - Esiste una particolare sequenza di arrivo delle autovetture tale che il semaforo cambia stato infinite volte
  - Il semaforo cambia stato infinite volte, indipendentemente dalla sequenza di arrivo delle autovetture
  - Può accadere che un ramo si stia svuotando mentre il semaforo è rosso per la strada cui il ramo appartiene
  - Se un ramo inizia a svuotarsi allora si svuoterà effetttivamente