Laboratorio dell'11 Novembre 2019

1 Esercizio preliminare

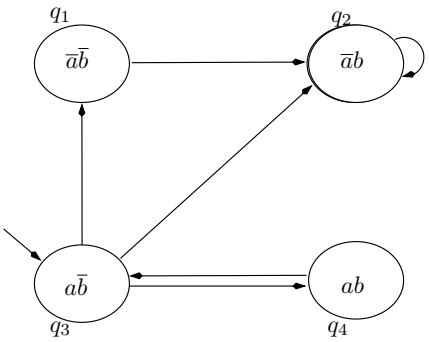

Figure 1: Modello di sistema per Esercizio 1.

Crea un modello NuSMV per il sistema mostrato nella Figura 1. Per ognuna delle seguenti formule di LTL, usa NuSMV per *(i)* determinare se la formula è verificata dal sistema, e *(ii)* trovare un'esecuzione che soddisfa la formula.

(a) $\Box a$
(b) $\Box(\neg a \rightarrow \bigcirc b)$
(c) $a \,\mathcal{U}\, b$
(d) $a \,\mathcal{U}\, \bigcirc(a \wedge \neg b)$
(e) $\bigcirc\neg b \wedge \Box(\neg a \vee \neg b)$
(f) $\bigcirc(a \wedge b) \wedge \Diamond(\neg a \wedge \neg b)$

Verifica che le risposte che ricevi con NuSMV corrispondano alla tua comprensione del modello e delle formule. In particolare, perché in alcuni casi sia la formula che la sua negazione possono essere false.

Inserisci le risposte nel file `ltl-exercise.smv` e caricalo nel database "Soluzioni Esercizio 1". Nella parte superiore del file inserisci il modello. Quindi, per ciascuna parte dell'esercizio, inserisci il codice NuSMV per la formula LTL, indicando se la formula è verificata dal sistema, ed un esempio di esecuzione che soddisfa la formula.

Per scrivere le `LTLSPEC` usa la sintassi NuSMV per gli operatori logici:

$\wedge$ `&` $\vee$ `|` $\neg$ `!` $\rightarrow$ `->` $\Box$ `G` $\Diamond$ `F` $\bigcirc$ `X` $\mathcal{U}$ `U`

2 Verifica di un buffer FIFO

In questo esercizio ti si chiede di verificare alcune proprietà del buffer FIFO mostrato nella Figura 2.

In astratto, un buffer FIFO è una coda a lunghezza variabile. Ha due interfacce, un'interfaccia di input per l'inserimento di nuovi elementi a un'estremità della coda e un'interfaccia di output per la lettura e la rimozione di elementi dall'altra estremità della coda.

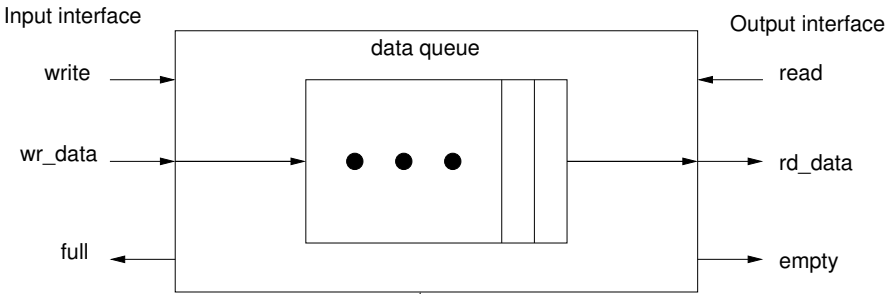

Figure 2: Diagramma a blocchi del buffer FIFO.

Per aggiungere un elemento al buffer, il segnale Booleano `write` viene impostato a vero ed i dati vengono inviati sul *data input* `wr_data`. A condizione che il buffer non sia pieno, il nuovo elemento viene aggiunto alla coda nello step successivo. La coda ha un numero massimo di elementi che può contenere in qualsiasi momento. L'output Booleano `full` indica se la coda è piena o no.

Il *data output* `rd_data` mostra l'elemento finale della coda, a condizione che la coda non sia vuota. La coda vuota è segnalata dall'output Booleano `empty` impostato a vero. Se l'input Booleano `read` è impostato a vero e la coda non è vuota, l'elemento finale della coda viene rimosso e l'elemento immediatamente precedente (se presente) appare nell'ouput `rd_data` nello step successivo.

Il file `fifo.smv` contiene il modello NuSMV del buffer FIFO. Per semplicità e per garantire tempi rapidi di esecuzione di NuSMV, la costante `DEPTH` con il numero massimo di elementi nella coda è impostata a 5 e la costante `WIDTH` per la dimensione in bit degli elementi a 1. In un sistema reale questi valori sono tipicamente molto più grandi.

Internamente, il sistema utilizza un buffer circolare per implementare la coda. Questo è costituito da un array di `word` di dimensione `DEPTH` e due puntatori: il puntatore di lettura `rp_p` e il puntatore di scrittura `wr_p`. Se la coda non è vuota, il puntatore di lettura punta all'ultimo elemento della coda. Se la coda non è piena, il puntatore di scrittura punta alla posizione dove inserire il prossimo elemento. Quando un nuovo elemento viene inserito nella coda, il puntatore di scrittura viene incrementato, facendolo ritornare a 0 quando necessario. Quando viene rimosso un elemento, il puntatore di lettura viene incrementato, facendolo ritornare a 0 quando necessario. La Figura 3 mostra due esempi della configurazione interna del sistema dove la coda contiene gli elementi `w0`, `w1` e `w2`, inseriti in quest'ordine.

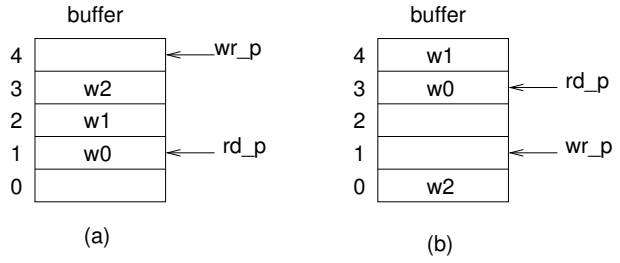

Figure 3: Esempi di configurazione interna del buffer FIFO.

Nell'implementazione del buffer FIFO fornita, la coda potrebbe essere vuota o piena quando i due puntatori sono uguali. Il modello utilizza la variabile di stato Booleana `empty` per distinguere tra questi due casi.

2.1 Proprietà da verificare

Aggiungi le formule LTL per le proprietà richieste al file `fifo-properties.smv` e caricalo nel database "Soluzioni Esercizio 2.1". Verifica le proprietà con NuSMV usando il comando

```
NuSMV -pre cpp fifo.smv
```

Il file `fifo.smv` importa il file `fifo-properties.smv` usando una direttiva `#include`. L'opzione `-pre cpp` per NuSMV è necessaria per eseguire il preprocessore C sul file `fifo.smv` ed interpretare la direttiva `#include` e le definizioni delle costanti `WIDTH` e `DEPTH`.

Nota: otterrai dei warning da NuSMV finché non rimpiazzi le specifiche LTL `TRUE` con delle formule significative.

Scrivi delle formule LTL per le seguenti proprietà (tutte verificate dal modello `fifo.smv`):

(a) *Non è mai vero che il buffer FIFO indica simultaneamente di essere sia pieno che vuoto.*

(b) *Se* `write` *è ricorrente e* `read` *non è ricorrente, allora la coda prima o poi diventa piena.*

(c) *In ogni momento, se il valore* 1 *viene inviato al data input e* `write` *è vero, allora prima o poi il valore* 1 *apparirà come data output,*

sotto ulteriori ipotesi ragionevoli per i segnali `read`, `empty` o `full`, che garantiscano che la proprietà sia vera. Consulta le informazioni sulle *word constants* presenti nel manuale di NuSMV per sapere come fare riferimento al numero 1.

(d) la stessa proprietà di (c), formulata per considerare qualsiasi valore di input, non solo il valore 1. Utilizza la "frozen variable" `data1` per scrivere la proprietà. Consulta il manuale di NuSMV per la documentazione sulle *frozen variables* (note anche come *variabili rigide* nella logica temporale).

(e) la stessa proprietà di (d), con la richiesta aggiuntiva che l'output `empty` sia falso in ogni momento compreso tra il momento in cui il dato è scritto ed il momento in cui il dato viene letto, estremi esclusi. Puoi trarre vantaggio dal fatto che il primo momento in cui il dato può essere letto è l'istante successivo alla scrittura.

(f) una proprietà simile a (d), che presuppone che due valori (anche distinti) vengano scritti consecutivamente e che controlla che gli stessi due valori siano presentati come output nello stesso ordine con cui sono stati scritti. Utilizza le frozen variable `data1` e `data2` per fare riferimento ai due valori.

Quando scrivi le formule NuSMV, tieni presente che la precedenza degli operatori LTL è `F G X ! U & | ->` (da più forte a più debole).

2.2 Correzione di un bug

L'implementazione del buffer FIFO ha un bug. In questo esercizio lo scopri e lo risolvi.

1. Nel posto indicato in `fifo-properties.smv`, scrivi una proprietà LTL che controlli che

   > *in ogni momento, se il buffer FIFO segnala che è vuoto, allora i puntatori di lettura e scrittura devono essere uguali.*

NuSMV dovrebbe trovare falsa la proprietà e mostrare un controesempio.

2. Creare una copia di `fifo.smv` chiamata `fifo-fixed.smv`. Modifica il codice del modulo `main` nel file `fifo-fixed.smv` per correggere questo errore. Le tue modifiche dovrebbero affrontare il problema generale identificato da questo errore.

3. Carica il file `fifo-fixed.smv` nel database "Soluzioni Esercizio 2.2"

3 L'ascensore

Considerate l'esempio dell'ascensore, descritto nel documento "Elevator Example" del laboratiorio 1. Il file `elevator.smv` contiene l'implementazione di tutte le componenti tranne il controllore, che genera nondeterministicamente tutti i possibili output.

1. Completate l'implementazione del controllore.

2. Verificate il corretto funzionamento del sistema rispetto agli scenari descritti nella bacheca. Per ogni scenario, definite uno o più invarianti che vi consentano di verificare il sistema.

3. Per ogni scenario risolto, caricate la soluzione nel database "Soluzioni Esercizio 3"