Modelli Concorrenti e Algoritmi Distribuiti
a.a. 2021/2022
Prova d'esonero 30 novembre 2021

Esercizio 1

Un sistema è costituito da due processi concorrenti A e B. Ogni processo ciclicamente esegue due operazioni in sequenza (A1 e A2 per il processo A e B1 e B2 per il processo B). In ogni ciclo l'operazione A2 può iniziare solo se l' operazione B1 del ciclo corrispondente è terminata e viceversa l'operazione B2 può iniziare solo se l' operazione A1 del ciclo corrispondente è terminata. Nel seguito viene riportata una soluzione che utilizza i semafori.

```
semaphore synchA = 0;
semaphore synchB = 0 ;
    void A( ){
    while(true) {
             A1;
             V(synchA);
             P(synchB);
             A2;
         }
    }

    void B( ){
       while(true) {
             B1 ;
             V(synchB);
             P(synchA);
             B2;
         }
    }
void main( ){
cobegin {A( ); B( );}
}
```

Se *n(op)* indica il numero di esecuzioni completate dell'operazione *op* in un qualsiasi stato *s,* quale tra le relazioni sottoelencate costituisce un invariante per questo soluzione?

a. $0 \leq n(A1) - n(B2) \leq 1$
b. $-1 \leq n(A1) - n(B2) \leq 1$
c. $0 \leq n(A1) - n(B2) \leq 2$

Si dimostri che la relazione scelta è un invariante.
(Suggerimento: valgono le relazioni: $0 \leq n(A1) - n(A2) \leq 1$ e $0 \leq n(B1) - n(B2) \leq 1$ sono gli invarianti dei cicli **while;** di tali invarianti NON è richiesta la prova)

Esercizio 2

Si consideri il problema dell'esercitazione "in coda dal fornaio", con la seguente modifica.
Sopra il banco del fornaio è presente un display luminoso che mostra:

- il valore del biglietto in corso di servizio, se il fornaio sta servendo un cliente;
- quello dell'ultimo cliente servito, se il fornaio sta leggendo il giornale.

Inizialmente il display mostra il valore 0.

Ogni cliente, dopo aver preso il suo biglietto fa una pausa per τ minuti $(0 \leq \tau \leq 10)$; la durata della pausa è una scelta soggettiva del cliente. Dopo tale periodo, il cliente confronta il numero del proprio biglietto con il valore del display: se il valore del display è maggiore del suo numero esce senza acquisti, altrimenti si mette in attesa della chiamata. Quando il fornaio lo chiama, il cliente gli fornisce l'ordine, aspetta di essere servito ed esce.

Se non ci sono clienti in attesa, il fornaio legge il giornale, altrimenti considera il biglietto successivo a quello mostrato dal display, verifica la presenza del cliente proprietario del biglietto considerato: se questi non è in attesa passa a considerare il biglietto successivo e così via, altrimenti aggiorna il display, chiama il cliente proprietario del biglietto mostrato dal display, aspetta il suo ordine, lo serve e lo congeda.

```
"Fornaio"
loop forever
   <se non ci sono clienti leggi il giornale>
   <considera il numero successivo al valore del display>
  while  <il cliente possessore del numero considerato non è presente>
         <considera il numero successivo>
  <aggiorna il display con il numero considerato>
  <chiama il cliente il cui biglietto corrisponde al valore del display>
  <prendi l'ordine>
  <servi>
  <congeda il cliente>
```

```
"cliente_i"
:
<preleva un numero>
<fa una pausa di T_i minuti>
<se il tuo numero è maggiore del numero del display:
       <mettiti in coda e aspetta la chiamata del tuo numero>
       <ordina la spesa>
       <aspetta che il fornaio ti congedi>
<esci dal negozio>
:
```

Realizzare questo sistema con il costrutto monitor (semantica *signal_and_urgent_wait*), facendo in modo che nei periodi in cui il fornaio è occupato a servire un cliente, altri clienti possano prelevare il biglietto e mettersi eventualmente in attesa della chiamata.

Modelli Concorrenti e Algoritmi Distribuiti
Prova d'esonero 30 novembre 2021
Foglio Risposte

Esercizio 1

la corretta invariante è

c. $0 \leq n(A1) - n(B2) \leq 2$

Si dimostri che la relazione scelta è un invariante.

Invarianti semaforici : NP(syncA) = 0 + NV(synchA)
NP(synchB) = 0 + NV(synchB)

Dal codice otteniamo i seguenti invarianti topologici :

N(A2) <= NP(synchB) <= NV(synchA) <= N(A1)

N(B2) <= NP(synchA) <= NV(synchB) <= N(B1)

Unendo i due possiamo ottenere :

N(B2) <= NP(synchA) <= NV(synchA) <= N(A1) per il 2° invariante semaforico

Essendo N(B2) <= N(A1) abbiamo che 0 <= N(A1) – N(B2) quindi abbiamo dimostrato la prima disequazione dell'invariante c

Dimostriamo la seconda riscrivendo l'invariante del ciclo while di A in questo modo :

$0 \leq n(A1) - n(A2) \leq 1$: $n(A2) \leq n(A1) \leq 1 + n(A2)$

Grazie alla combinazione del primo inv semaforico con il secondo inv topologico sappiamo che : $n(A2) <= n(B1)$
Quindi :

$n(A1) \leq 1 + n(A2) <= 1 + n(B1)$

riscriviamo l'invariante del ciclo while del processo B

$0 \leq n(B1) - n(B2) \leq 1$: $n(B2) \leq n(B1) \leq 1 + n(B2)$

Ed otteniamo :

$n(A1) \leq 1 + n(A2) <= 1 + n(B1) <= 1 + 1 + n(B2)$ $n(A1) <= 2 + n(B2)$ $n(A1) - n(B2) <= 2$

cioè la seconda parte dell'inviariante c

unendo le due parti dimostrate abbiamo che

$0 \leq n(A1) - n(B2) \leq 2$

Esercizio 2
QUESTA RISOLUZIONE PUÒ ESSERE MIGLIORATA:
in primis avere un solo monitor riduce il parallelismo, sarebbe meglio averne 2-3 (un fornaio ed una macchinetta tipo).
è stupido anche far fare un ciclo ogni numero chiamato. Proseguire con i numeri all'interno di ChiamoProssimo() ed uscire dalla funzione solo quando un cliente è stato trovato, altrimenti si fanno un po' troppe operazioni inutili.

```
const int MAX <- 100;

monitor forno {
        condition biglietti[MAX];
        integer num_clienti <- 0;
        boolean presenti[MAX] <- false;
        integer last,serving <- 0;
//il valore della variabile "serving" è ciò che viene mostrato sul display
        integer numeroCliente <- 0;
        integer clientiInCoda <- 0;
        condition full, baker, empty, ordine;

        controllaClienti(){
        if (clientiInCoda == 0) //il negozio è vuoto, legge il giornale.
                wait(empty);
        }

        public integer prendiNumero(){
                if(num_client == MAX) wait(full);
                last <- (last + 1) % MAX;
                num_clienti++;
                return last;
        }

        public boolean mettitiInCoda(integer numeroCliente){
                if(numeroCliente > serving){
                        presenti[numeroCliente] <- true;
                        clientiInCoda ++;
                        signal(empty);
                        wait(biglietti[numeroCliente]);
                        return true;
                }
                else
                        return false;
        }

        public boolean chiamoProssimo(){
                serving <- (serving + 1)  %MAX //incrementa il display
                if (presenti[serving]){
                        signal(biglietti[serving]);
                        wait(ordine);
                        return true
                }// se c'è qualcuno in attesa su quel numero chiamalo
                else
                        return false;
        }

        public void ordina(String spesa){
                signal(ordine);
                wait(bigliettto[serving]);
```

```
        }

        public void servi(){
                signal(biglietto[serving]);
                wait(baker);
        }


public void esci(numeroCiente){
                if(presenti[numeroCliente]){
                        presenti[numeroCliente] <- false;
                        clientiInCoda --;
                }
                num_clienti--;
                signal(full);
                signal(baker);
        }
}

Fornaio
        loop forever:
                forno.controllaClienti();
                if(forno.chiamaProssimo()){
                        <prepara ordine>
                        forno.servi();
                }// se quel numero non c'è non deve servire nessuno

Cliente
        String spesa;
        Integer mioNumero;

        mioNumero <- forno.prendiNumero();
        <sleep T >
        If( forno.mettitiInCoda(mioNumero))
                forno.ordina(spesa);
        forno.esci(mioNumero);
        <esci dal negozio>



//ho pensato al fatto che potrebbe verificarsi un problema nel caso in cui un
cliente sia rimasto in attesa fino al "reset" del contatore.
Ad esempio giunge il cliente numero 2, prende il numero e aspetta 101 "turni", a
quel punto il display si è resettato ed il cliente torna al banco del panettiere e
crede sia il suo turno (0 < 2), in verità non è così perché il suo turno è stato
saltato.
Non ho trovato una soluzione a questo problema, ma ho notato che tale problema è
possibile si verifichi anche nella realtà.
```