Modelli Concorrenti e Algoritmi Distribuiti

Esercizi sull'uso di invarianti semaforici

Esercizio 1

Il codice sotto riportato modella il comportamento di un sistema costituito da un pasticcere e da n clienti. I clienti si servono dei biscotti prelevandoli uno alla volta da un vassoio in grado di contenerne m. Quando un cliente che desidera mangiare un biscotto si accorge che il vassoio è vuoto, avverte il pasticcere che cuoce m biscotti e li pone sul vassoio.

```
semaphore   mutex = 1
semaphore   pieno = 0
semaphore   vuoto = 0
int   porzioni =  0

void Client(){
:
P(mutex)
if (porzioni = = 0) {
      V(vuoto);
      P(pieno);
      };
porzioni --;
V(mutex);
<mangia biscotto>;
:
}

void Pasticcere (){
:
while(true) {
      P(vuoto);
      <prepara biscotti e riempi il vassoio>;
      porzioni = m;
      V(pieno);
      }
}
```

Si provi che :

a) non esiste un istante t, in cui il pasticcere sta preparando i biscotti e un cliente esegue il decremento sulla variabile `porzioni.`
b) il sistema è deadlock-free.

Esercizio 2

Lo schema di codice sotto riportato si riferisce ad un sistema costituito da due processi concorrenti: il processo **P** che stampa la sillaba “TA” e il processo **Q** che stampa la sillaba “RA”. La stampante è la stessa e la stampa di una sillaba è un’operazione atomica, per cui l’esecuzione concorrente dei due programmi provoca la stampa di una sequenza di sillabe TA e RA di lunghezza indefinita. La sincronizzazione è realizzata tramite due semafori.

a) Quale delle seguenti sequenze può essere il prefisso di una sequenza generata dall’ esecuzione concorrente dei due processi?

1. TA TA RA TA TA RA
2. TA RA TA TA TA RA
3. TA RA TA TA TA TA

b) Quale delle seguenti disuguaglianze è vera in ogni istante t dell’esecuzione concorrente dei due processi (NTA(t) e NRA(t) indicano rispettivamente il numero di sillabe “TA” e “RA” stampati sino all’istante t)? Si dimostri la validità della diseguaglianza scelta utilizzando gli invarianti semaforici.

1. $0 \leq NTA(t) - 3NRA(t) \leq 1$
2. $0 \leq NTA(t) - NRA(t) \leq 3$
3. $-3 \leq NTA(t) - 4NRA(t) \leq 1$

c) Si dimostri che non esiste possibilità di deadlock.

```
semaphore semA = 1;
semaphore semB = 3;

process P {
      while (true) {
        P(semA);
        <stampa “TA”>;
        V(semB);
      }
}

process Q {
      while (true) {
        P(semB);
        P(semB);
        P(semB);
        P(semB);
        <stampa “RA”>;
        V(semA);
        V(semA);
        V(semA);
        V(semA);

      }

}
```

Esercizio 3

Lo schema di codice sotto riportato si riferisce ad un sistema costituito da un ORSO e da un numero n ($n \geq 1$) di API. Le api depongono, una alla volta, una quantità fissa (quantum) di miele in un favo che ha la capacità di 4 quantum. L'orso accede al favo solo quando questo è completamente pieno e lo svuota completamente. Il favo è condiviso per cui le operazioni di deposito e di prelievo vanno eseguite in mutua esclusione.
Provare che:
.

a) la soluzione è corretta, i.e. detti Nd(t) e Ns(t) rispettivamente il numero di operazioni di deposito e di svuotamento eseguite fino all'istante t:

$$0 \leq Nd(t) - 4* Ns(t) \leq 4$$

b) non è possibile che l'orso svuoti il favo mentre un'ape sta deponendo un quantum di miele;

```
semaphore full = 0;
semaphore empty = 4;
semaphore mutex = 1;

process APEj {
      while (true) {
        P(empty);
        P(mutex);
        <deposita un quantum>;
        V(mutex);
        V(full);
      }
}

process ORSO {
      while (true) {
        P(full);
        P(full);
        P(full);
        P(full);
        <svuota il favo>;
        V(empty);
        V(empty);
        V(empty);
        V(empty);

      }

}
```

Esercizio 4

Si considerino due processi concorrenti $\mathbf{P}_a$ e $\mathbf{P}_b$:

```
process Pa {
      a = a0
      while (true)
              a++;
      }

process Pb {
      b = b0
      while (true)
              b++;
      }
```

I valori iniziali a_0 e b_0 soddisfano le disuguaglianze:

$a_0 \leq b_0$

$b_0 \leq 2a_0$

a) Si introducano nei codici di $\mathbf{P}_a$ e $\mathbf{P}_b$ le opportune primitive semaforiche in modo che, detto `val(a,t)` il valore della variabile a all'istante `t` e `val(b,t)` il valore della variabile b all'istante `t`, le disuguaglianze

1. `val(a,t)` $\leq$ `val(b,t)`
2. `val(b,t)` $\leq$ 2 `val(a,t)`

siano verificate in ogni istante `t`. Si dimostri poi la correttezza della soluzione proposta usando gli invarianti semaforici.

b) Si discuta il problema del deadlock.

Esercizio 5

Lo schema di codice sotto riportato si riferisce ad un sistema costituito da due processi concorrenti: il processo $\mathbf{P_A}$ che stampa il carattere “a” e il processo $\mathbf{P_B}$ che stampa il carattere “b”. La stampante è la stessa e la stampa di un carattere è un’operazione atomica, per cui l’esecuzione concorrente dei due programmi provoca la stampa di una sequenza di caratteri a e b di lunghezza indefinita. La sincronizzazione è realizzata tramite due semafori.

a) Quali delle seguenti sequenze possono essere prefissi di sequenze generate da una esecuzione concorrente dei due processi?

1. aabbabb
2. abaabba
3. aabbaab

b) Quale delle seguenti disuguaglianze è vera in ogni istante t dell’esecuzione concorrente dei due processi (na(t) e nb(t) indicano rispettivamente il numero di caratteri “a” e “b” stampati sino all’istante t)? Si dimostri la validità della diseguaglianza scelta utilizzando gli invarianti semaforici.

1. $0 \leq na(t) - 2* nb(t) \leq 1$
2. $0 \leq na(t) - nb(t) \leq 2$
3. $1 \leq na(t) - nb(t) \leq 3$

c) Si dimostri che non esiste possibilità di deadlock.

```
semaphore semA = 2;
semaphore semB = 0;

process PA {
      while (true) {
        P(semA);
        <stampa “a”>;
        V(semB);
      }
}

process PB {
      while (true) {
        P(semB);
        <stampa “b”>;
        V(semA);
      }

}
```

Esercizio 6

Si consideri la seguente variante al problema del Produttore/Consumatore in cui il processo Consumatore preleva i dati a coppie, mentre il processo Produttore li inserisce uno alla volta. Il buffer è gestito con un vettore circolare di dimensione N con N >1.

```
T           buffer[N]
int         testa = 0
int         coda = 0
semaphore   full = 0
sempahore   empty = N
```

```
process Produttore {
T  d;
  while (true) {
    <produci d>
    P(empty);
    buffer[coda] = d;
    coda = (coda+1)%N;
    V(full);
  }
}
```

```
Process Consumatore {
T x1 x2;
   while (true){
     P(full);
     P(full);
     x1 = buffer[testa];
     x2 = buffer[(testa+1)%N;
     testa = (testa +2)%N;
     V(empty);
     V(empty);

   }
}
```

Usando gli invarianti semaforici dimostrare che:

a) in ogni istante t_e in cui il processo Consumatore inizia l'estrazione della coppia di valori dal buffer (quindi dopo aver eseguito le due primitive **P**(full)), nel buffer sono disponibili almeno due dati, i.e.

$$N(d,t_e) - N(e,t_e) \geq 2$$

[N(d,t) rappresenta il numero di operazioni di deposito effettuate fino all'istante t, mentre N(e,t) rappresenta il numero di di operazioni di estrazioni effettuate sino all'istante t.]

b) il sistema non va in deadlock

Esercizio 7

Lo schema di codice sotto riportato si riferisce ad un sistema costituito da due processi concorrenti: il processo $\mathbf{P_A}$ che stampa il carattere "(" e il processo $\mathbf{P_C}$ che stampa il carattere ")". La stampante è unica e la stampa di un carattere è un'operazione atomica, per cui l'esecuzione concorrente dei due processi provoca la stampa di una sequenza di parentesi aperte e chiuse di lunghezza indefinita. Si vuole che per ogni sequenza di caratteri stampati siano soddisfatti i seguenti requisiti:

(1) la sequenza è *ben parentesizzata*, cioè per ogni suo prefisso t il numero di parentesi aperte è sempre maggiore o uguale al numero delle parentesi chiuse:

$$\forall t.\quad num(\text{“(“},t) - num(\text{“)”},t) \leq 0$$

(2) per ogni prefisso t, il livello di annidamento, cioè la differenza tra il numero di parentesi aperte e il numero di parentesi chiuse, è sempre minore o uguale a 3:

$$\forall t.\quad num(\text{“(“},t) - num(\text{“)”},t) \leq 3$$

(num (σ,t) indica numero di occorrenze del simbolo σ nel prefisso t).

d) Si inseriscano negli schemi dei programmi di $\mathbf{P_A}$ e $\mathbf{P_C}$ le opportune primitive semaforiche e si indichino le inizializzazione dei semafori, in modo che le diseguaglianze riportate in (1) e in (2) siano verificate.
e) Si dimostri la correttezza della soluzione.
f) Si discuta il problema del deadlock.