Semafori Operazionali

Un semaforo operazionale ha una sola chiamata:

```
S.semop (int valore)
```

dove *valore* se positivo rappresenta una restituzione di risorse, se negativo una richiesta (in modo simile alla V e alla P). Se il valore negativo è superiore in valore assoluto al numero di risorse disponibili la semop diviene bloccante. Il processo verrà riattivato non appena ci saranno risorse disponibili a sufficienza perché la richiesta possa essere completamente soddisfatta. Le richieste pendenti vengono estratte e soddisfatte in modo FIFO.

a) Scrivere l'invariante

b) Dimostrare semafori operazionali hanno lo stesso potere espressivo dei semafori ordinari

Soluzione

a) Scrivere l'invariante

Sia init il valore iniziale del semaforo.

**Sia SP l'insieme di operazioni semop completate a un certo istante,
tali per cui il valore era negativo**

**Sia SV l'insieme di operazioni semop completate a un certo istante,
tali per cui il valore era positivo**

Sia NP il valore assoluto della sommatoria dei valori delle operazioni semop contenute in SP

Sia NV il valore della sommatoria dei valori delle operazioni semop contenute in SV

L'invariante e':

NP <= NV + init

b) Dimostrare che semafori operazionali hanno lo stesso potere espressivo dei semafori ordinari

Primo passo: implementare semafori normali utilizzando semafori operazionali

```
class Semaphore
{
  OperationalSemahore sem;

  Semaphore(int init)
  {
    sem = new OperationalSemaphore(init);
  }

  void P()
  {
    sem.semop(-1);
  }

  void V()
  {
    sem.semop(1);
  }
}
```

Secondo passo: implementare semafori operazionali utilizzando semafori normali

```
semop(int val)
{
  if (val < 0) {
    < await(value+val >= 0) -> value=value+val; >
  } else if (val > 0) {
    < value = value + val >
  }
}

class OperationalSemaphore {
  Semaphore mutex = new Semaphore(1); // Semaforo di mutua
esclusione
  Semaphore attesa = new Semaphore(0); // Semaforo di sospensione
  int counter = 0; // Numero di elementi sospesi su attesa
  Queue queue = new Queue();
       // Contiene i valori richiesti dai semafori
  int value = 0; // Valore del semaforo operazionale

  Semaphore(int init)
  {
    value = init;
  }

  void semop(int val)
  {
    if (val < 0) {
      mutex.P();
      if ( value + val < 0 ) {       //!(value + val >=0)
        counter++;
        queue.add(val);
        mutex.V();
        attesa.P();
      }
      value = value + val;
      SIGNAL();
    } else if (val > 0) {
      mutex.P();
      value = value + val;
      SIGNAL();
    }
  }

  /* Nota: in questo caso, le strutture dati di "base" della
 soluzione di Andrews non bastano; dobbiamo aggiungere una coda,
 per poter sapere qual e' il valore del prossimo elemento
     Ho anche spostato il controllo su counter */


  SIGNAL()
  {
```

```
    if (counter > 0 && value + queue.get() >= 0) {
      counter--;
      queue.remove();
      attesa.V();
    } else {
      mutex.V();
    }
  }
}
```