Esercizio E3.3

Impostazione

Al posto delle entry dovremo adesso utilizzare le porte. Inoltre, dovendo utilizzare le primitive di comunicazione asincrone, sarà necessario prevedere esplicitamente lo scambio di alcuni messaggi con l'unico scopo di garantire esplicitamente lo stesso livello di sincronizzazione tra processi implicitamente indotto negli stessi dall'uso delle primitive relative al *rendez-vous*.

Soluzione

Il processo `P1` chiama la entry `E1` di `P3`. In `P3` la dichiarazione di tale entry prevede il parametro `y` di tipo `int` e modo `out`. Quindi, alla fine della chiamata di `P3.E1(b)`, a `P1` viene restituito un valore. È quindi necessario dichiarare, locale a `P1`, una porta (`risultato`) di tipo `int` da cui lo stesso possa ricevere il valore che la entry `E1` restituisce. Inoltre, in `P3` dovremo dichiarare al, posto della entry `E1`, una corripondente porta da cui `P3` possa ricevere messaggi contenenti le stesse informazioni ricevute tramite le chiamate della entry. Poiché la entry `E1` non contiene nessun parametro di modo `in`, cioè passato per valore a `P3`, la porta `E1` del processo `P3` sarà di tipo `signal`.
Con queste indicazioni, il codice del processo `P1` diventa:

```
process P1 {
      port int risultato;
      int b;
      signal s;
      process proc;
      ……
      send (s) to P3.E1;
      proc receive(b) from risultato;
      ……
}
```


Il processo `P2` chiama la entry `E2` di `P3`. In `P3` la dichiarazione di tale entry prevede il parametro `x` di tipo `int` e modo `in`. Quindi nessun valore viene restituito al processo chiamante. Nonostante ciò, in `P2` dovremo dichiarare una porta (`ak`) di tipo `signal` su cui il processo si pone in attesa di un messaggio dopo aver inviato la richiesta alla porta `E2` di `P3` in modo tale da forzare la sincronizzazione tipica del *rendez-vous*. In assenza di ciò, una volta inviato il messaggio alla porta `E2` di `P3`, il processo `P2` proseguirebbe senza sospendersi essendo asincrone le primitive di comunicazione. Poiché la entry `E2` ha un parametro `int` di modo `in`, la porta `E2` viene dichiarata in `P3` di tipo `int`.
Con queste indicazioni, il codice del processo `P2` diventa:

```
process  P2 {
      port signal ak;
      int a;
      signal s;
      process proc;
      ……
      send (a) to P3.E2;
      proc=receive(s) from ak;
      ……
}
```

Nel processo `P3` vengono dichiarate le porte `E1` e `E2` come sopra indicato. Lo statement `accept E1(int out y)` viene tradotto mediante la ricezione di un segnale dalla porta `E1` e, alla fine del codice relativo al corpo di tale `accept`, viene restituito il valore `y` al processo chiamante (`P1`) tramite la sua porta `risultato`. Nel corpo di questo `accept` è contenuto, in forma nidificata, l'`accept` di `E2` che viene tradotto mediante la ricezione dell'interto `x` dalla porta `E2` e, dopo l'istruzione `y=f(x)` che ne costituisce il corspo, viene restituito un segnale tramite la porta `ak` per risvegliare il processo chiamante.
Con queste indicazioni, il codice del processo `P3` diventa:

```
process P3 {
      port signal  E1;
      port int E2;
      signal s;
      process proc1, proc2;
      int x,y;
      ……..
      proc1=receive(s) from E1;
            proc2=receive(x) from E2;
            y=f(x);
            send (s) to proc2.ak;
      send (y) to proc1.risultato;
      ……..
}
```