Alla ricerca dell’invariante perduto

La ricetta d’indicizzazione

Abbiamo int C[6][5] e vogliamo riempirlo di valori. Li leggiamo dallo stream “input” (vedi Sezione 3.1.1 del testo):

PRE=(input contiene almeno 30 interi)

POST=($\forall$a $\in$[0..5], $\forall$b $\in$[0..4], C[a][b] è definito)

R1=($\forall$a $\in$**[0..i-1]**, $\forall$b $\in$[0..4], C[a][b] è definito, 0<=i<=6)

For(int i=0; i<6; i++) \\R1

{

Leggi la riga i-esima

{

POST2=(ho letto la riga i-esima)

$=(\forall b \in [0..4],\ C[i][b]$ è definita$)$

R2=$(\forall b \in [0..j-1],\ C[i][b]$ è definita$,\ 0<=j<=5)$

```
for(int i=0; i<6; i++) //R1
{
 for(int j=0; j<5; j++) //R2
     input>>C[i][j];
//POST2
}
//POST
```

Dati 2 insiemi A e B, B è contenuto in A se
$a \in B \Rightarrow a \in A$

Un multi-insieme A contiene elementi con
numerosità: A(b) = n. di copie di b in A

B è m-contenuto in A se $B(a)>0 \Rightarrow A(a) >= B(a)$

C[6][5] ➔ bool B[6][5]

POST1=($\forall$a$\in$[0..5], $\forall$b$\in$[0..4], B[a][b] $\Leftrightarrow$ C[a] è contenuta in C[][b])

Usiamo di nuovo la ricetta di indicizzazione

R1= ($\forall$a$\in$[0..i-1], $\forall$b$\in$[0..4], B[a][b] $\Leftrightarrow$ C[a] è contenuta in C[][b])

R2=($\forall$b$\in$[0..j-1], B[i][b] $\Leftrightarrow$ C[i] è contenuta in C[][b])

```
for(int i=0; i<6; i++) //R1
{
  for(int j=0; j<5;j++) //R2
```

//POST2 =(∀b∈[0..4], B[i][b] ⇔ C[i] è contenuta in C[][b])

```
}
```

//POST1=(∀a∈[0..5], ∀b∈[0..4], B[a][b] ⇔ C[a] è contenuta in C[][b])

calcola B[i][j]

Significa determinare se ogni elemento di C[i] è in C[][j]

Servono ancora due cicli: il primo per scandire gli elementi di C[i] ed il secondo per cercarlo nella colonna C[][j]

POST3=(OK $\Leftrightarrow \forall b \in [0..4], \exists\, a \in [0..5], C[i][b]=C[a][j]$)

R3=(OK $\Leftrightarrow \forall b \in [0..k-1], \exists\, a \in [0..5], C[i][b]=C[a][j]$)

il quarto ciclo deve determinare se C[i][k] è presente in C[][j]

POST4=(trovato $\Leftrightarrow \exists$ a $\in$[0..5], C[i][k]=C[a][j])

da cui con la solita ricetta:

R4=(trovato $\Leftrightarrow \exists$ a $\in$[0..z-1], C[i][k]=C[a][j], 0<=z<=5)

NOTA che potremmo essere più precisi sul valore di a, ma non ci interessa

```
OK=true;
for(int k=0; k<5;k++) // R3
{
   bool trovato=false;
    for(int z=0; z<6; z++)  //R4
        if(C[i][k]==C[z][j])
              trovato=true;
 //POST4 ci dice che trovato va bene
    if(!trovato)
        OK=false;
}    //POST3  OK da la risposta giusta
B[i][j]=OK;
```

fa cose inutili

basta un solo booleano

```
OK=true;
for(int k=0; k<5 && OK;k++) // R3
{
   bool trovato=false;
    for(int z=0; z<6 && !trovato; z++)  //R4
        if(C[i][k]==C[z][j])
              trovato=true;
 //POST4 ci dice che trovato va bene
    if(!trovato)
        OK=false;
}    //POST3  OK da la risposta giusta
B[i][j]=OK;
```

```
OK=true;
for(int k=0; k<5 && OK;k++) // R3
{
   OK=false;
   for(int z=0; z<6 && !OK; z++)  // R4
      if(C[i][k]==C[z][j])
            OK=true;
 // POST4
}   //POST3  OK da la risposta giusta
B[i][j]=OK;
```

i nuovi POST4 e R4 sono facili:

POST4=(OK $\Leftrightarrow \exists$ a $\in$[0..5], C[i][k]=C[a][j])

da cui con la solita ricetta:

R4=(OK $\Leftrightarrow \exists$ a $\in$[0..z-1], C[i][k]=C[a][j], 0<=z<=5)

nuova condizione d’uscita

Esercizio 6.2 del testo

Si tratta di riempire B, come prima, ma le righe e le colonne vanno considerate come multi-insiemi e quindi la relazione da considerare è m-contenimento.

Ovviamente i cicli 1 e 2 non cambiano. E’ necessario cambiare solo i cicli 3 e 4.

POST3=(OK ⇔ C[i][0..4] è m-contenuta in C[][j])
facile applicare ricetta di indicizzazione
R3? =(OK ⇔ C[i][0..k-1] è m-contenuta in C[][j])
ma serve ? non tanto

R3? non ci dice come testare l’m-contenimento

ogni volta che troviamo C[i][k] in C[a][j], dobbiamo ricordarci che C[a][j] non dobbiamo più usarlo per altri elementi di C[i]

basta introdurre un array bool T, lungo come una colonna, cioè 6, inizialmente (che significa?) tutto false e, nel caso precedent,e T[a]=true

test di m-appartenenza diventa:

```
if(C[i][k]==C[z][j]  && !T[z])
{trovato=true; T[z]=true;}
```

R3? =(OK $\Leftrightarrow$ C[i][0..k-1] è m-contenuta in C[][j])
ma serve ? non tanto

OK => $\exists$[a0..a(k-1)] a due a due diversi e in [0..5] t.c. (C[i][0]=C[a_0][j], ...C[i][k-1]=C[a(k-1)][j]) && (T[a0]=true,...T[a(k-1)]=true) &&(gli altri elementi di T sono false)

La ricetta di indicizzazione serve anche per l’esercizio 2 di mercoledì scorso

Esercizio 2: data char B[5][10] calcolare l’indice minimo di una **colonna** di B che contiene lo stesso numero di ‘a’ e di ‘b’. Se non c’è alcuna colonna che soddisfi questa condizione, allora l’indice deve avere valore -1

PRE=(B[5][10] definita)

POST=(OK => (0<=indice<=9 && !SI(B[][0..indice-1]) &&
SI(B[][indice]))
!OK=>(indice=-1 && !SI(B[][0..9]))
)

i-1 → 9

(OK => (0<=indice<=i-1 && !SI(B[][0..indice-1]) &&
SI(B[][indice]))
!OK=>(indice=-1 && !SI(B[][0..i-1]))
)

indice=i-1

Verifica della condizione d'uscita:
R=(OK => (indice=i-1 && !SI(B[][0..indice-1]) && SI(B[][indice])),
!OK=>(indice=-1 && !SI(B[][0..i-1])), 0<=i<=10)

R &&!(i<10 && !OK) => (i=10 || OK)

a) Se OK allora R garantisce che indice è la minima colonna che soddisfa la condizione, come richiede la POST.

b) Se i=10 && !OK=> R implica che indice=-1 e che !SI(B[][0..9]), come richiesto dalla POST è verificata altra POST, altro invariante

PRE=(B[5][10] definita)

POST=(indice !=-1 => (0<=indice<=9 && !SI(B[][0..indice-1])
&& SI(B[][indice])
indice=-1 => !SI(B[][0..9])
)

i-1 → 9

R=(indice !=-1 => (0<=indice<=i-1 && !SI(B[][0..indice-1])
&& SI(B[][indice]))
indice=-1 => !SI(B[][0..i-1])
0<=i<=10)

indice =i-1