Verifica – parte IIB

Rif. Ghezzi et al.
6.3.4.1

università di ferrara
DA SEICENTO ANNI GUARDIAMO AVANTI.

- Test
  - In piccolo
    - White box
    - Black box
    - Condizioni di confine
    - Problema dell' Oracolo
  - In grande
    - Test di modulo
    - Test di integrazione
    - Test di Sistema
    - Test di Accettazione

università di ferrara
DA SEICENTO ANNI GUARDIAMO AVANTI.

Criteri di selezione per test strutturali (white box)

- Copertura delle istruzioni
- Copertura delle decisioni (o degli archi)
- Copertura delle condizioni
- Copertura dei cammini

università di ferrara
DA SEICENTO ANNI GUARDIAMO AVANTI.

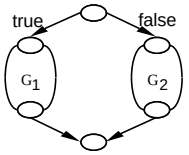
Grafo di controllo

- Costruito ricorsivamente
- Ogni istruzione elementare è rappresentata da un nodo
- Siano S_1 e S_2 due istruzioni, e G_1 e G_2 i corrispondenti grafi. Allora:

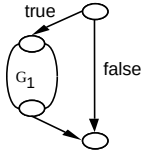
università di ferrara
DA SEICENTO ANNI GUARDIAMO AVANTI.

Grafo di controllo

if cond then
    s_1;
else
    s_2;
end if

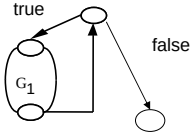


if cond then
    s_1;
end if

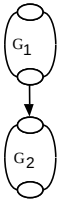


Grafo di controllo

while cond loop
    s_1;
end loop

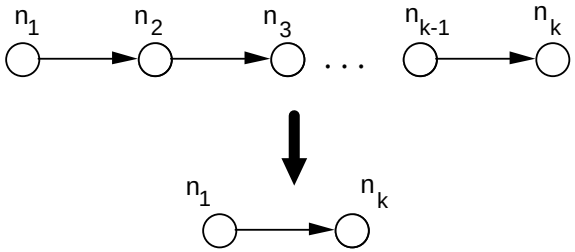


s_1;
s_2;



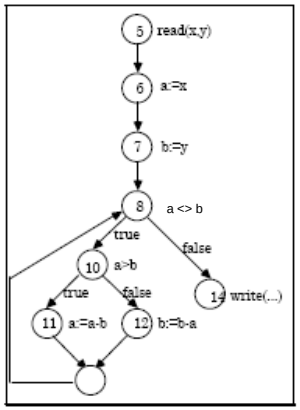
Semplificazione

- Una sequenza di archi si può collassare in un solo arco:



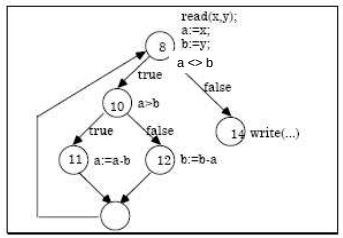
Esempio: algoritmo di Euclide

```
1    program gcd (input, output);
2        var
3            x, y, a, b: integer;
4        begin
5            read (x, y);
6            a := x;
7            b := y;
8            while a <> b do
9                begin
10                   if a > b
11                           then a := a - b;
12                           else b := b - a
13               end;
14       write ('MDC dei dati e''', a)
15       end.
```

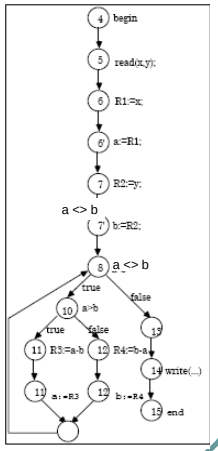


Esempio: algoritmo di Euclide

Sequenza collassata



Espansione dell'assegnamento (R1, ..., R4 registri)



Criterio di copertura delle istruzioni

- Scegliere i casi di test in modo da eseguire, complessivamente, tutte le istruzioni elementari del programma almeno per un caso di test.
- Solitamente si fa riferimento alla grammatica del programma e si considera istruzione tutto ciò che può essere derivato dal costrutto <statement>: ad esempio, nei linguaggi imperativi, assegnamenti, operazioni di I/O e chiamate di subroutine.
- Misura di copertura C_0 = <numero di istruzioni eseguite> / <numero di istruzioni eseguibili>

Esempio

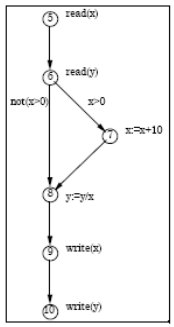
```
begin
   read(x); read(y);
   while x != y loop
    if x > y then
         x:= x – y;
    else
         y:= y – x;
    end if;
   end loop;
   gcd := x;
end;
```

Si ottiene copertura delle istruzioni (C_0=1) con il test set {(3,3), (4,3), (3,4)}

Esempio

Nel grafo di controllo, ogni nodo deve essere percorso almeno per un caso di test.

```
1  Program statement (input, output);
2      var
3          x,y : real;
4      begin
5          read(x);
6          read(y);
7          if x > 0 then x:=x+10;
8          y:=y/x;
9          write(x);
10         write(y);
11     end.
```



Un test con x>0 e y qualsiasi (ad esempio {(20,30)}) garantisce la copertura delle istruzioni, ma non rileva l'errore che si ha per x = 0 alla riga 8.

Esempio: minimalità

```
read (x); read (y);
if x > 0 then
    write ("1");
else
    write ("2");
end if;
if y > 0 then
    write ("3");
else
    write ("4");
end if;
```

{<x = 2, y = 3>, <x = - 13, y = 51>, <x = 97, y = 17>, <x = - 1, y = - 1>} copre tutte le istruzioni

{<x = - 13, y = 51>, <x = 2, y = - 3>} è minimo

Debolezza del criterio

Frammento di programma che calcola
Il valore assoluto di un numero

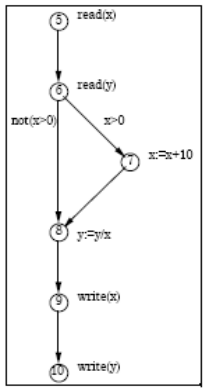
```
if x < 0 then
    x := -x;
end if;
z := x;
```

{x=-3} copre le istruzioni, ma non considera il caso di x>0 in cui il ramo then non viene eseguito (e che causa un fallimento)

Copertura degli archi (o delle decisioni)

- Scegliere il test set T in modo che ogni arco del grafo di controllo del programma sia percorso almeno per un d in T.
- Più fine del criterio di copertura delle istruzioni
- Misura di copertura C_1 = <numero di archi percorsi> / <numero di archi percorribili>

Esempio



- Il test set visto prima non garantisce la copertura degli archi.
- Il test set {(20,30),(0,30)} garantisce la copertura degli archi e rileva l'errore nella riga 8 (divisione per 0).

Debolezza del criterio

```
found := false; counter := 1;
while (not found) and counter < number_of_items loop
      if table (counter) = desired_element then
            found := true;
      end if;
      counter := counter + 1;
end loop;
if found then
      write ("the desired element is in the table");
else
      write ("the desired element is not in the table");
end if;
```

Il test set: (1) tabella vuota, (2) tabella con 3 elementi, di cui il secondo è l'elemento ricercato copre gli archi ma non rileva l'errore (< invece di ≤).

Costituenti

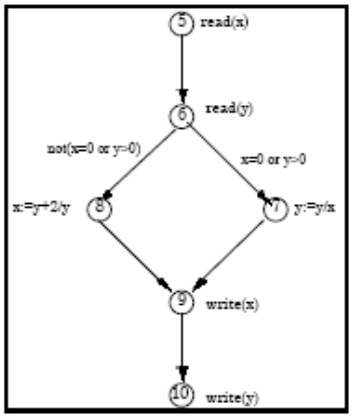
- Generalmente, le condizioni logiche sono composte da formule atomiche (espressioni relazionali o variabili booleane) e loro negazioni, composte tramite congiunzioni e disgiunzioni
- Ogni formula atomica o sua negazione è detta costituente della condizione.

Copertura delle condizioni

- Scegliere un test set T tale che, eseguendo P per ogni elemento di T, tutti i possibili valori dei costituenti delle condizioni composte risultano esercitati almeno una volta.
- Criterio di copertura degli archi e delle condizioni: in aggiunta, ogni arco del flusso di controllo di P risulta percorso. Ovviamente più fine del criterio di copertura degli archi.

Esempio

```
1  Program branch (input, output);
2  var
3      x,y:real;
4  begin
5      read(x);
6      read(y);
7      if (x = 0 or y > 0) then  y:=y/x
8      else    x:=y+2/y;
9      write(x);
10     write(y);
11 end.
```

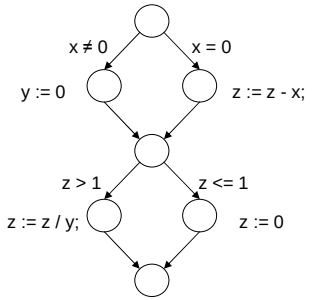


Esempio

- Il test set B = {(5,5),(5,-5)} copre le decisioni, ma non rileva la possibile divisione per 0 alle righe 7 e 8.
- Il test set C = {(0,-5),(5,5)} soddisfa la copertura delle condizioni e rileva la possibile divisione per 0 alla riga 7, ma non quella alla riga 8 (non copre le decisioni).

Debolezza del criterio

```
if x ≠ 0 then
    y := 0;
else
    z := z - x;
end if;
if z > 1 then
    z := z / y;
else
    z := 0;
end if;
```

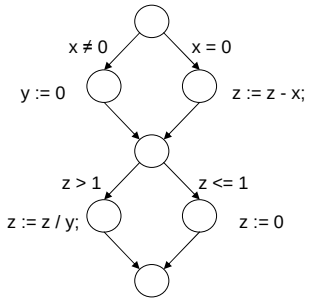


{<x = 0, z = 1>, <x = 1, z = 3>}
copre gli archi e le condizioni,
ma non rileva la possibilità di una divisione per 0

Copertura dei cammini

- Scegliere un insieme di test T tale che l'esecuzione di P per ogni d di T provochi l'esecuzione di tutti i cammini che collegano il nodo iniziale a quello finale nel grafo di controllo di P.
- Più fine dei criteri precedenti
- Il numero di cammini può essere elevato (o infinito) anche per programmi semplici

Esempio



Il test set {(0,1),(0,2),(1,1),(1,2)} copre i cammini (e rileva la divisione per 0).

Copertura dei cammini

- Il numero di cammini può essere elevato (o infinito in caso di cicli) anche per programmi semplici.
- In presenza di istruzioni condizionali (if o case) occorre cercare di combinare il più possibile i cammini (e in maniera oculata)

Copertura dei cicli

- Linee guida in presenza di cicli: scegliere il test set in modo che il ciclo venga eseguito nelle seguenti modalità:
  - Il numero minimo di volte compatibile con la struttura di controllo
  - Il massimo numero di volte (se esiste)
  - Un numero di volte intermedio

Non fattibilità

- Una volta scelto un criterio, bisogna individuare gli input che lo soddisfano
- Esempio:

```
read(x);
z:=2*x;
If z=4 then
statement
```

- Per eseguire statement occorre dare 2 come input

Non fattibilità

```
if x>0 then
   if x<0 then
   statement
```

- In generale non è possibile garantire la copertura completa (statement non è raggiungibile)
- In generale non è possibile determinare algoritmicamente la raggiungibilità delle istruzioni