Verifica – parte IIIA

Rif. Ghezzi et al.

6.4 - 6.4.1 *(+parte analisi dataflow)*

Sommario

- **Analisi statica**
  - Informale
    - Code walk-through
    - Ispezioni di codice
  - Formale
    - Analisi di flusso delle variabili (data-flow)
    - *Prove formali di correttezza (es. Logica di Hoare)*
- **Test**: criteri di selezione DataFlow
- **Esecuzione simbolica**

Analisi

- Tecnica statica (non richiede esecuzione del programma)
- Vantaggio: il risultato vale non per una singola esecuzione (come il test), ma per una *classe* di esecuzioni.
- Svantaggi:
  - spesso soggetta a fallibilità umana
  - opera su ***un modello del sistema*** anziché sul sistema stesso

Analisi informale

- Analisi di
  - specifiche dei requisiti
  - specifica di progetto
  - codice

  e simulazione manuale del comportamento
- Non dovrebbe eseguirla l'autore dell'artefatto.

Code walk-through

- Analisi del codice eseguita in collaborazione da diverse persone.
- Simulazione su carta del comportamento del programma in alcuni casi di test.
- Linee guida:
  - gruppi ristretti (da 2 a 5 persone)
  - i partecipanti dovrebbero ricevere la documentazione alcuni giorni prima

Code walk-through

- l'incontro dovrebbe durare un tempo prefissato
- la discussione ha il solo scopo di trovare gli errori, non di eliminarli
- partecipanti chiave:
  - progettista, che presenta il suo lavoro
  - moderatore
  - segretario (verbalizzatore)
- assenza dei manager

Ispezioni del codice

- Organizzazione simile a code walk-through, ma:
  - lettura del codice piuttosto che simulazione
  - ricerca di errori appartenenti a classi prefissate
- Errori comuni:
  - variabili non inizializzate
  - salti all'interno di cicli
  - assegnamenti incompatibili

Ispezioni del codice

- cicli che non terminano
- indici di array fuori campo
- errori su allocazione dinamica di memoria
- mancata corrispondenza fra parametri formali e attuali
- confronti di uguaglianza fra valori float

- La frequenza e le modalità delle sessioni dipendono dal prodotto e dal processo.

Analisi formale

- Le tecniche utilizzate nei compilatori …
  - Analisi lessicale
  - Analisi sintattica
  - Type checking

Il numero di anomalie rilevate staticamente dipende dalle caratteristiche di dinamicità del linguaggio (C, C++ *vs* Java … per late binding ad esempio)

- Analisi *data-flow*

Analisi di flusso dei dati

- L'evoluzione del valore delle variabili durante l'esecuzione del programma è difficile da analizzare staticamente.
- Si possono però considerare le operazioni eseguite sulle variabili, rappresentare le sequenze possibili e controllare che rispettino determinate regole

Operazioni su variabili

- Ogni comando che coinvolge variabili esegue una o più delle seguenti operazioni:
  - **Definizione** (d): assegna alla variabile un nuovo valore
  - **Uso** (u): richiede il valore della variabile
  - **Annullamento** (a): al termine dell'esecuzione del comando, il valore della variabile non è più definito

Esempio

```
1   procedure swap (x1, x2:real)
2   var x: real;
3   begin
4   x2:= x;
5   x2:= x1;
6   x1:= x;
7   end;
```

Per la variabile x:

- annullamento (2)
- uso (4)
- uso (6)

- (stringa auu)
- Per x1:dud
- Per x2:ddd

Esempio

- L'esame delle sequenze può rilevare anomalie
  - La sequenza auu per x mostra che il valore di x è usato senza essere definito.
  - La sequenza ddd per x2 è sintomo di una possibile anomalia (perchè definire una variabile e non usarla?)
- Errore: inversione di x e x2 alla riga 4.

Regole

- R1: in ogni sequenza, l'uso di una variabile deve essere preceduto dalla definizione, senza annullamenti intermedi
- R2: in ogni sequenza, la definizione di una variabile deve essere seguita da un uso, prima di annullamenti o ulteriori definizioni

Esempi

- Sequenze legali:
  - aduduu
  - duadudu
- Sequenze non legali
  - adu**dd**u
  - d**au**duu
  - du**au**du

Analisi di flusso

- Non è limitata alle sole variabili
- Applicabile a tutti i casi in cui l'esecuzione di un programma comporta operazioni la cui sequenza è rilevante.
- Es: file
  - apertura (a)
  - chiusura (c)
  - lettura (l)
  - scrittura (s)

Regole per file

- Supponendo accesso in sola lettura o sola scrittura:
  - l e s devono essere precedute da a, senza c intermedie
  - due a devono essere separate da una c
  - una l(s) non può essere seguita da una s(l) senza una c e una a intermedie
  - una c deve essere preceduta da una a senza c intermedie

Decisioni e cicli

- Gli esempi visti finora hanno una sola sequenza, finita, di operazioni.
- In generale:
  - le decisioni possono portare a sequenze alternative
  - i cicli a sequenze infinite
- Occorre un linguaggio in grado di esprimere questi casi

Espressioni regolari

- Estende il concetto di sequenza aggiungendo all'operatore di composizione sequenziale gli operatori di alternativa e di iterazione.
- Simboli:
  - · (sequenza, spesso omesso)
  - + (alternativa)
  - * (iterazione)

Sintassi

- Dato un alfabeto (insieme finito di simboli) A,
  - ε (stringa nulla) è un'espressione regolare
  - ogni simbolo di A è un'espressione regolare
  - se e_1 ed e_2 sono espressioni regolari,
    - $e_1 \cdot e_2$ (o $e_1 e_2$) è un'espressione regolare
    - $e_1 + e_2$ è un'espressione regolare
    - e_1^* è un'espressione regolare
  - nient'altro è un'espressione regolare

Esempi

- Alfabeto {a,u,d}
- Definizione di insiemi di stringhe:
  - a+u $\rightarrow$ {a,u}
  - a* $\rightarrow$ {ε, a, aa, ...}
  - (a+u)* $\rightarrow$ {ε, a, u, aa, uu, ...}
  - (ad+ ε)* $\rightarrow$ {ε, ad, adad, adadad, ...}
  - d(a*+u)* $\rightarrow$ {d, da, du, duu, daaaa, …}

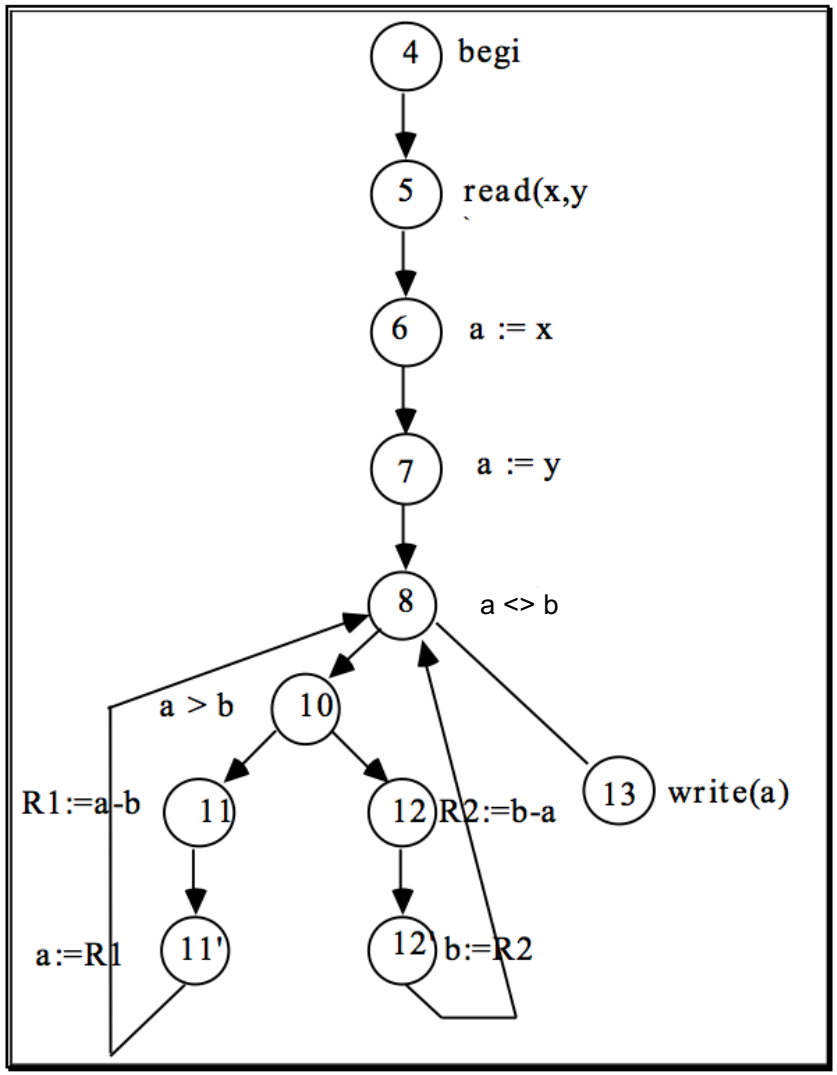
Esempio

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

Esempio

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

Grafo di controllo



Operazioni su variabili

nodo	variabili definite	variabili usate	variabili annullate
4			x, y, a, b
5	x, y		
6	a	x	
7	a	y	
8		a, b	
10		a, b	
11		a, b	
11'	a		
12		a, b	
12'	b		
13		a	

Analisi di flusso

- Un’espressione regolare rappresenta un insieme (anche infinito) di stringhe
- Per ogni tipo di analisi, definire l’alfabeto
  - per le variabili, {d, u, a}
  - per i file, {a, c, l, s}
- Analisi del programma e definizione delle possibili sequenze per mezzo di espressioni regolari

Costruzione di espressioni regolari per una variabile

- L'espressione regolare relativa a un nodo è il simbolo corrispondente all'operazione (ε indica nessuna operazione)
- Se e1 ed e2 sono le espressioni regolari per x relative ai grafi G1 e G2
  - l'espressione relativa alla sequenza G1 G2 è e1 e2
  - l'espressione relativa all'alternativa fra G1 e G2 è e1+e2
  - l'espressione relativa al ciclo che ripete G1 è e1*

Costruzione di espressioni regolari

- Se *e, e1* ed *e2* sono le espressioni relative a cond, G1 e G2,
- l'espressione relativa a

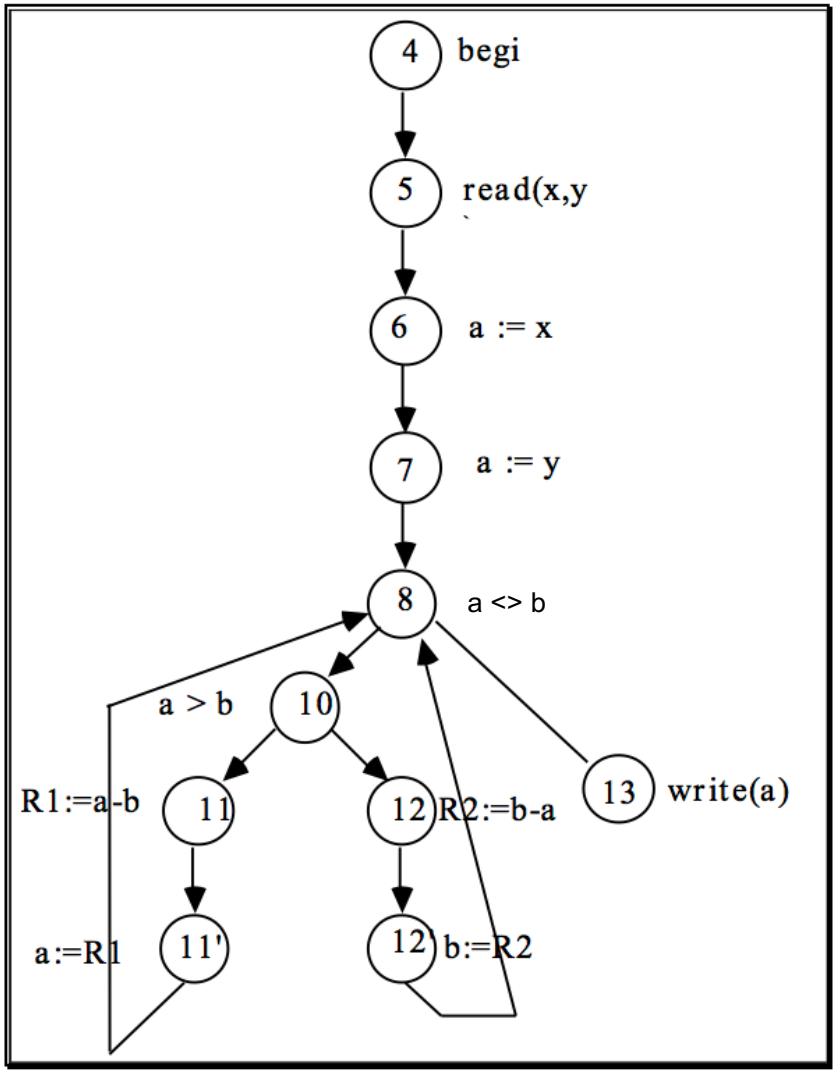
  if (cond) **then** G1 **else** G2

  è *e(e1+e2)*
- l'espressione relativa a

  while (cond) G1

  è *e(e1 e)**

Grafo di controllo



$P([\rightarrow 8];a) = add$

$P([8\rightarrow];a) = (u(ud+u)u)^*u$

$P([4\rightarrow];a) = addu(u(ud+u)u)^*u$

Esempio

- Espressione regolare per la variabile a, per tutti i cammini uscenti dal nodo 8, nodo 8 escluso:

  $P([8\rightarrow];a) = (u(ud+u)u)^*u$

- Espressione regolare per la variabile a, per tutti i cammini entranti nel nodo 8, nodo 8 escluso:

  $P([\rightarrow 8];a) = add$

Esempio

- Espressione regolare per la variabile a per l'intero grafo:

  P([4→];a) = a**dd**u(u(ud+u)u)*u
- Esame della consistenza delle espressioni regolari
- La prima definizione non è seguita da un uso.

Criterio di n-copertura dei cicli

- Soddisfatto da un test set T se e solo se ogni cammino contenente un numero di iterazioni di ogni ciclo non superiore a n è eseguito per almeno un dato di test in T.
- Il ***valore appropriato di n*** discende da considerazioni sul codice e, in particolare, sulle operazioni sulle variabili.

Esempio: gcd

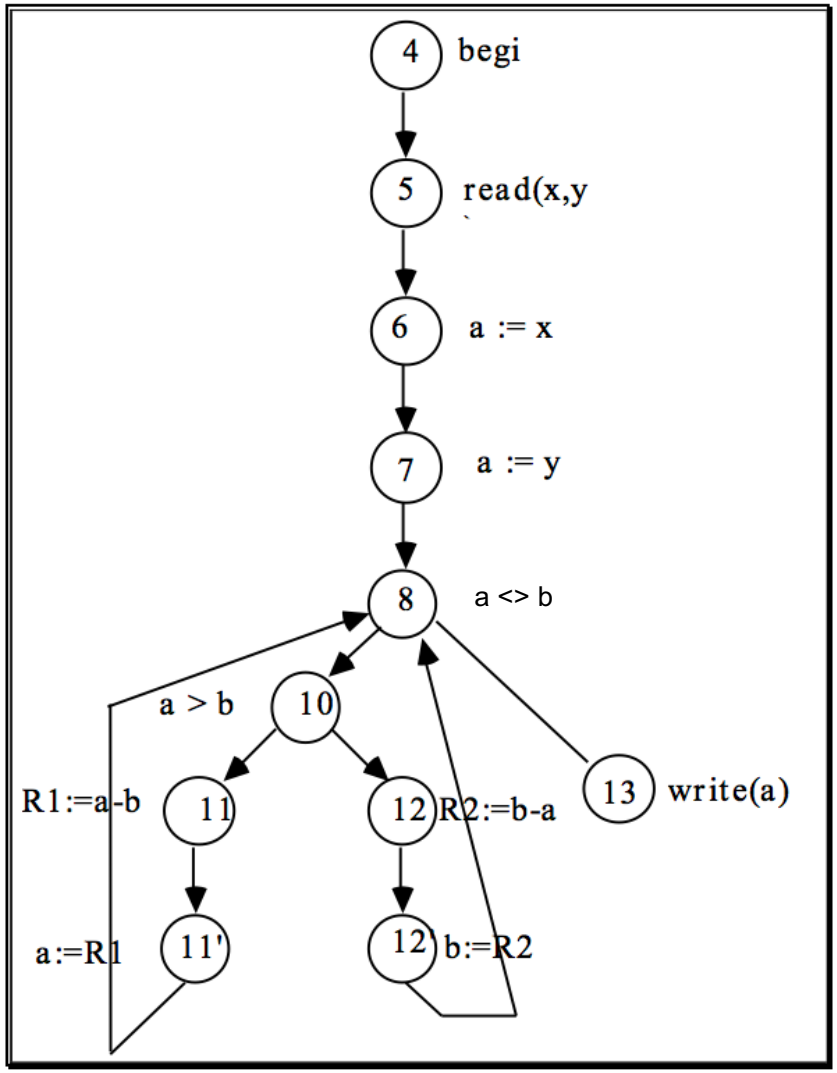
```
1  program gcd (input, output);
2      var
3          x, y, a, b: integer;
4      begin
5          read (x,y);
6          a := x;
7          a := y;
8          while a <> b do
9              begin
10                 if a > b
11                         then a := a - b;
12                         else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

- Nell'if all'interno del ciclo while si possono usare valori delle variabili definiti in precedenti iterazioni.
- La rilevazione di eventuali errori in queste sequenze richiede l'esecuzione di ***almeno 2 iterazioni***.

Test: criteri di selezione data flow

- Determinano la significatività dei cammini in un programma a partire dall'analisi di flusso delle variabili
- Selezione di un cammino basata sul numero di sequenze definizione/uso delle variabili
- Un ***ciclo*** si esegue un ***numero maggiore di volte*** solo se ciò permette di eseguire nuove sequenze definizione/uso.

Esempio: gcd



- Per x e y: sufficiente la 0-copertura dei cicli
- Per a e b:
  - La 2-copertura dei cicli esegue tutte le sequenze definizione-uso
  - Una n-copertura dei cicli, con n>2, non esegue nuove sequenze definizione-uso

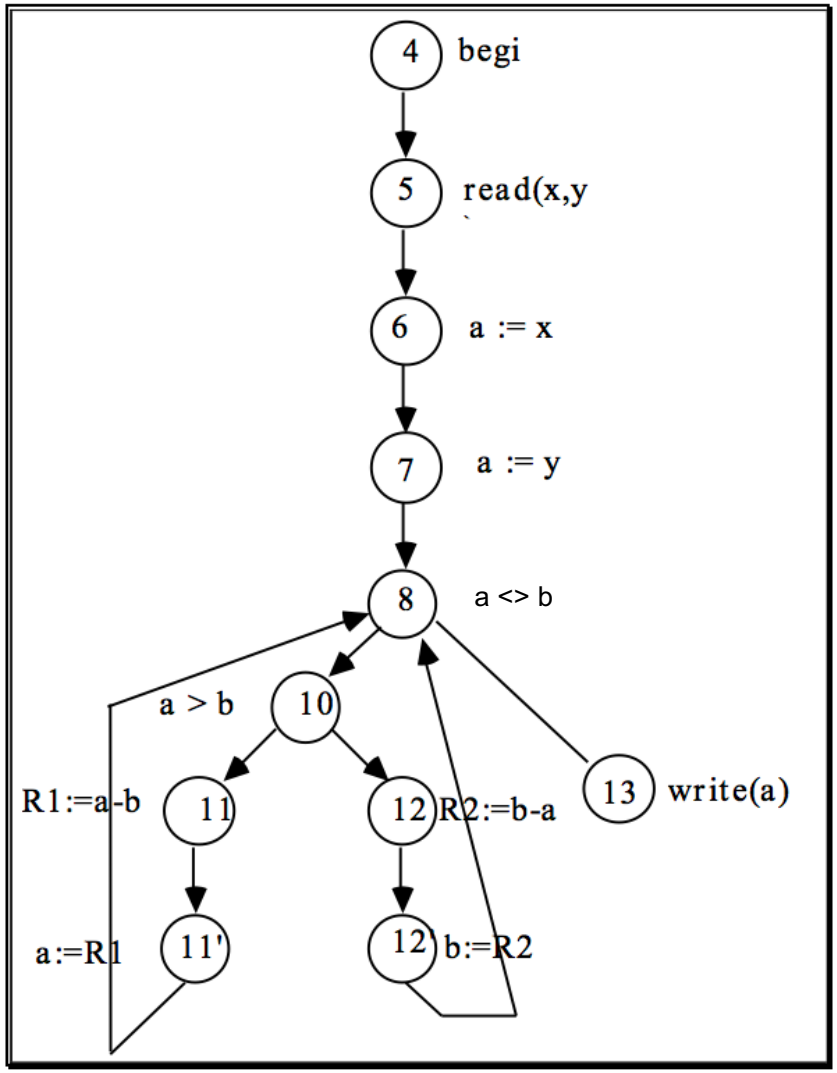
Criteri di selezione data-flow

- Considerano solo definizioni e usi (gli errori dovuti ad annullamenti si possono eliminare con l'analisi statica).
- A ogni nodo del grafo di controllo si associano:
  - L'insieme ***def*** delle variabili definite dal comando corrispondente
  - L'insieme ***use*** delle variabili usate
- Costruzione del grafo in modo da avere ***def*** e ***use*** disgiunti per ogni nodo (registri)

Definizioni

- Sia x una variabile di un programma P.
- Il cammino *(i, n_1, .., n_m, j)* nel grafo corrispondente al programma P è un *cammino libero da definizioni rispetto alla variabile x dal nodo i al nodo j* se e solo se la variabile *x* non appartiene a nessuno degli insiemi *def* associati ai nodi n_1, .., n_m, *j*.

Esempio: gcd

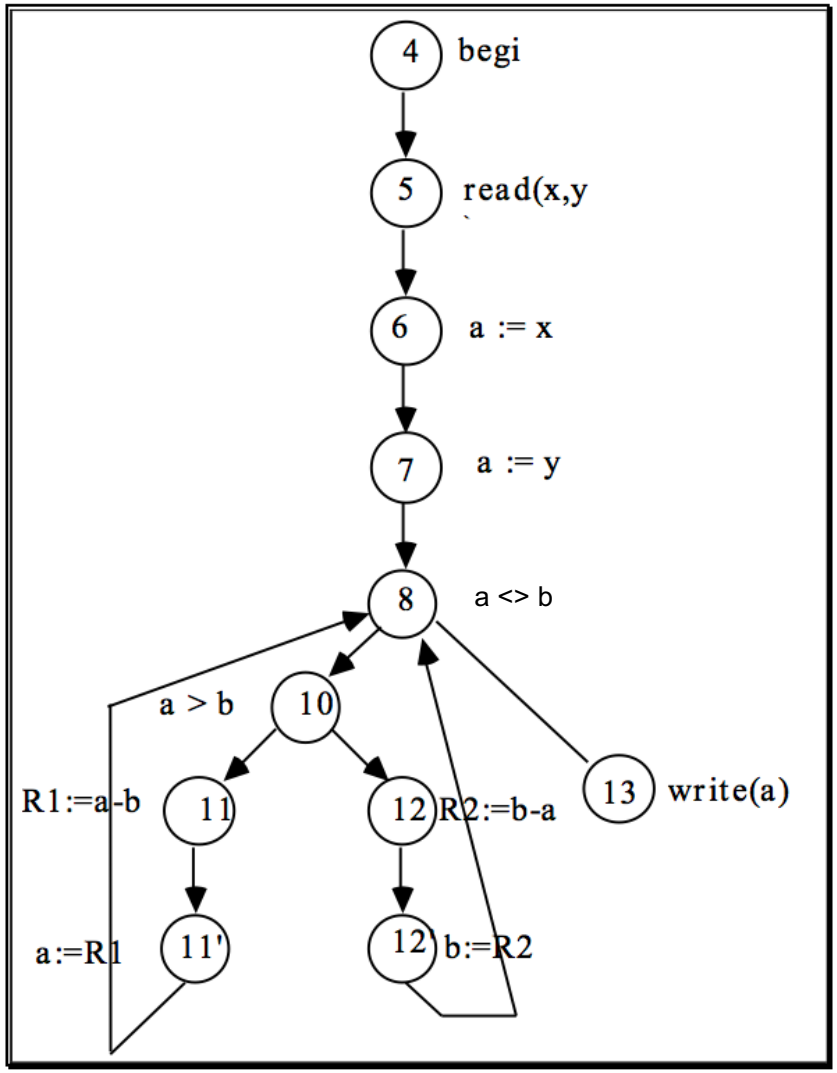


- Il cammino <6,7,8,10,12,12’,8,10> è libero da definizioni rispetto alla variabile a.
- Il cammino <6,7,8,10,11,11’,8> non lo è.

Definizioni

- Per ogni nodo *i* e ogni variabile *x* appartenente all'insieme *def* associato al nodo *i*, si definisce ***insieme du(x, i)*** come l'insieme di tutti i nodi *j* tali che esiste un cammino libero da definizioni rispetto alla variabile *x* dal nodo *i* al nodo *j* e *x* è usata nel nodo *j.*
- *In pratica, è l'insieme dei nodi raggiungibili da i che usano la definizione di x in i*

Esempio: gcd

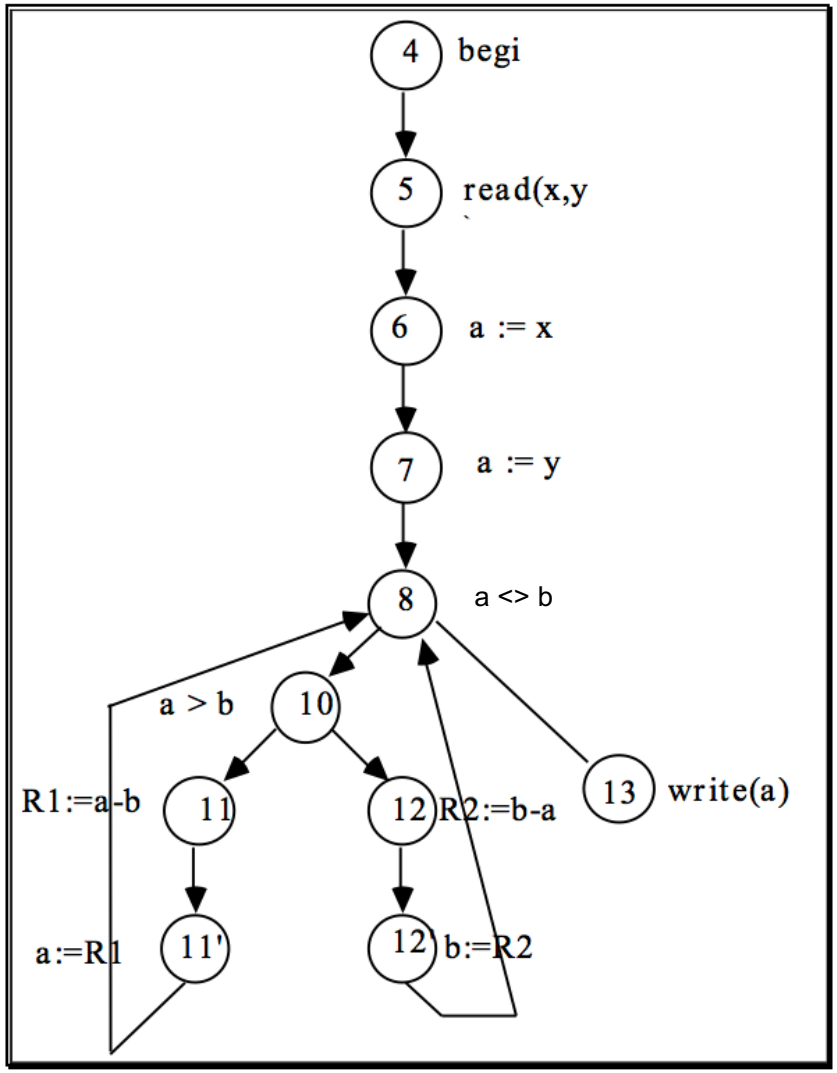


- du(a,6) = {8, 10, 11, 12, 14}
- du(a,11') = {8, 11, 12, 14}
- du(x,5) = {6}

Criterio di copertura delle definizioni

- Soddisfatto da un test set T per un programma P se e solo se per ogni nodo *i* e per ogni variabile *x* appartenente all'insieme *def(i)* ***esistono un nodo u*** di *du(i, x)* ed un elemento d di T tali che l'esecuzione di P(d) percorre un cammino libero da definizioni rispetto a x dal nodo i al nodo u.

Esempio: gcd

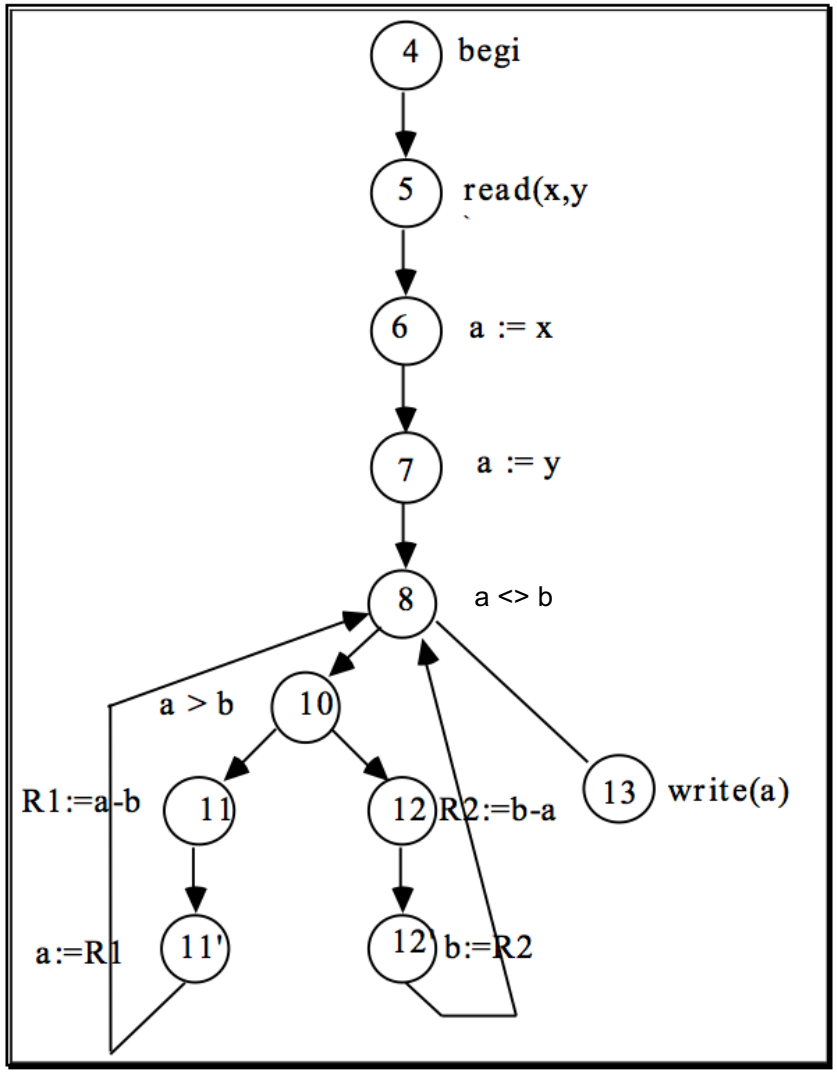


- Il criterio è soddisfatto da test set che comportano al massimo una iterazione del ciclo while.
- Es. non richiede che la def. di a in 11' sia seguita dall'uso in 11.
- Pertanto non rileva eventuali errori in questa sequenza

Criterio di copertura di tutti gli usi

- Soddisfatto da un test set T per un programma P se e solo se per ogni nodo *i* del grafo di controllo di P, per ogni variabile *x* appartenente all'insieme *def(i)* e ***per ogni nodo u*** appartenente a *du(i,x)*, esiste un elemento d di T tale che l'esecuzione di P(d) percorre un cammino libero da definizioni rispetto a x dal nodo i al nodo u.

Esempio: gcd



- Il criterio seleziona test set adeguati.
- Es: la def. di a in 11’ deve essere seguita dagli usi in 8,10, 11, 12, 14.

Criterio di copertura dei cammini du

- Soddisfatto da un test set T per un programma P se e solo se per ogni nodo *i* del grafo di controllo di P, ogni variabile *x* appartenente all'insieme *def(i)*, ***ogni cammino libero da definizioni*** rispetto a x da *i* a ogni elemento di *du(i,x)* è percorso nell'esecuzione di P(d) per almeno un d di T.