Verifica – parte IIIC

Rif. Ghezzi et al.

6.5

Esecuzione simbolica

- Approccio intermedio fra analisi e test
- Consiste nell'esecuzione del programma con valori simbolici anziché numerici.
- Un'esecuzione simbolica corrisponde all'esecuzione di molti test.
- I risultati possono essere usati per dimostrare proprietà del programma.

Esempio

```
read(a); read(b);
x:=a+1;
x:=x+b+2;
write(x);
[A+B+3]
```

```
a=A; b=B;
x=A+1;
x=A+1+B+2=A+B+3;
output=A+B+3
```

- Notazione:
  - Convenzionalmente, il valore simbolico letto per una variabile è indicato dal suo identificatore in maiuscolo (il valore letto per x è X)
  - Fra parentesi quadre si indica il valore dell'output

Esempio: asserzione

{true}
read(a);
x:=a*a;
x:=x+1;
write(x);
{output>0}

a=A;
$x=A^2$;
$x=A^2+1$;
output=A^2+1

Sicuramente positiva

Esempio

```
read(a)
if a>0 then
   <ramo true>;
else
   <ramo false>;
end if;
```

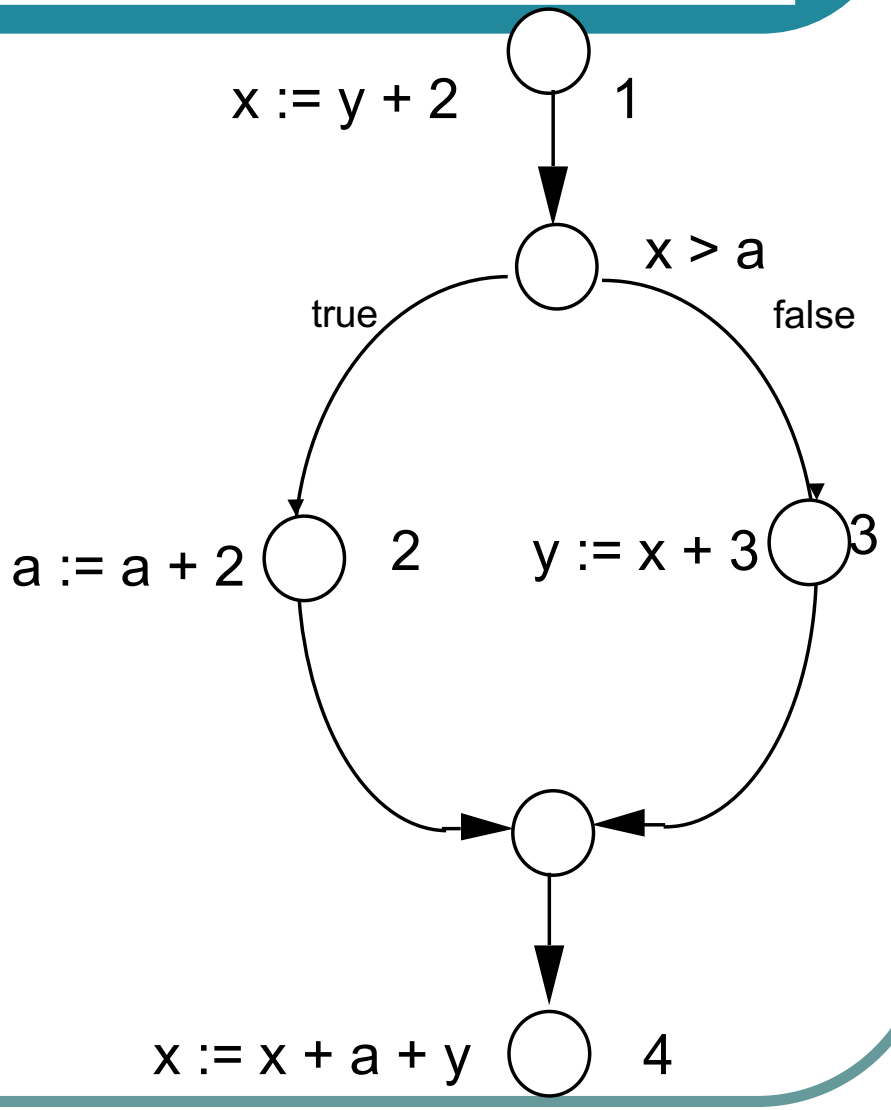
a=A

?...

Il valore A non contiene informazione sufficienti per scegliere il cammino

Esempio

```
x := y + 2;
if x > a then
   a := a + 2;
else
   y := x + 3;
end if;
x := x + a + y;
```



Esempio

- La scelta di un cammino fra quelli possibili è determinata da una *path condition*.
- Esempio: per il ramo else, path condition è $Y + 2 \leq A$.
- Risultato dell'esecuzione:

$<\{a = A, y = Y + 5, x = 2 * Y + A + 7\},$

$<1, 3, 4>,$ $Y + 2 \leq A>$

cammino path condition

Esecuzione condizionata

- In generale, se P è un frammento di programma e G_P è il suo grafo di controllo, l'esecuzione simbolica di P su un cammino è il suo *stato simbolico*, dato da:
  - ***symbolic_variable_values*** (equazioni del tipo identifier=symbolic_expression)
  - ***execution_path*** (sequenza dei nodi visitati)
  - ***path_condition*** (espressione logica sui valori simbolici delle variabili che porta all'esecuzione del cammino)

Regole per l'esecuzione simbolica

- Interprete: entità che svolge l'esecuzione simbolica aggiornando lo stato simbolico
- Stato iniziale:
  - valori simbolici non definiti
  - cammino d'esecuzione nullo
  - path condition true
- read(x):
  - rimuove ogni legame per x
  - aggiunge il legame x=X

Regole per l'esecuzione simbolica

- write(espressione):
  - n è un valore inizializzato a 1 e incrementato a ogni write
  - output(n) = valore simbolico di <espressione>
- assegnamento x := espressione:
  - crea un'espressione simbolica SV utilizzando i valori simbolici delle variabili in <espressione>
  - aggiunge il legame x = SV

Regole per l'esecuzione simbolica

- Dopo l'esecuzione dell'ultima istruzione di una sequenza corrispondente a un nodo N di G_p, il nome del nodo si aggiunge all'execution path
- if cond then s1 else s2 endif:
  - si valuta cond: eval(cond) valore simbolico
  - se eval(cond) ha un valore di verità già definito, l'esecuzione procede nel ramo corrispondente
  - altrimenti si sceglie arbitrariamente un valore:
    - true: eval(cond) si aggiunge in and alla path condition e l'esecuzione procede nel ramo true;
    - false: not eval(cond) si aggiunge in and alla path condition e l'esecuzione procede nel ramo false

Regole per l'esecuzione simbolica

- while cond loop S end loop:
  - si valuta cond: eval(cond) valore simbolico
  - se eval(cond) ha un valore di verità già definito, l'esecuzione procede nel ramo corrispondente (S o uscita dal ciclo)
  - altrimenti si sceglie arbitrariamente un valore:
    - true: eval(cond) si aggiunge in and alla path condition e si esegue S;
    - false: not eval(cond) si aggiunge in and alla path condition e l'esecuzione procede all'uscita dal ciclo
- Gli execution path e le corrispondenti path condition possono essere infiniti.

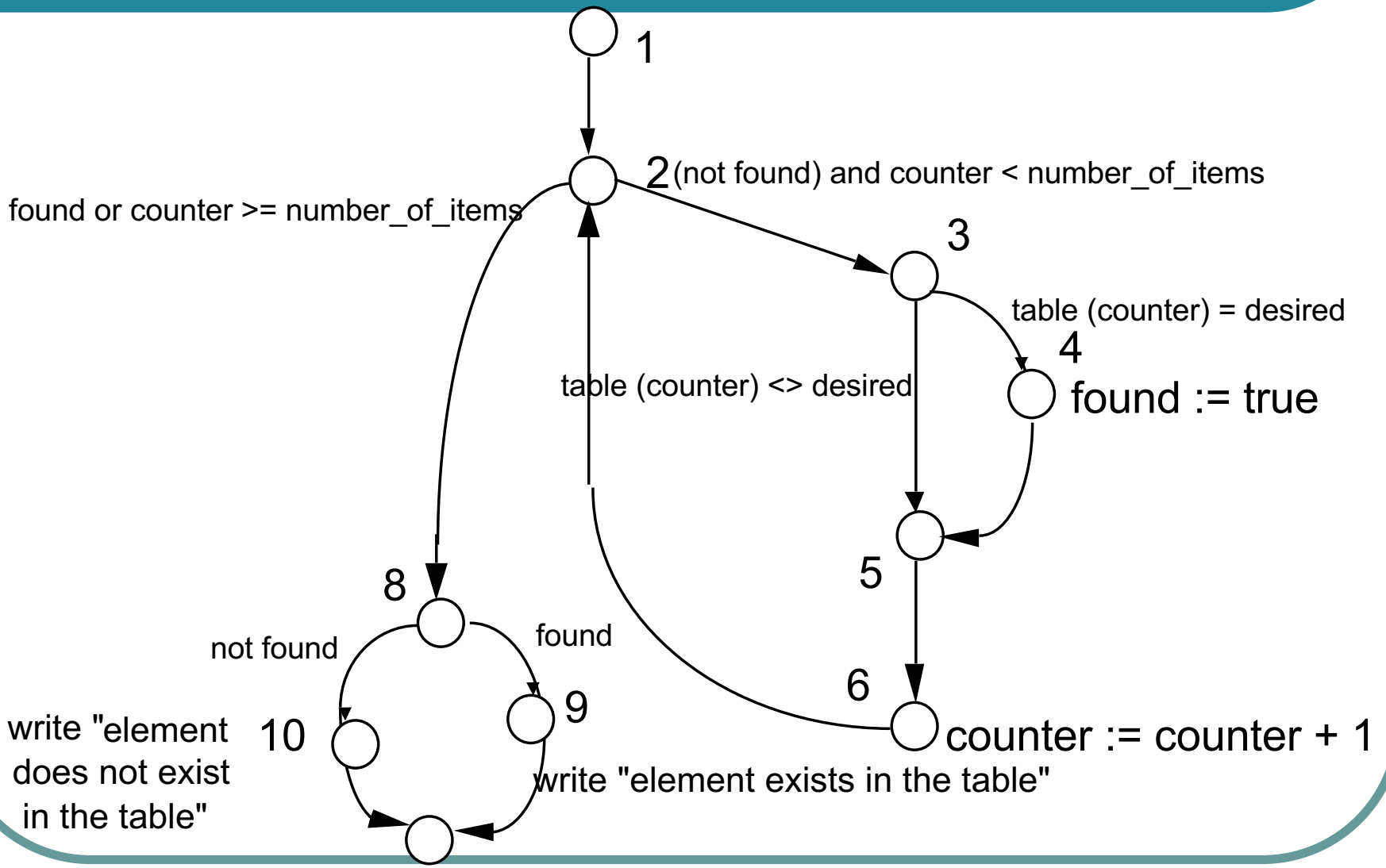
Esecuzione simbolica e test

- L'esecuzione simbolica può essere utilizzata nella selezione dei casi di test
- Una volta fissato un cammino da eseguire (per rispettare un determinato criterio) si può
  - individuare algoritmicamente la path condition necessaria
  - cercare di soddisfarla con i dati di input (problema ***indecidibile*** in generale)

Esempio

```
1. found := false; counter := 1;
2. while (not found) and counter < number_of_items loop
3.  if table (counter) = desired_element then
4.      found := true;
5.  end if;
6.  counter := counter + 1;
7. end loop;
8. if found then
9.  write ("element exists in the table");
10. else write ("element does not exist in the table");
12. end if;
```

Grafo di controllo



Esempio

- Cammino <1,2,3,4,5,6,2,8,9>

Esecuzione	**Symbolic state**
Inizio	Found=?, counter=? , PC= true
1	Found = false, counter = 1, PC = true
2 (not found) and counter < number_of_items? (true)	Found = false, counter = 1, PC = (1<number_of_items)
3 table (counter) = desired_element ? (scelta: true)	Found = false, counter = 1, PC = (table(1) = desired_element and 1<number_of_items)
4	Found = true, counter = 1, PC = (table(1) = desired_element and 1<number_of_items)
5,6	Found = true, counter = 2, PC = (table(1) = desired_element and 1<number_of_items)
2 found or counter >= number_of_items? (true) 8 found? (true), 9	Found = true, counter = 2, PC = (table(1) = desired_element and 1<number_of_items), output(1) = "the desired element exists in the table"