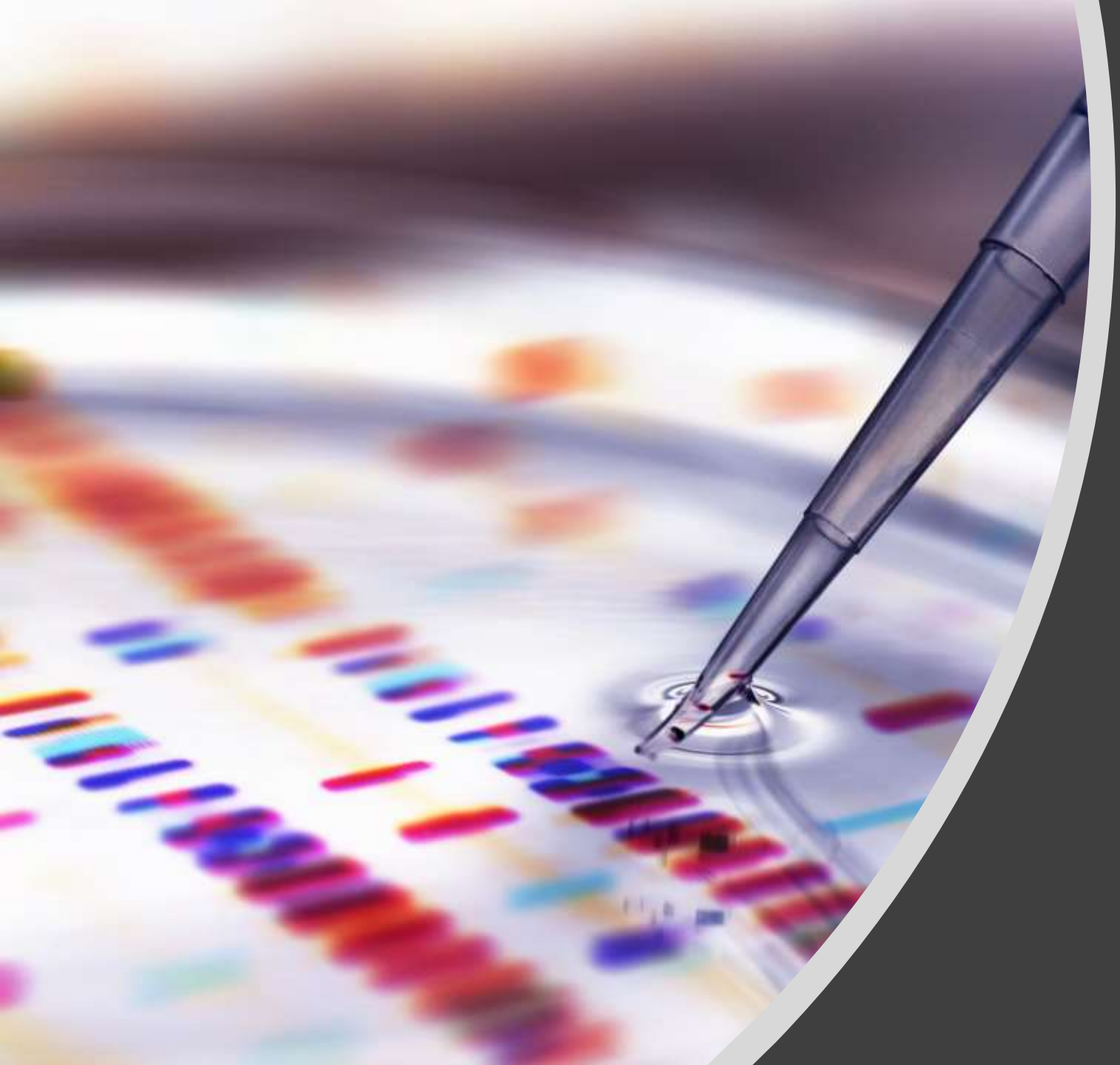


Data Flow Testing

Universidad Autónoma de Coahuila

Facultad de Sistemas

Carlos Nassif Trejo García

Idea general

Revisar el uso de apuntadores

Instanciar dos veces

Motivaciones

Accesibilidad deseable de una variable (lectura / escritura)

Verificar la exactitud de un valor generado por una variables

Estático

Data Flow anomaly: Defectos potenciales de un programa

Dinámico

- Identificar paths de un programa desde el código fuente basado en una clase de criterios de data Flow testing

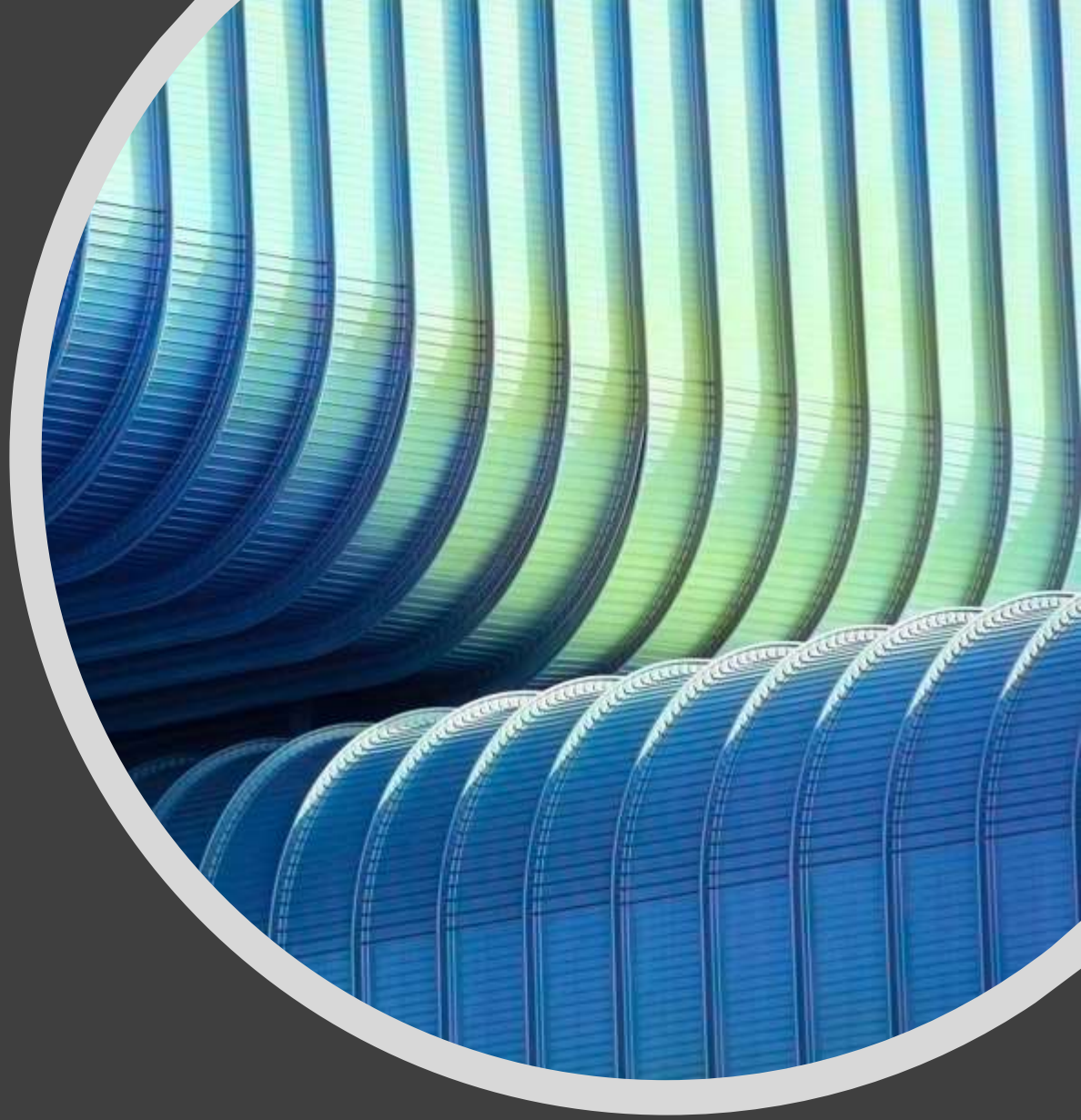
Control Flow Testing V. Data Flow Testing

Similitud

- Generar paths
- Generar casos de prueba a partir de esos paths

Diferencia

- CFT: Criterios son usados en una etapa previa
- DFT: Criterios son usados después



Anomalía

- Desviación o forma anormal de hacer algo
- Asignar dos valores a una variable sin usar la primera asignación
- Usar una variable sin asignarla primero
- Generar un valor y nunca utilizarlo

Tipo 1: Definido y después definido otra vez

- Primera computación es redundante
- Primera computación tiene un error
- Segunda computación tiene un error
- Computación faltante entre los dos

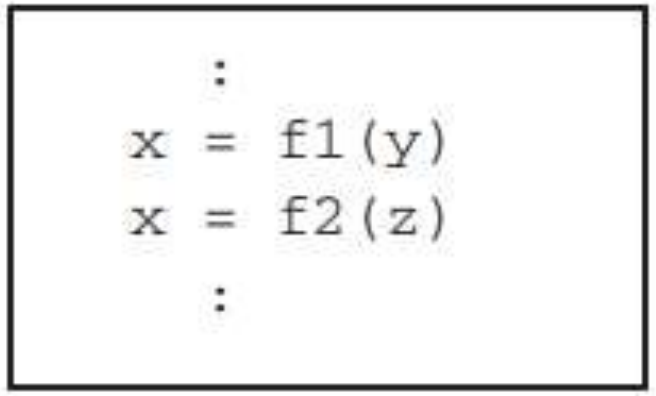


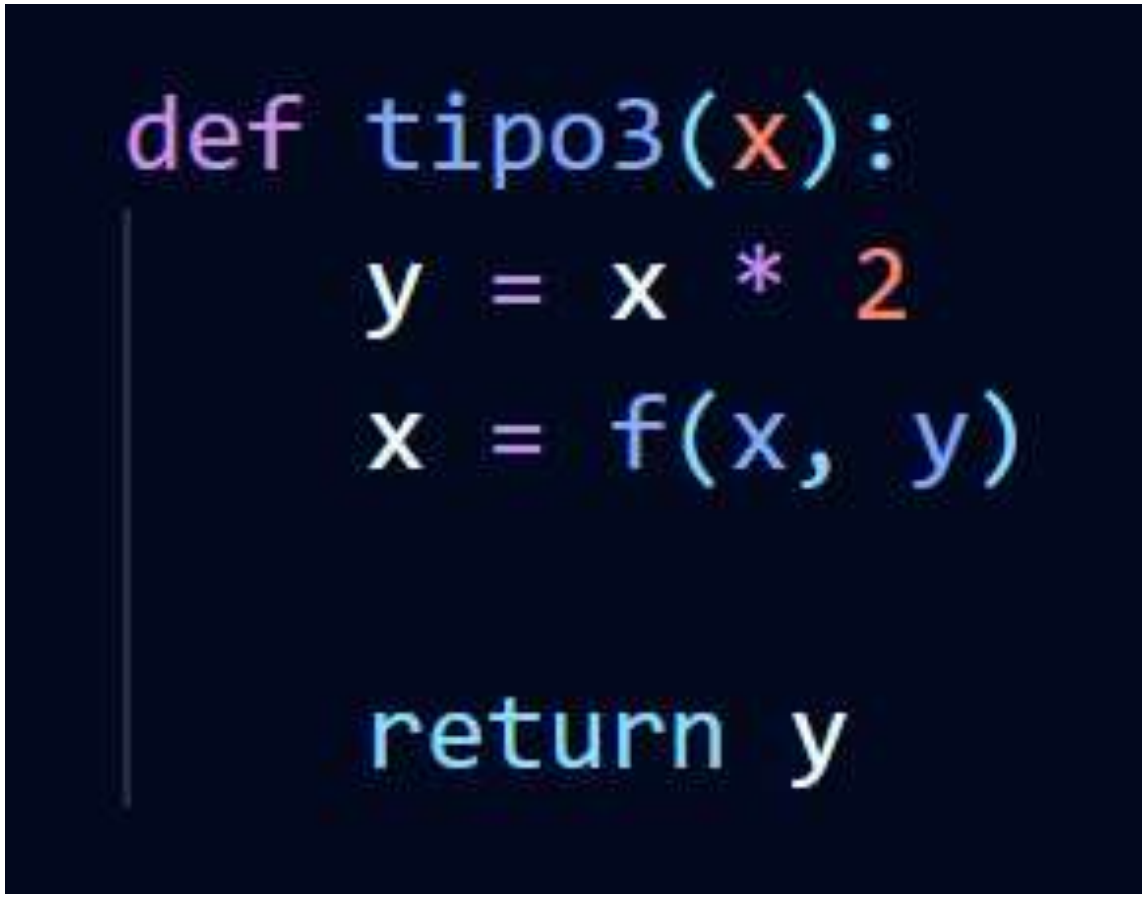
Figure 5.1 Sequence of computations showing data flow anomaly.

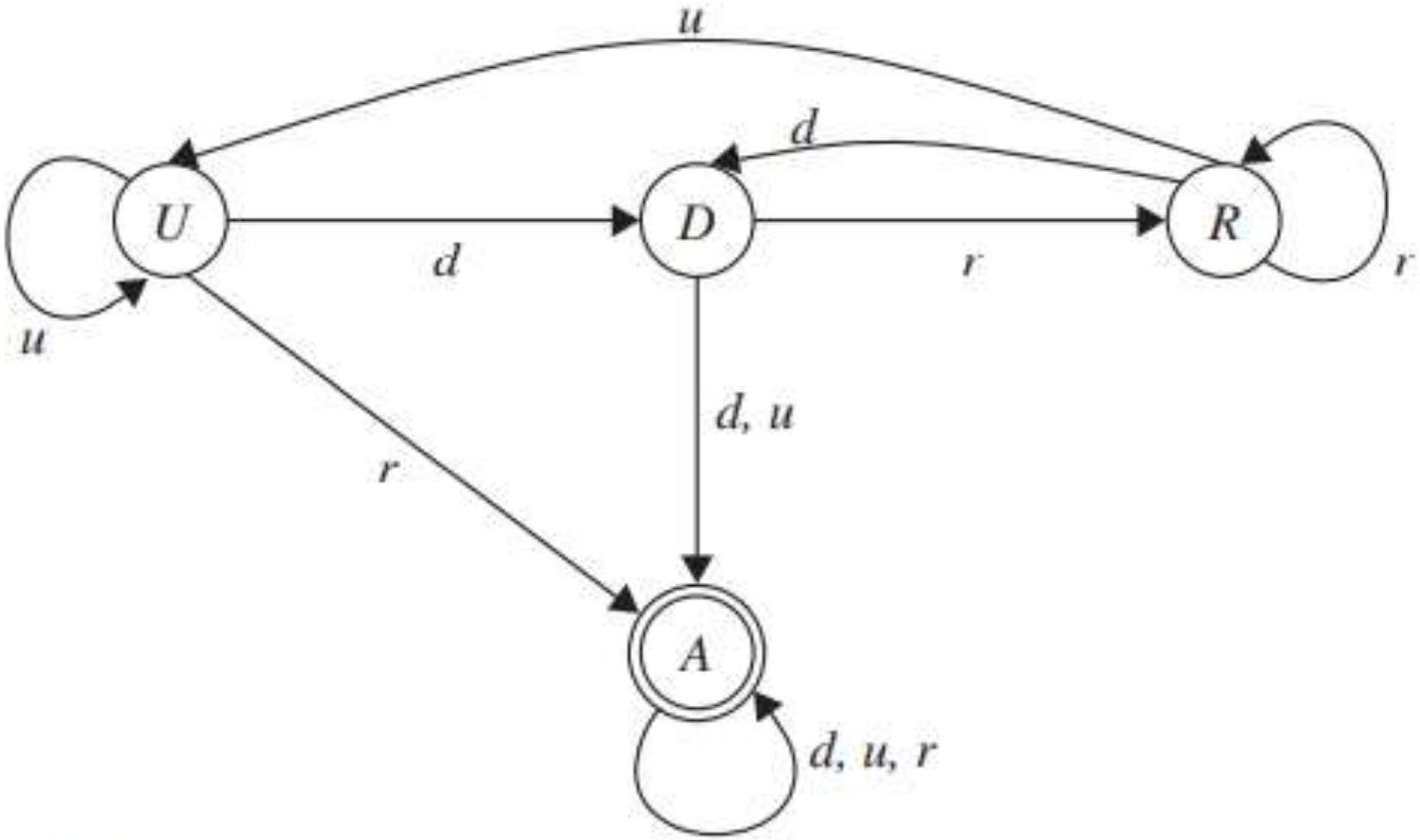
Tipo 2: Indefinido pero referenciado

```
def tipo2(x, y, z):
    x = x + t + z
    return x
```

Tipo 3: Definido pero no referenciado

- Definir una variable
- Indefinirla sin usarla





Legend:

States

U: Undefined

D: Defined but not referenced

R: Defined and referenced

A: Abnormal

Actions

d: Define

r: Reference

u: Undefine

Dinámico

- No podemos estar seguro que una variable fue asignado con el valor correcto si no hay casos de prueba que causan la ejecución de un path desde la asignación hasta la referencia de ella.

Program Path

- Seleccionar paths de entrada – salida con el objetivo de cubrir ciertas definiciones de datos y patrones de uso (data Flow testing criteria)

1. Dibujar un grafo de data Flow
2. Seleccionar 1 o más criterios de data Flow testing
3. Identificar paths en el grafo que satisface los criterios
4. Derivar expresiones de predicados
5. Resolver las expresiones para obtener datos de entrada

Data Flow Graph

- Definición: Un valor es movido a una ubicación de memoria de una variable
- Indefinido o muerto: El valor y la ubicación se vuelven indeterminados
- Usado: El valor es obtenido desde la ubicación de memoria
  - Computacional: Un nuevo valor es producido
  - Predicado: Controla el Flow de la ejecución

```
int VarTypes(int x, int y){
      int i;
      int  *iptr;
      i   = x;
      iptr  = malloc(sizeof(int));
      *iptr = i + x;
      if (*iptr > y)
          return (x);
      else {
          iptr  = malloc(sizeof(int));
          *iptr = x + y;
          return(*iptr);
          }
}
```

Figure 5.3 Definition and uses of variables.

Directed graph

- Secuencia de definiciones y c-uses con cada nodo del grafo
- Conjunto de p-uses asociados a cada arista
- El nodo de entrada tiene una definición de cada parámetro y cada variable global
- El nodo de salida produce indefiniciones de cada variable local

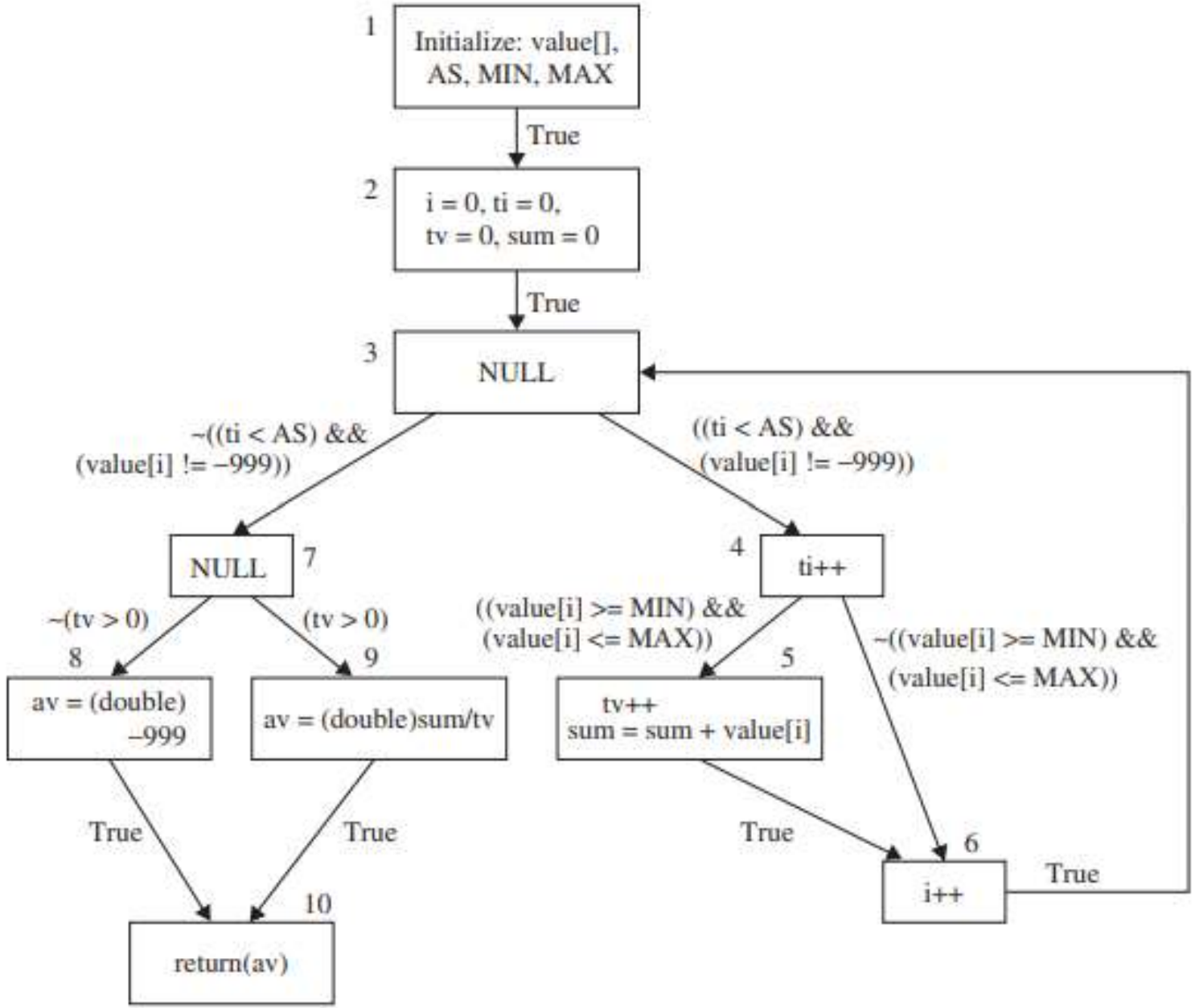


Figure 5.4 Data flow graph of ReturnAverage() example.

Términos del Data Flow

- Encontrar paths que contengan pares de definiciones y de uso de variables
- Global c-use: Un c-uso de una variable x en un nodo i, es denominado global c-use si x fue definido en un nodo anterior otro que el nodo i.
- Definition Use Path: Path donde nodo inicial defines una variable x, y el nodo final usas la variable X
- Definition Clear Path: Path donde el nodo inicial es el único que define una variable x
- Definición Global: Un nodo i tiene una definición global de una variable x si dicho nodo define x y existe un def-clear path con respecto al nodo i hacia algún nodo / arista (global c-use / p-use) que use x.

TABLE 5.1 Def() and c-use() Sets of Nodes in Figure 5.4

Nodes i	def(i)	c-use(i)
1	{value, AS, MIN, MAX}	{}
2	{i, ti, tv, sum}	{}
3	{}	{}
4	{ti}	{ti}
5	{tv, sum}	{tv, i, sum, value}
6	{i}	{i}
7	{}	{}
8	{av}	{}
9	{av}	{sum, tv}
10	{}	{av}

TABLE 5.2 Predicates and p-use() Set of Edges in Figure 5.4

Edges (i, j)	predicate(i, j)	p-use(i, j)
(1, 2)	True	{}
(2, 3)	True	{}
(3, 4)	(ti < AS) && (value[i] ! = −999)	{i, ti, AS, value}
(4, 5)	(value[i] < = MIN) && (value[i] > = MAX)	{i, MIN, MAX, value}
(4, 6)	¯((value[i] < = MIN) && (value[i] > = MAX))	{i, MIN, MAX, value}
(5, 6)	True	{}
(6, 3)	True	{}
(3, 7)	¯((ti < AS) && (value[i] ! = −999))	{i, ti, AS, value}
(7, 8)	¯(tv > 0)	{tv}
(7, 9)	(tv > 0)	{tv}
(8, 10)	True	{}
(9, 10)	True	{}

Tipos de paths

- Simple path: Path en donde todos los nodos (excepto el inicial y el final), son distintos
- Loop-Free Path: Todos los nodos son distintos
- Complete path: Path desde la entrada hasta la salida

Data Flow Testing Criteria

- All Du-Path
- All-Uses
- All-C-Uses / Some-P-Uses
- All-P-Uses / Some-C-Uses
  - All-Defs
  - All-P-Uses

All-Defs

- Para cada variable x y para cada nodo i en donde exista una definición global de x, selecciona un path completo donde incluya una def-clear path del nodo i hacia
  - Nodo j que contenga un c-use global de x
  - Arista (j,k) teniendo un p-use de x

tv

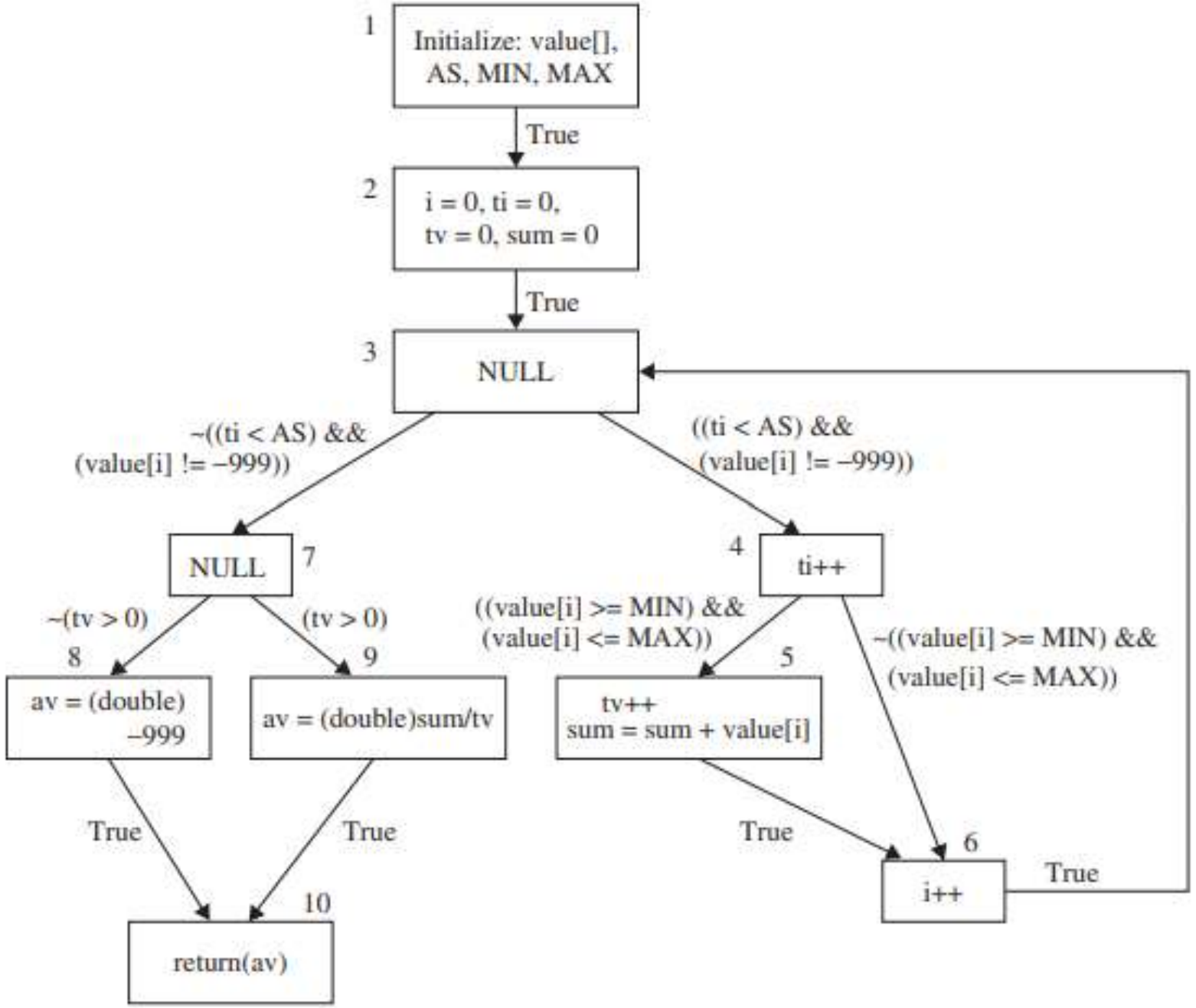


Figure 5.4 Data flow graph of ReturnAverage() example.

All C-Uses

- Para cada variable x y para cada nodo i, dado que i contenga una definición global de x, selecciona path completos que incluya def-clear paths desde el nodo i hata todos los nodos j dado que exista un uso global de x en j.

ti

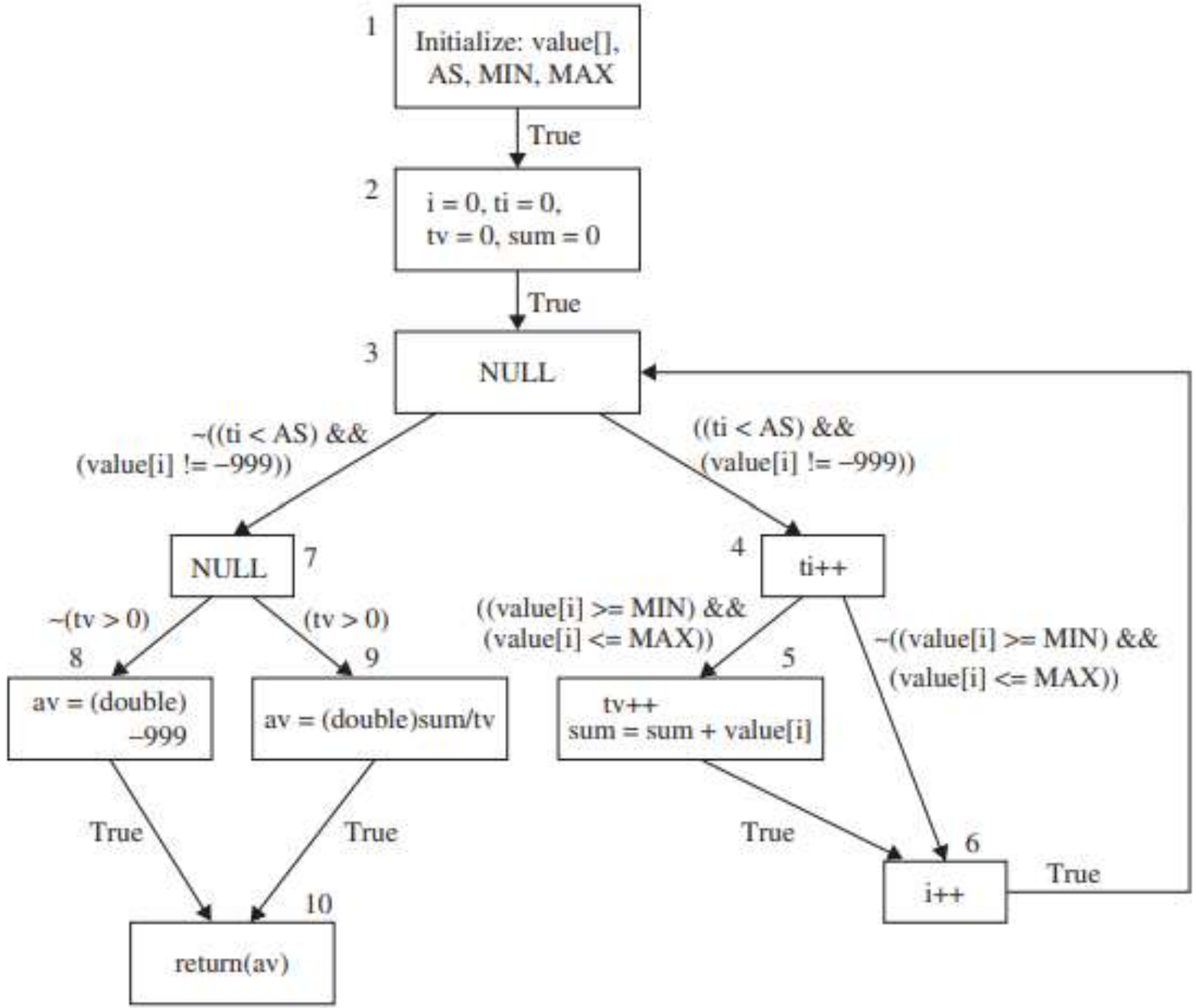


Figure 5.4 Data flow graph of ReturnAverage() example.

All p-uses

- Para cada variable x y para cada nodo i dado que x contenga una definición global en i, selecciona paths completos que incluya def-clear paths desde el nodo i hacia las aristsas (j,k) dado que exista un p-use en dicha arista.

tv

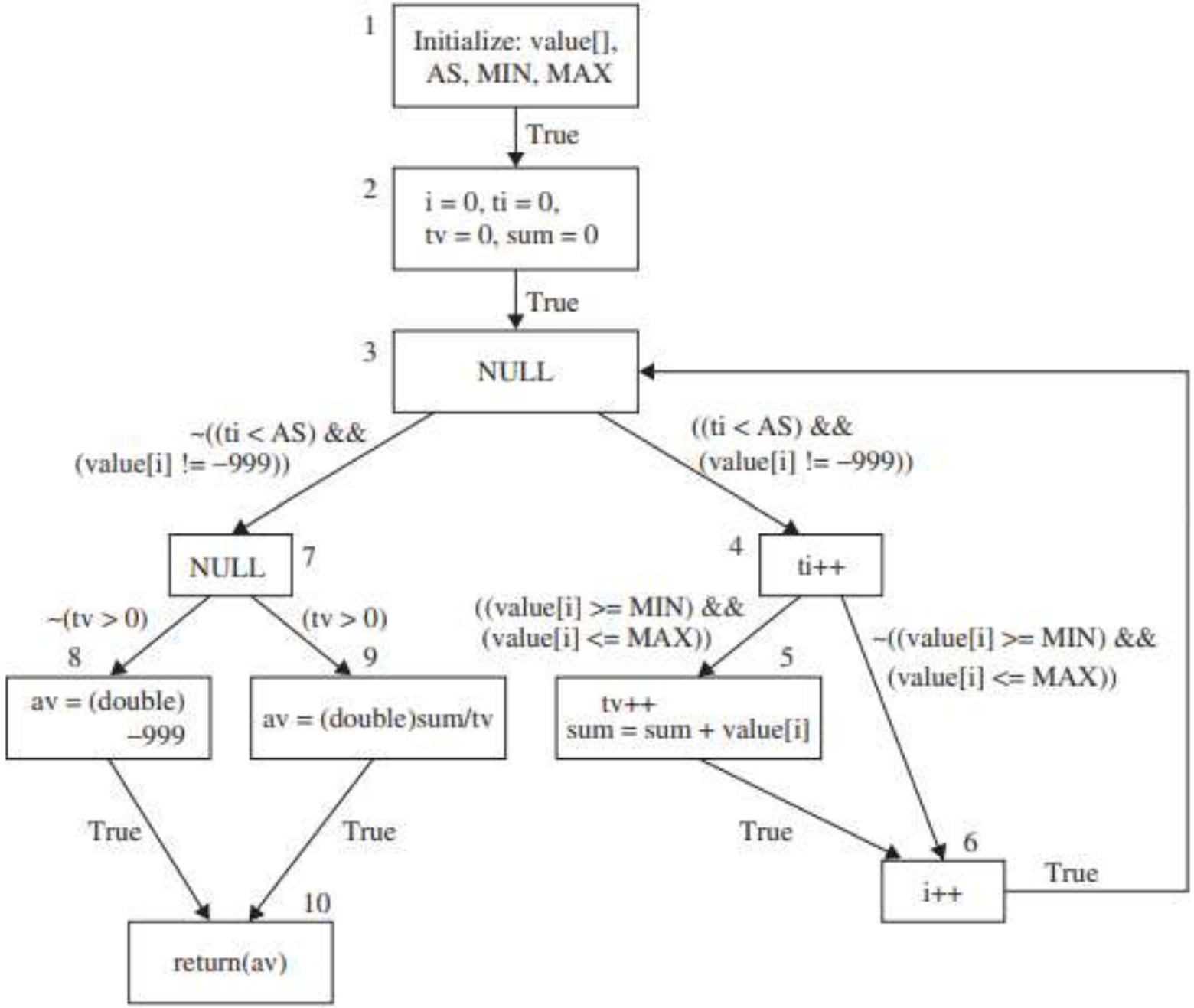


Figure 5.4 Data flow graph of ReturnAverage() example.

All p-uses / Some c-uses

- X no tiene p-uses
- Para cada variable x y para cada nodo i dado que x contenga una definición global en i, selecciona paths completos que incluya def-clear paths desde el nodo i hacia algunos nodos j dado que exista un uso global c-use de x en j

i

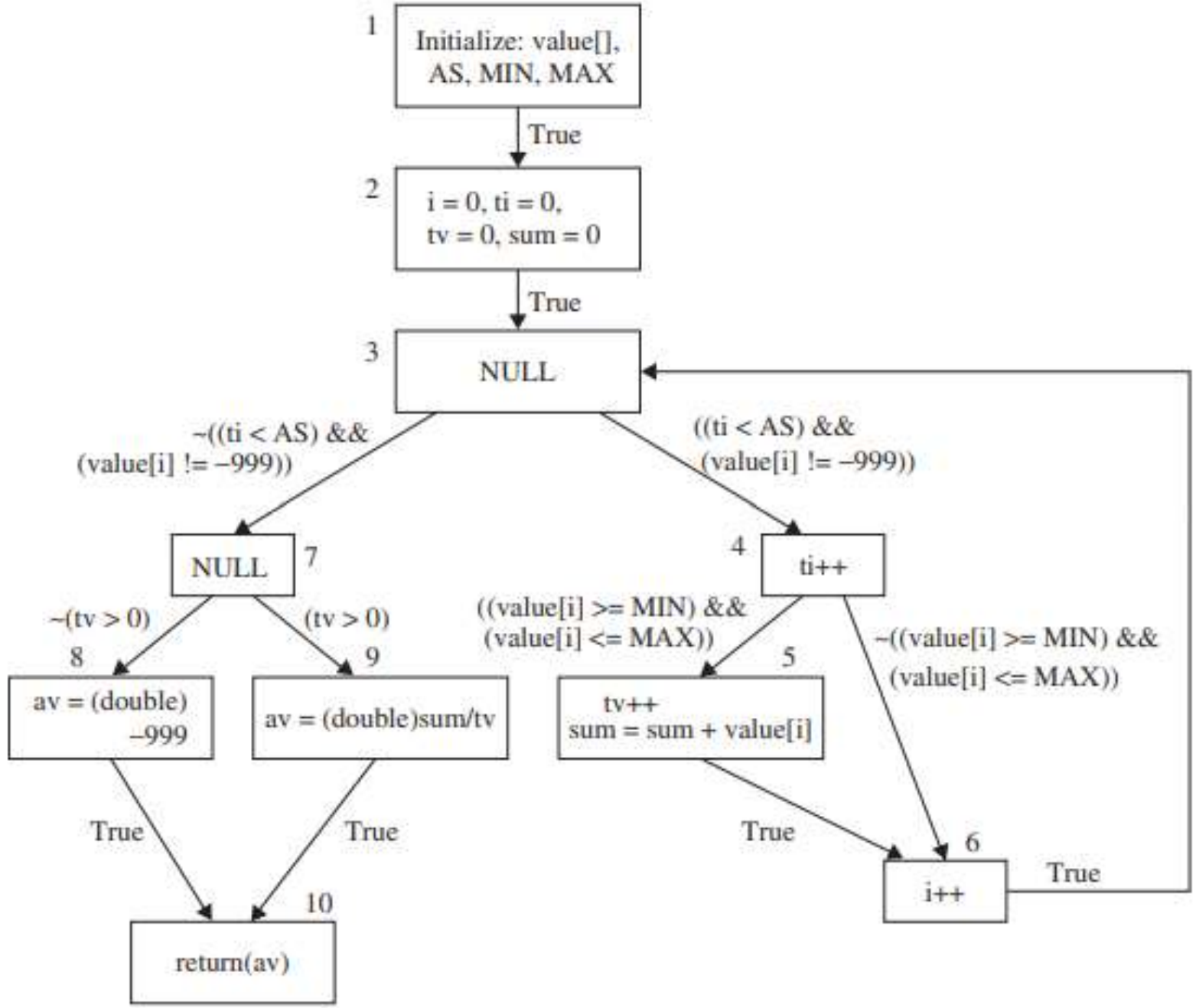


Figure 5.4 Data flow graph of ReturnAverage() example.

All-c-uses / Some-p-uses

- X no tiene c-uses
- Para cada variable x y para cada nodo i dado que x contenga una definición global en i, selecciona paths completos que incluya def-clear paths desde el nodo i hacia algunas aristas (j,k) dado que exista un p-use de x en (j,k)

AS

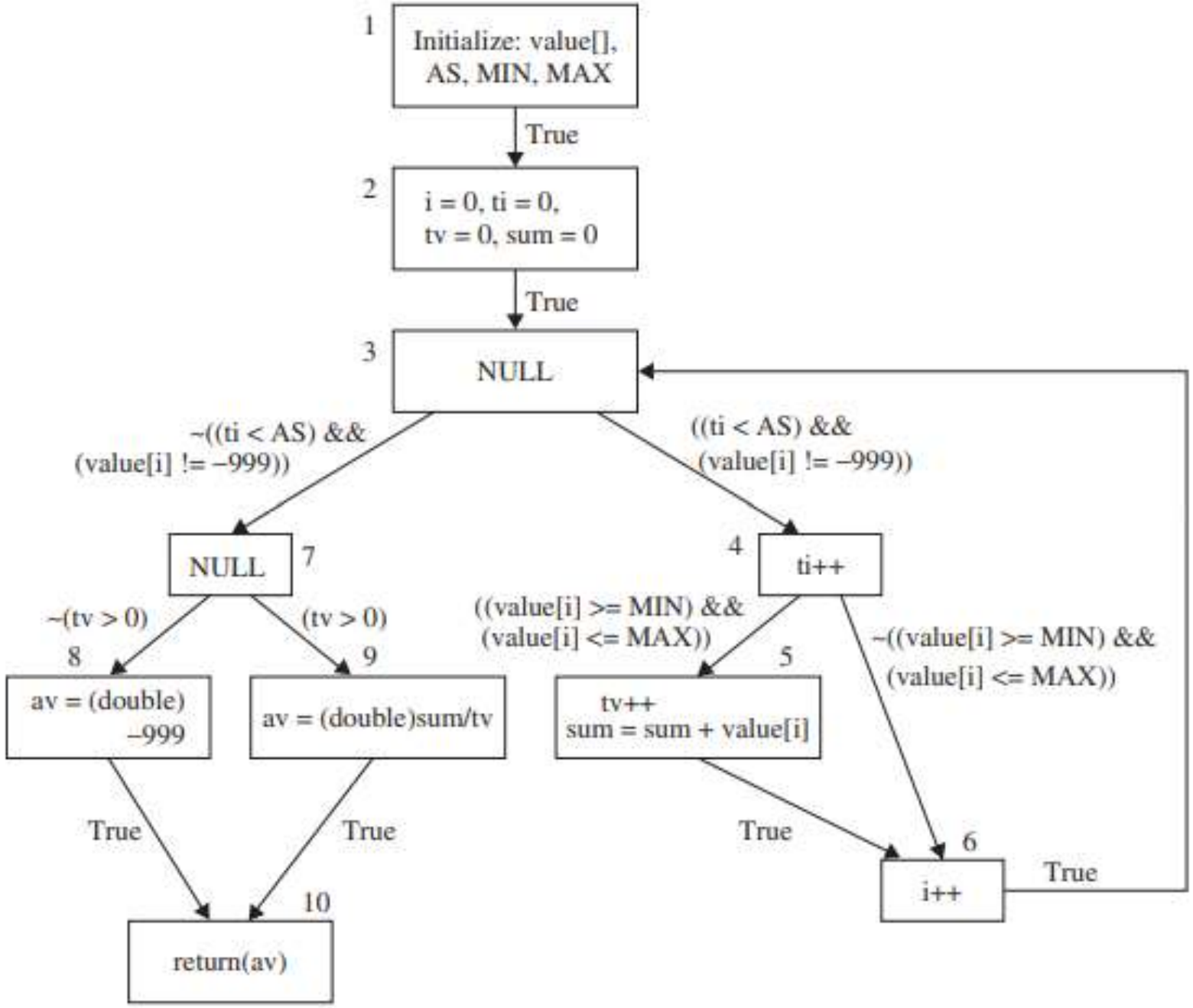


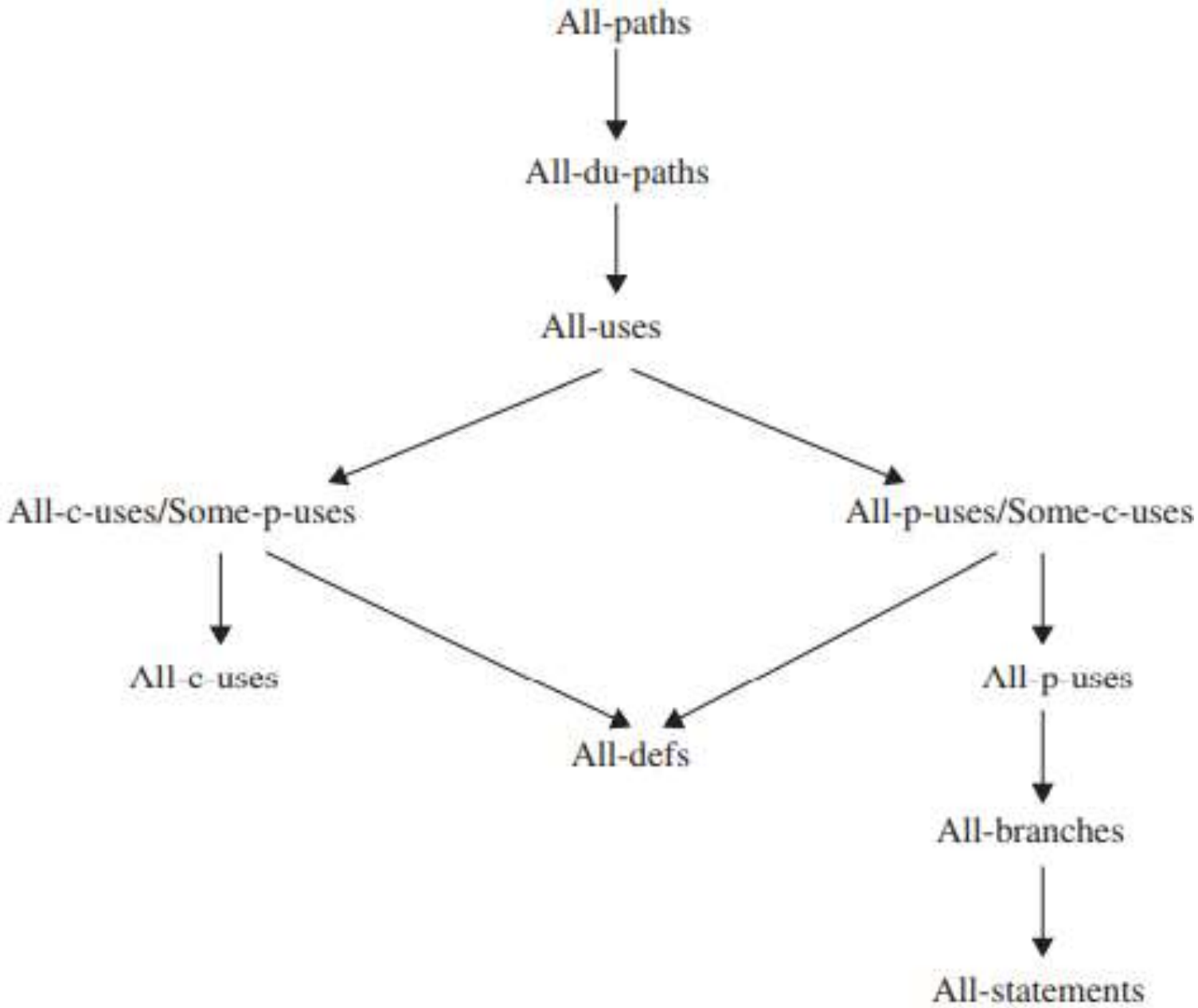
Figure 5.4 Data flow graph of ReturnAverage() example.

All uses

- Conjunto de all-p-uses y all-p-uses

All-du-paths

- Para cada variable x y para cada nodo i dado que x contenga una definición global en i, selecciona paths completos que incluya todos los du-paths desde el nodo i hasta
  - Todos los nodos j dado que exista un uso global c-use de x en j, y
  - Todas las aristas (j,k) dado que exista un p-use de x en (j,k)


All-paths
All-du-paths
All-uses
All-c-uses/Some-p-uses
All-p-uses/Some-c-uses
All-c-uses
All-defs
All-p-uses
All-branches
All-statements

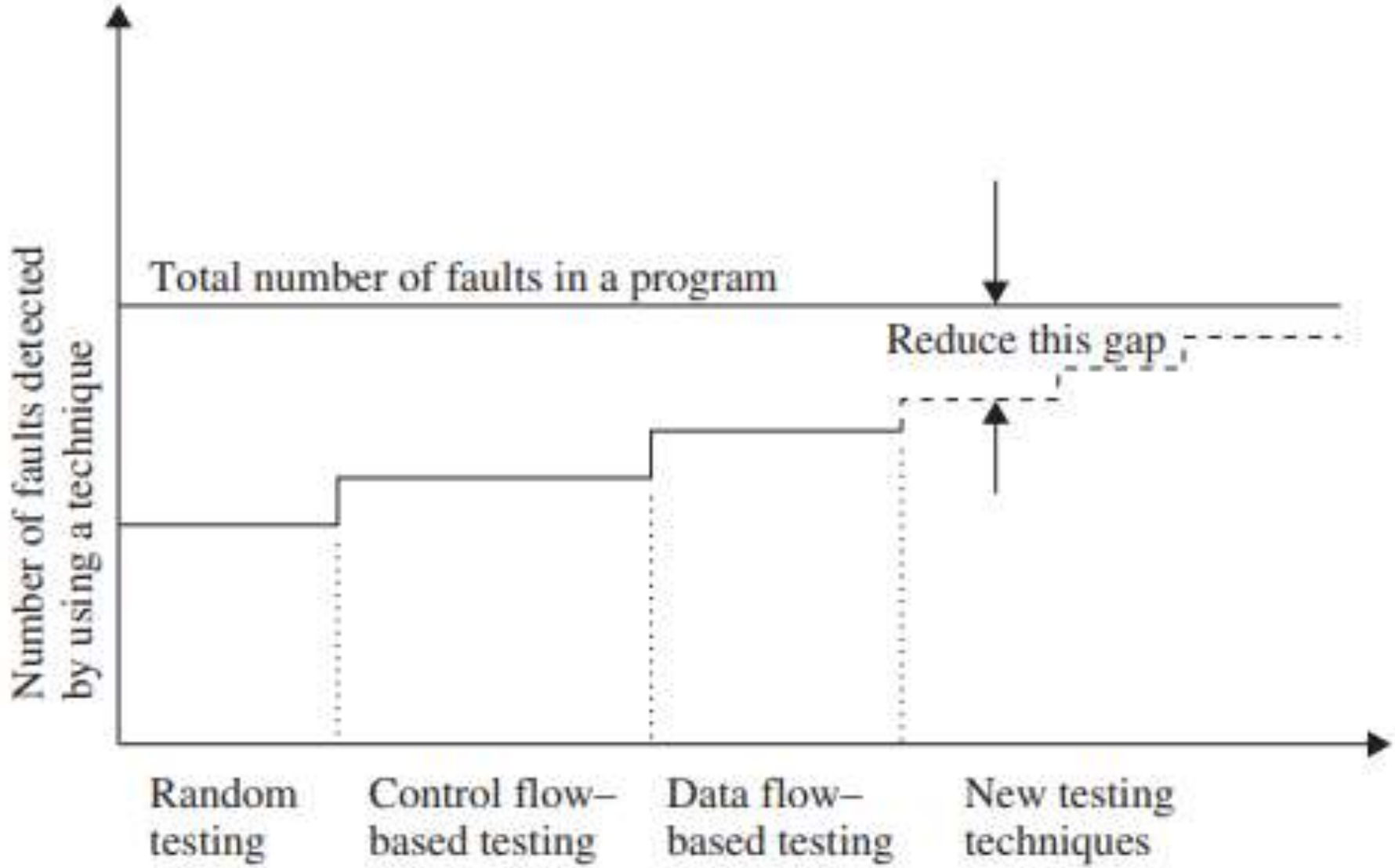


Figure 5.7 Limitation of different fault detection techniques.