Code Coverage

- 100% coverage == perfect testing ?
- code coverage = a useful metric
- code coverage for unit/integration/ system tests (or small/medium/large as Google)

necessary but not sufficient

Analiza/Verificare statică

- nu se rulează codul (white-box testing)
- produce avertismente înainte de compilare

- verifică codul pe anumite şabloane standardizate la nivelul organizației
- îmbunătățeşte calitatea codului
- testare statică vs. testare dinamică

Criteria

Control Flow

Data Flow

Control Flow

- Reprezentarea grafului Control-Flow

  noduri = operații/condiții executate secvențial

  muchii = fluxul de control dintre operații

- Scopul – acoperirea grafului C-F, prin reducerea cazurilor de test

Control Flow

Tipuri de acoperire:

- acoperirea operațiilor executabile
- acoperirea ramurilor
- acoperirea deciziilor și a condițiilor -> independența subexpresiilor
- acoperirea rutelor de execuție -> posibilele rute activate în execuția codului sursă

Ex: condiții, cicluri, ieşire din condiție/ciclu,...

Dezavantaje:

- necesitatea înțelegerii codului de către testor pt. producerea grafului C-F
- multe linii de cod (module/unități) -> numărul mare de cazuri de test

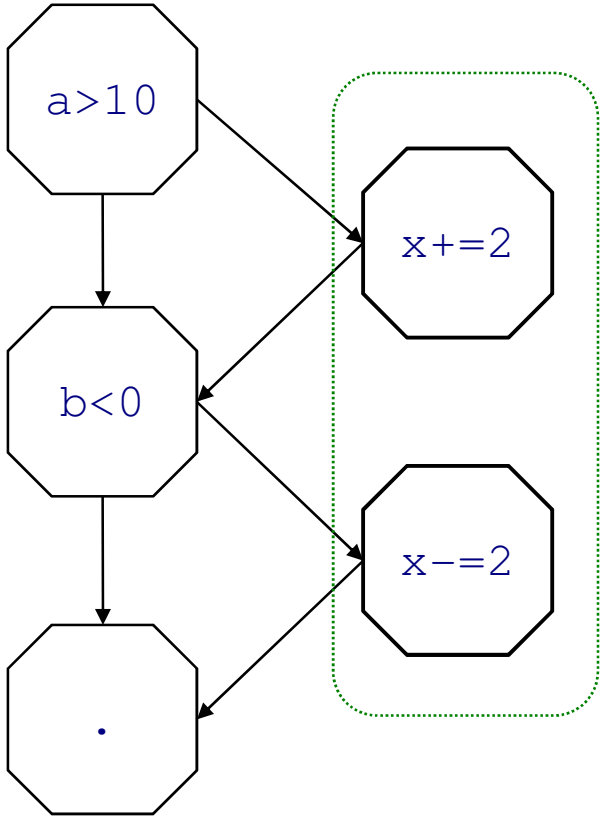
Control Flow. Acoperirea 100% a operațiilor executabile

```
if (a>10) x+=2;
if (b<0) x-=2;
```

a=11, b=-3

Acoperire completă a operațiilor executabile

Acoperire incompletă a rutelor de execuție. Comportamentul codului poate fi diferit pentru alte perechi (a,b)

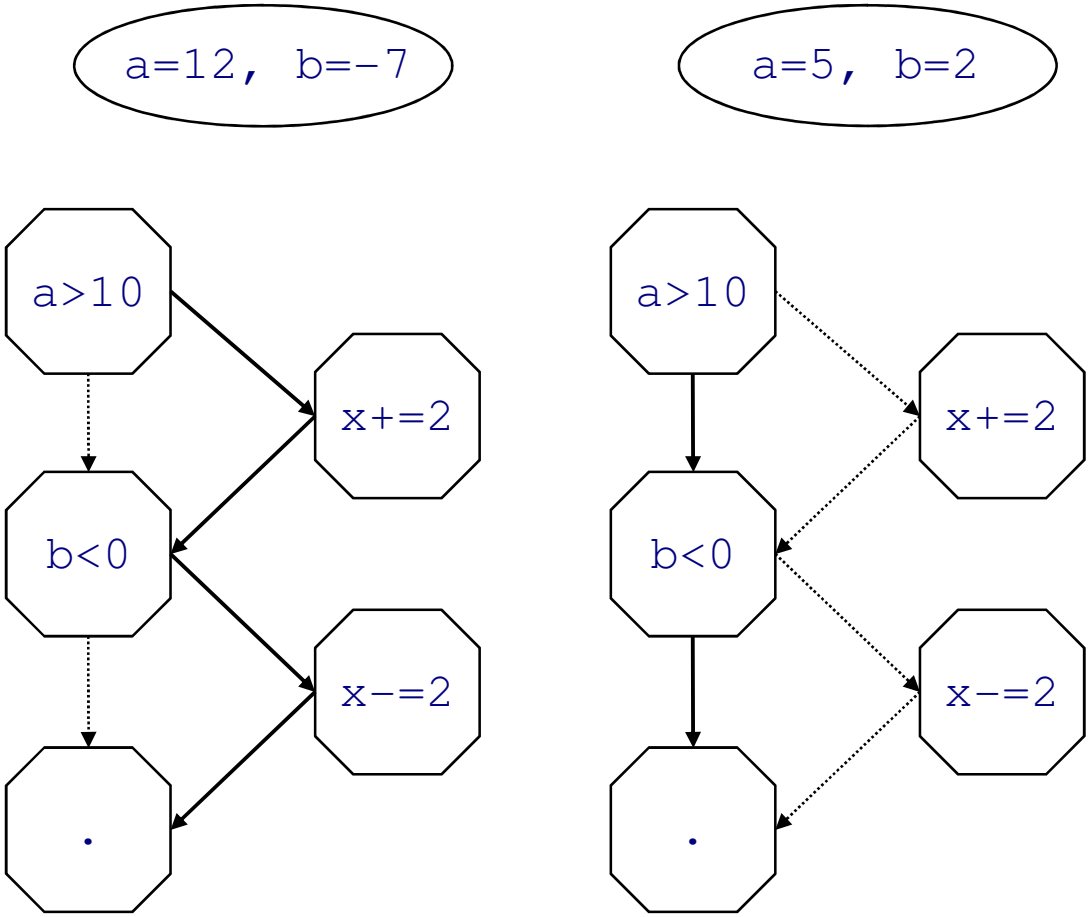


Control Flow. Acoperirea 100% a deciziilor

```
if (a>10) x+=2;
if (b<0) x-=2;
```

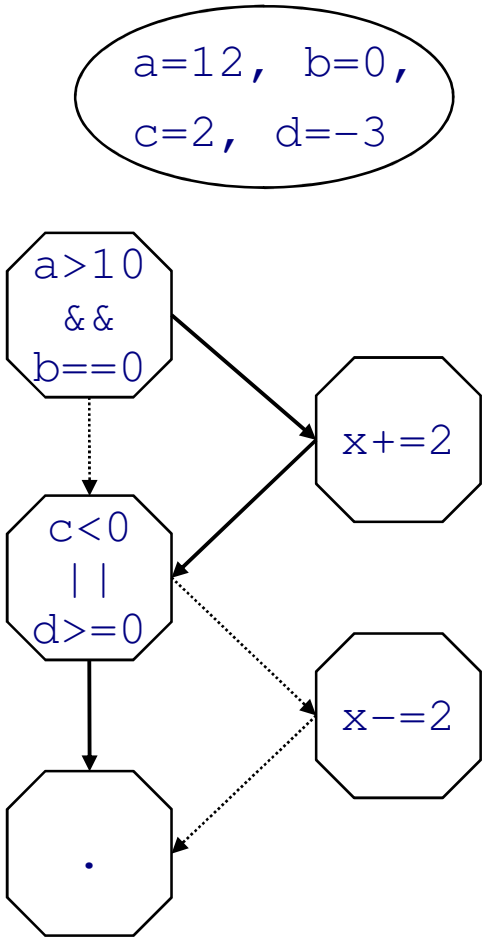
Acoperire incompletă
a rutelor de execuție.

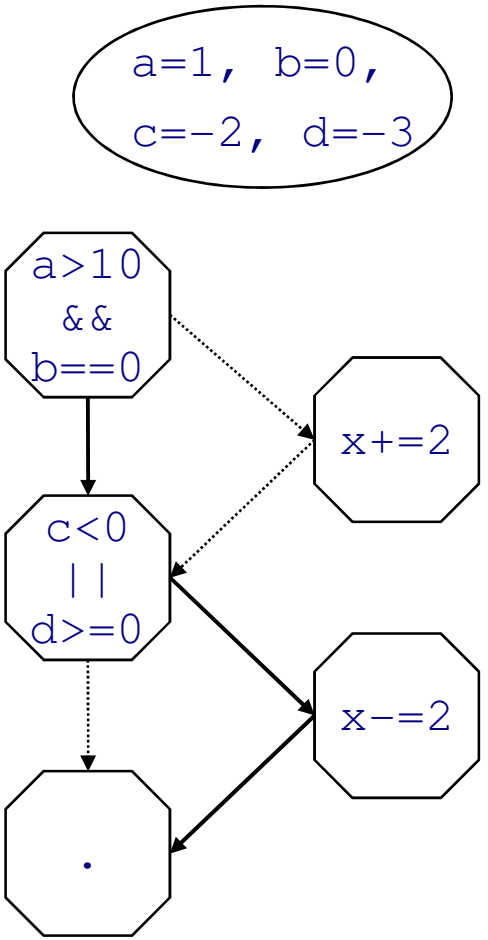
Din 4 rute au fost
acoperite 2.



Control Flow. Acoperirea 100% a condițiilor

```
if (a>10 && b==0) x+=2;
if (c<0 && d>=0) x-=2;
```

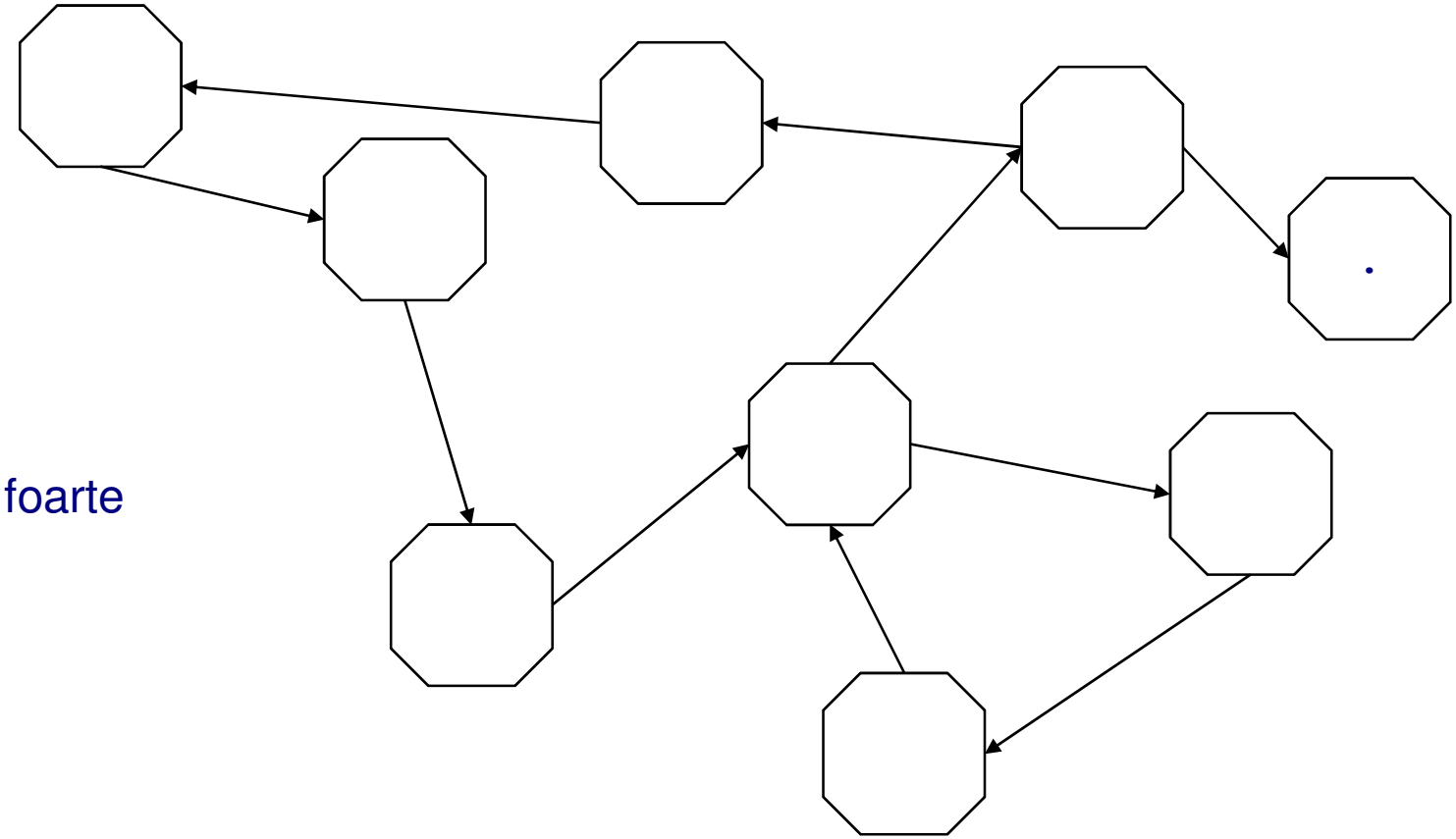




Acoperire incompletă a rutelor de execuție.

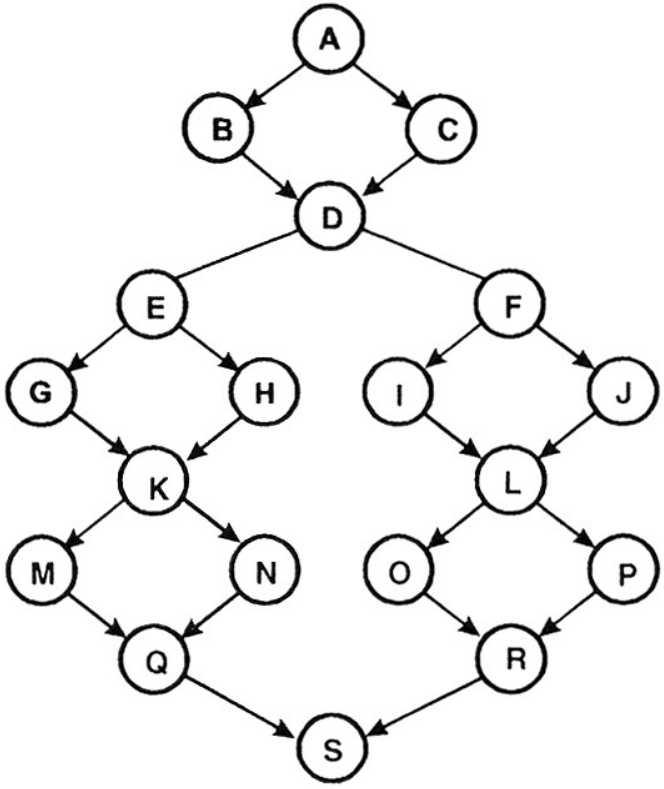
Din 4 rute au fost acoperite 2.

Control Flow. Acoperirea 100% a rutelor de execuție



Cicluri – număr foarte mare de teste

Control Flow. Complexitate ciclomatică (C)



C = muchii – noduri + 2

sau

C = p + 1 / p – numărul de decizii binare (valabil pt. tipuri de decizii exclusiv binare)

C = 24 - 19 + 2 = 7

sau

C = 6 + 1 = 7

C = numărul minim de rute de bază independente neciclice, care împreună traversează toate muchiile, având proprietatea că oricare două rute au cel puțin o muchie diferită. Garantează 100% acoperirea operațiilor executabile și a condițiilor

Control Flow. Complexitate ciclomatică

- obținerea grafului Control-Flow din codul software
- calcularea Complexității Ciclomatice (C)
- selectarea unei mulțimi de rute de bază (început/sfârșit)
- crearea unui caz de test pentru fiecare rută de bază
- executarea testelor

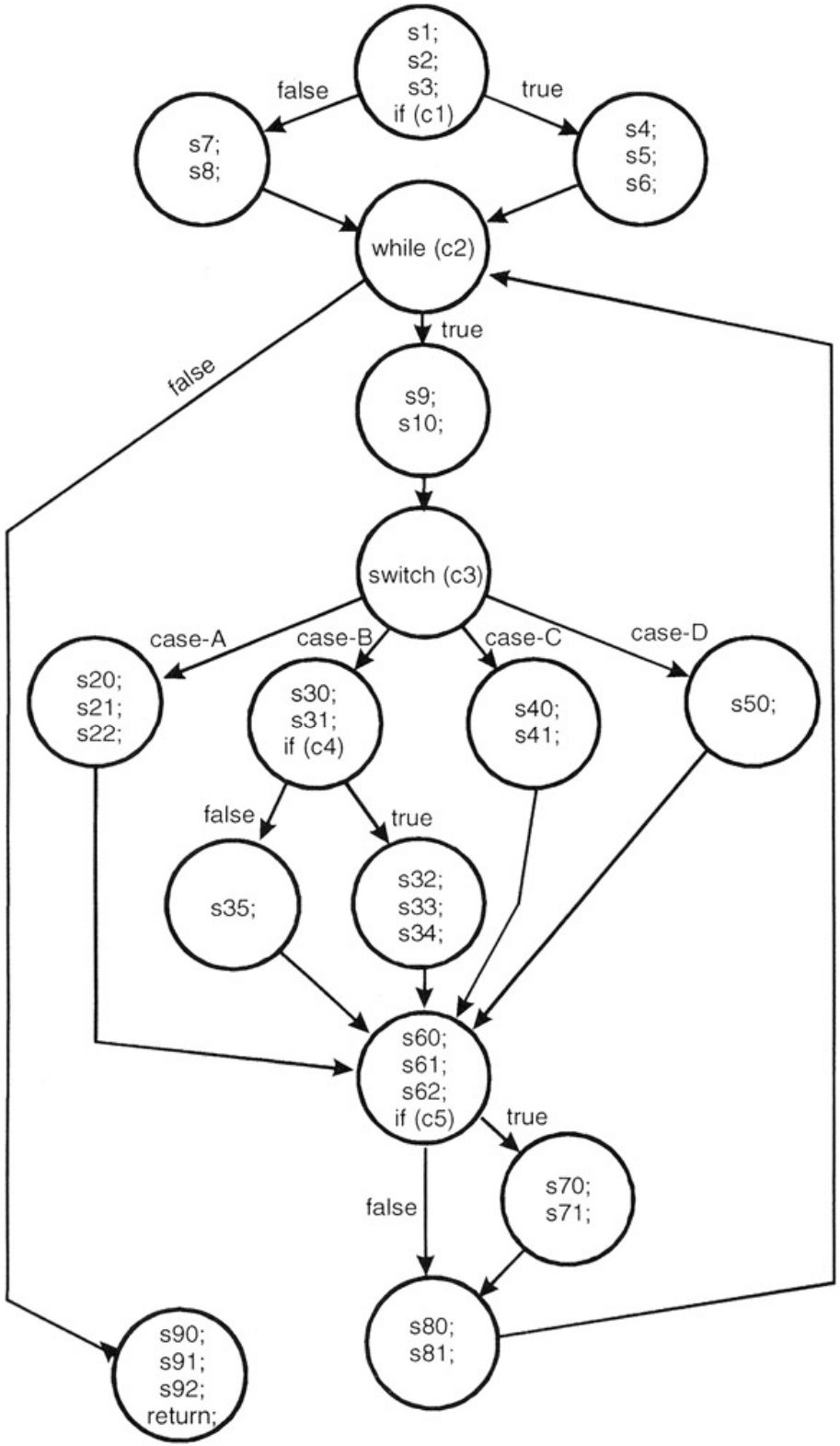
[Tom McCabe]

Control Flow. Complexitate ciclomatică. Exemplu

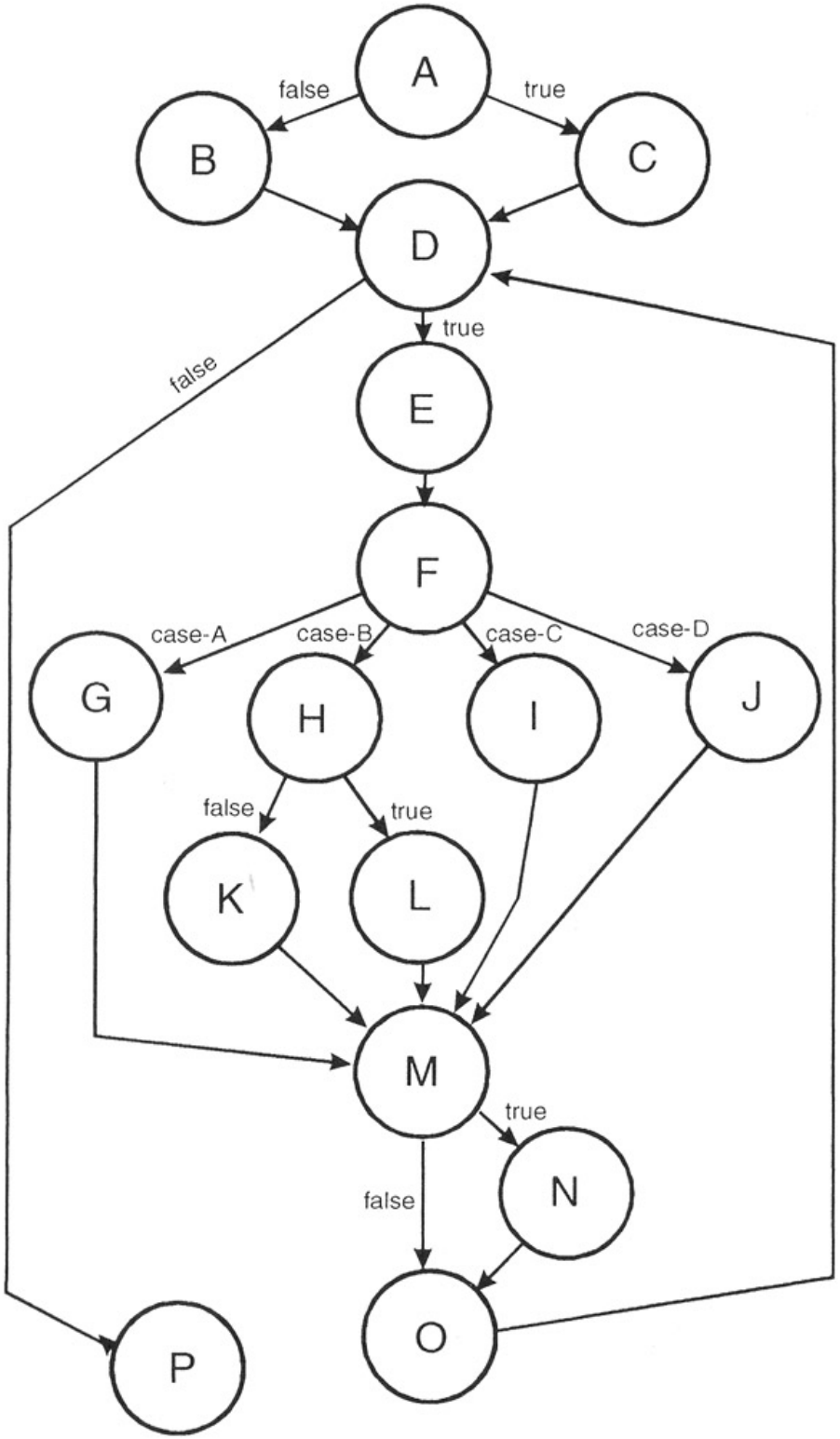
[Brown & Donaldson]

```
boolean evaluateBuySell (TickerSymbol ts) {
      s1; s2; s3;
      if (c1) {s4; s5; s6;}
      else {s7; s8;}
      while (c2) {
            s9; s10;
            switch (c3) {
                  case-A: s20; s21; s22; break; // End of Case-A
                  case-B: s30; s31;
                              if (c4) { s32; s33; s34; }
                              else { s35; } break; // End of Case-B
                  case-C: s40; s41; break; // End of Case-C
                  case-D: s50; break; // End of Case-D
            } // End Switch
            s60; s61; s62;
            if (c5) {s70; s71; }
            s80; s81;
      } // End While
      s90; s91; s92;
      return result;
}
```

si = operația i
cj = condiția j


s1;
s2;
s3;
if (c1)
false
true
s7;
s8;
s4;
s5;
s6;
while (c2)
false
true
s9;
s10;
switch (c3)
case-A
case-B
case-C
case-D
s20;
s21;
s22;
s30;
s31;
if (c4)
s40;
s41;
s50;
false
true
s35;
s32;
s33;
s34;
s60;
s61;
s62;
if (c5)
true
s70;
s71;
false
s80;
s81;
s90;
s91;
s92;
return;

C = 22 – 16 + 2 = 8



Variantă 8 rute de bază

1. ABDP
2. ACDP
3. ABDEFGMODP
4. ABDEFHKMODP
5. ABDEFIMODP
6. ABDEFJMODP
7. ABDEFHLMODP
8. ABDEFIMNODP

Cazuri de test / condiții

Test Case	C1	C2	C3	C4	C5
1	False	False	N/A	N/A	N/A
2	True	False	N/A	N/A	N/A
3	False	True	A	N/A	False
4	False	True	B	False	False
5	False	True	C	N/A	False
6	False	True	D	N/A	False
7	False	True	B	True	False
8	False	True	C	N/A	True

```
...
     if (c1) {
          while (c2) {
               if (c3) { s1; s2;
                    if (c5) s5;
                    else s6;
                    Break;} // → End while
               else
               if (c4) { }
               else { s3; s4; break;}
          } // End while
     } // End if
     s7;
     if (c6) s8; s9;
     s10;
...
```

si = operația i
cj = condiția j

Data Flow

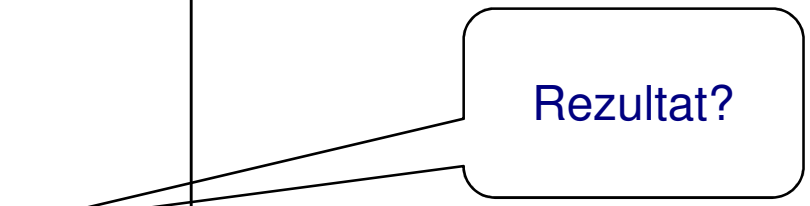
```
#include <stdio.h>

main() {

 int x;

 printf ("%d",x);

}
```

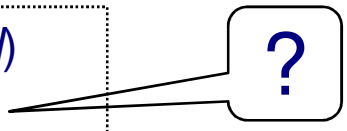


Testarea Data-Flow = detectează utilizarea eronată a variabilelor în codul sursă

Data Flow

Posibilități de primă apariție a unei variabile în cod

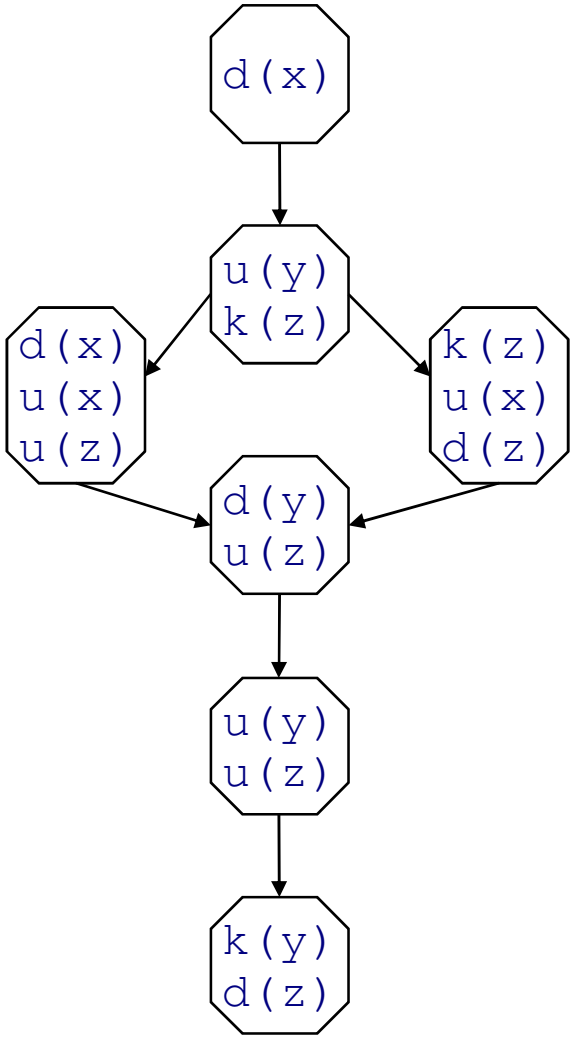
1. ~*d* variabila nu există (notat prin ~), apoi este definită (*d*)
2. ~*u* variabila nu există, apoi este utilizată (*u*)
3. ~*k* variabila nu există, apoi este distrusă (*k*)



Perechi (d,u,k)

dd – definire succesivă -> posibil eroare de codare
du – definire apoi utilizare -> corect
dk – definire apoi distrugere -> posibil eroare de codare
ud – utilizare apoi (re)definire -> ok
uu – utilizare succesivă -> ok
uk – utilizare apoi distrugere -> ok
kd – distrugere apoi (re)definire -> ok
ku – distrugere apoi utilizare -> eroare majoră
kk – distrugere succesivă -> posibil eroare de codare

Data Flow



Variabila **x**

~d – corect
dd – eroare
du – ok

du – ok

Variabila **y**

~u – eroare!
ud – ok
du – ok
uk - ok

dk – eroare

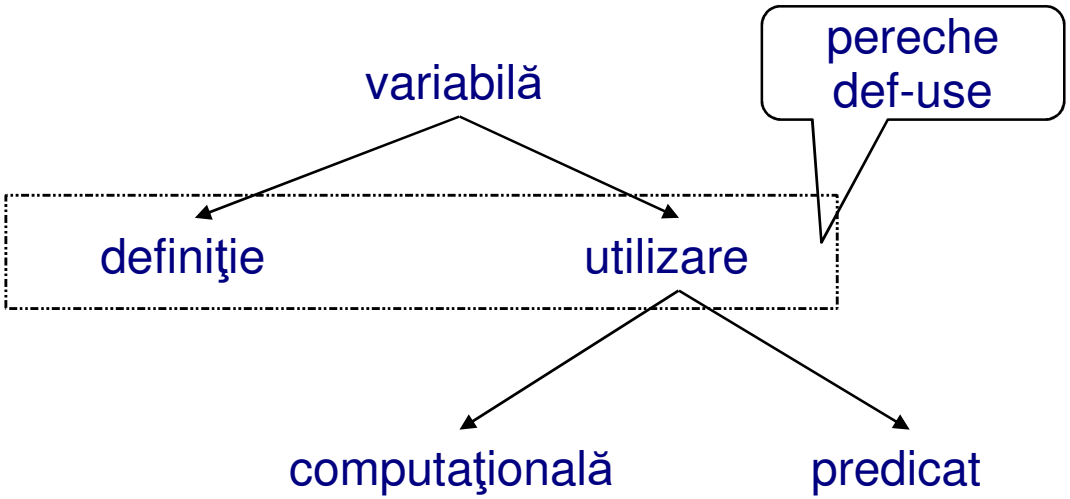
Variabila **z**

~k – eroare
ku – eroare!
uu – ok

ud – ok

kk – eroare

Data Flow



- Se bazează pe Control-Flow

- **Pentru fiecare variabilă se definește cel puțin un caz de test pentru fiecare pereche „def-use”**

Testare statică. Analiză statică

- FindBugs, Checkstyle
- Splint, Frama-C, BLAST
- CppCheck
- Perl::Critic

Exemple de analiză statică

- Recursivitate infinită
- O singură instrucţiune return la nivel de funcţie – depanare uşoară
- Greşeli de editare (=, ==) - typos
- Comparaţii între obiecte diferite
- Evitarea erorii NullPointerException
- Cod redundant

Exemple de analiză statică

Neiniţializarea variabilei

```
int f( bool b )
{
   int i;
   if ( b )
   {
      i = 0;
   }
   return i;    // i este neiniţializată dacă b =
}               false
```

Exemple de analiză statică

Dereferenţierea pointerului NULL

```
#include <malloc.h>

void f( )
{
  char *p = ( char * ) malloc( 10 );
  *p = '\0';

  //  ...
 free( p );
}
```

```
#include <malloc.h>
void f( )
{
  char *p = ( char * )malloc ( 10 );
  if ( p )
  {
    *p = '\0';

    //  ...
    free( p );
  }
}
```

Exemple de analiză statică

Ingnorarea valorii returnate

```
#include <stdio.h>

void f( )
{

  fopen( "test.c", "r" ); // valoarea
returnată ignorată

  // ...
}
```

```
#include <stdio.h>

void f( )
{
  FILE *stream;
  if((stream = fopen( "test.c", "r" )) == NULL )
    return;

  // ...
}
```

Exemple de analiză statică

Lipsă argument

```
#include <string.h>
void f( )
{

  char buff[15];
  sprintf(buff, "%s %s", "Hello, World!");
}
```

```
#include <string.h>
void f( )
{
  char buff[15];
  sprintf(buff, "%s %s ", "Hello","World");
}
```

Exemple de analiză statică

Indice depăşit

```
int buff[14]; // array de 0..13 elemente
void f()
{
   for (int i=0; i<=14;i++)
   {
     buff[i]= 0;

     // ...
   }
}
```

```
int buff[14]; // array of 0..13 elements
void f()
{
   for ( int i=0; i < 14; i++)
   {
     buff[i]= 0;
    // ...
   }
}
```

Exemple de analiză statică

Comparaţie OR cu o constantă>0

```
#define INPUT_TYPE 2
void f(int n)
{
   if(INPUT_TYPE || n)
   {
      puts("întotdeauna");
   }
   else
   {
      puts("niciodată");
   }
}
```

```
#define INPUT_TYPE 2
void f(int n)
{
   if((INPUT_TYPE & n) == 2)
   {
      puts("comparaţie AND pe biţi adevărată");
   }
   else
   {
      puts("comparaţie AND pe biţi falsă");
   }
}
```

Exemple de analiză statică

Operator incorect

```
void f( int i )
{
   while (i = 5)
   {
   // ...
   }
}
```

```
void f( int i )
{
   while (i == 5)
   {
   // ...
   }
}
```

Exemple de analiză statică

Incrementare eronată a contorului

```
void f( )
{
   int i;

   for (i = 100; i >= 0; i++)
   {
      //  ...
   }
}
```

```
void f( )
{
   int i;

   for (i = 100; i >= 0; i--)
   {
      //  ...
   }
}
```

Exemple de analiză statică

Comparaţie între bit field şi tipul boolean

```
struct myBits
{
  short flag : 1;
  short done : 1;
  //...
} bitType;

void f( )
{
  if (bitType.flag == 1)
  {
  // ...
  }
}
```

```
void f ()
{

if(bitType.flag==bitType.done)
  {
    // ...
  }
}
```

Exemple de analiză statică

Utilizare sizeof pentru o expresie

```
void f( )
{
  size_t x;
  char a[10];

  x= sizeof (a - 4);
  // ...
}
```

```
void f( )
{
  size_t x;
  char a[10];

  x= sizeof (a) - 4;
  // ...
}
```

Exemple de analiză statică

Recursivitate infinită

```
void f(int n)
{
  f=n*f(n-1);
}
```

```
void f(int n)
{
  if (n==1) return 1;
          else  f=n*f(n-1);
}
```

Exemple de analiză statică

Ciclu infinit

```
int i=2;
while (i)
{
  // ... i nu este modificat
}
```

```
int i=2;
while (i)
{
  // ... i este modificat
}
```

```
while (true)
{
  // ... nu există condiţie de ieşire break
}
```

```
while (true)
{
  // ...
 if (x==y) break;
}
```