3: Le test structurel

Statique / Dynamique

- Analyse **dynamique** : nécessite l'exécution du code binaire

Principe : à partir du code source (ou d'un modèle) et spécification, produire des DT qui exécuteront un ensemble de comportements, comparer les résultats avec ceux attendus…

1. Techniques de couverture du graphe de contrôle
   a) Couverture du flot de contrôle (toutes les instructions, branches, chemins…)
   b) Couverture du flot de données (toutes les définitions de variables, les utilisations…)
2. Test mutationnel (test par injection de défaut)
3. Exécution abstraite
4. Test évolutionniste (algorithme génétique)
5. …

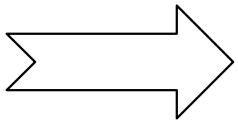
- Analyse **statique** : ne nécessite pas l'exécution du code binaire
  1. Revue de code
  2. Estimation de la complexité
  3. Preuve formelle (prouveur, vérifieur ou model-checking)
  4. Exécution symbolique
  5. Interprétation abstraite

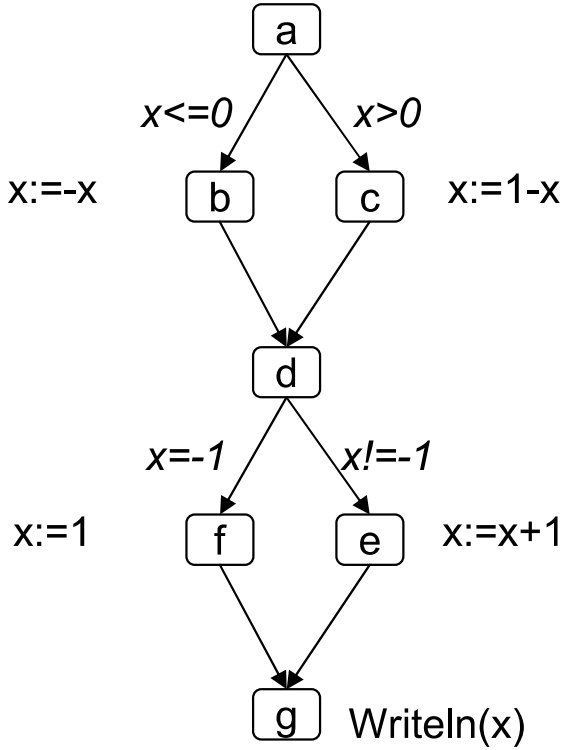
Test structurel dynamique avec technique de couverture du graphe de contrôle

But: produire des DT qui exécuteront un ensemble de comportements du programme

- Utilise : spécification, code source et code exécutable
- Un programme => un graphe de contrôle

```
begin
if (x<=0)then x:=-x
else x:=1-x;
if (x=-1)then x:=1
else x:=x+1;
end
```





Graphe orienté et connexe (N,A,e,s)

- e: un sommet entrée (a)
- s: un sommet sortie (g)
- Un sommet = un bloc d'instructions
- Un arc = la possibilité de transfert de l'exécution d'un nœud à un autre

- Une **exécution possible** = un **chemin de contrôle** dans le graphe de contrôle

  [a,c,d,e,g] est un chemin de contrôle

  [b,d,f,g] n'est pas un chemin de contrôle

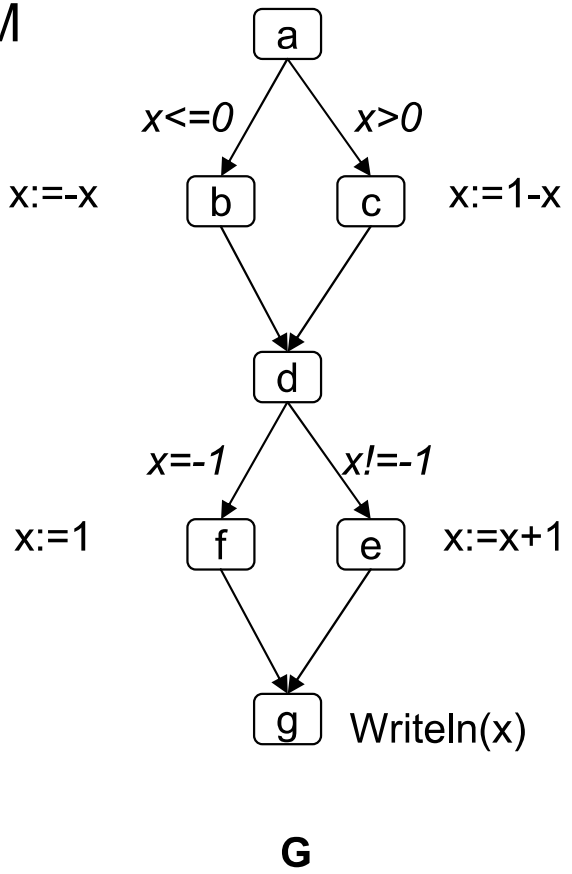
Expression des chemins d'un graphe de contrôle

Le graphe G peut être exprimé sous une forme algébrique : soit M l'ensemble des chemins de contrôle du graphe G :

M = abdfg+abdeg+acdfg+acdeg

= a.(bdf+bde+cdf+cde).g

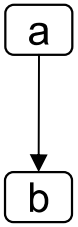
= a.(b+c)d.(e+f).g (expression des chemins de G)



G

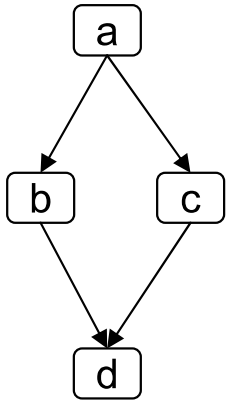
Construction de l'expression des chemins :

séquentielle



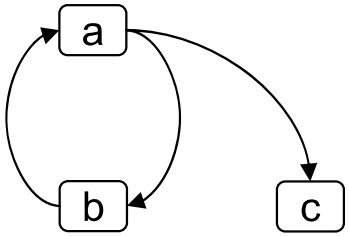
ab

alternative



a.(b+c).d

itérative



$a.(ba)^{*}.c$

$a.(ba)^{\underline{4}}.c$

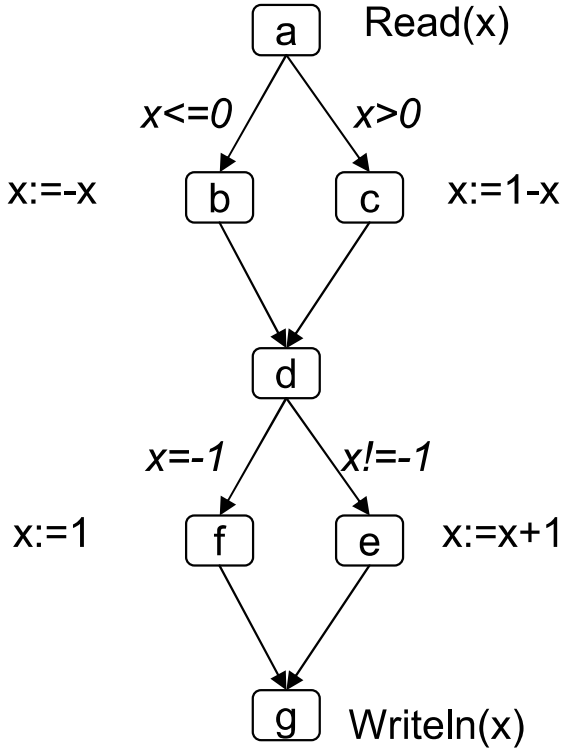
Notation utilisée (pour rappel…)

Soit X={a,b,c} un alphabet

- ε le mot vide
- $(cb)^2=cbcb$
- $b^+=b+b^2+b^3+b^4+b^5+\ldots$
- $b^*=\varepsilon+b+b^2+b^3+b^4+b^5+\ldots$
- $b^{\underline{4}}=\varepsilon+b+b^2+b^3+b^4$

Chemin exécutable

```
Read(x)
if (x<=0)then x:=-x
else x:=1-x;
if (x=-1)then x:=1
else x:=x+1;
Writeln(x)
```



DT1={x=2}

DT1 sensibilise le chemin [acdfg] : [acdfg] est un chemin exécutable

[abdgf] est un chemin non exécutable : aucune DT capable de sensibiliser ce chemin

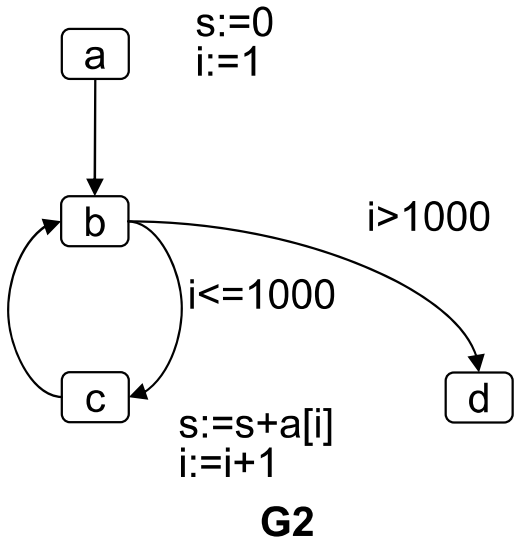
Sensibiliser un chemin peut parfois être difficile : intérêt des outils automatiques (mais attention problème de trouver des DT qui sensibilise un chemin est non décidable)

Existence de chemins non exécutables : signe de mauvais codage ?

Chemin exécutable / chemin non exécutable

- Nombre de chemins de contrôle de G :
  - se déduit directement de l'expression des chemins de G
  - a(b+c)d(e+f)g => 1.(1+1).1.(1+1).1 = 4 chemins de contrôle
  - Nb chemins exécutables + Nb chemins non exécutables
  - Parfois le Nb chemins non exécutables peut être important :

```
begin
s:=0;
for i:=1 to 1000 do s:=s+a[i];
end
```



Expression des chemins de G2 : $a.b.(cb)^{\underline{1000}}.d$

Nombre de chemins :

$1.1.(1.1)^{\underline{1000}}.1 = 1^{\underline{1000}} = 1+1^1+1^2+ \ldots +1^{1000}=1001$

Parmi ces 1001 chemins, un seul est exécutable: $a.b.(cb)^{1000}.d$

Exercice 1

```
lire(b,c,x);
if b<c
then begin
     d :=2*b ;
     f :=3*c
     if x>=0
     then begin
          y := x ;
          e := c ;
          if (y=0)
          then begin
               a :=f-e ;
               if d<a
               then begin
                    writeln(a)
                    end
               else begin
                    writeln (d)
                    end
               end
          end
     end
```

- Donner le graphe de contrôle G(P3) associé au programme P3.
- Donner 3 chemins de contrôle du graphe G(P3).
- Donner l'expression des chemins de contrôle de G(P3).
- Soit DT1={b=1,c=2,x=2}. Donner le chemin sensibilisé par DT1.
- On s'intéresse aux instructions en italique… Donner des DT qui vont couvrir ces instructions.
- Donner un chemin de contrôle non exécutable de G(P3).

Problème des chemins non exécutables

- Étant donné un chemin qu'on a envie de sensibiliser, comment trouver une DT qui exécute ce chemin ? Problème très difficile:
  1. décider si le chemin est exécutable ou pas;
  2. s'il l'est trouver une DT.

Le problème 1 est indécidable.

[indécidable = formellement impossible de construire un algorithme **général** qui décide de l'exécutablilité ou de la non exécutabilité de n'importe quel chemin]

La présence de chemins non-exécutables est souvent signe de code mal écrit, voire erroné !

Il existe des outils (plus ou moins automatiques) de sensibilisation de chemins (basés sur l'interprétation abstraite ou sur des techniques de vérification de programmes)

Exercice 2

```
Lire(choix)

if choix=1

then x=x+1 ;

if choix=2

then x=x-1 ;

writeln(choix ;
```

1. Donner le graphe de contrôle correspondant au programme P4.
2. Donner l'expression des chemins de contrôle de G(P4). En déduire le nombre de chemins de contrôle.
3. Donner les chemins de contrôle non exécutables. Conclure.
4. Proposer une nouvelle solution pour ce programme. Construisez son graphe de contrôle et donner l'expression des chemins de contrôle ainsi que le nombre de chemins de contrôle.

Exercice 3

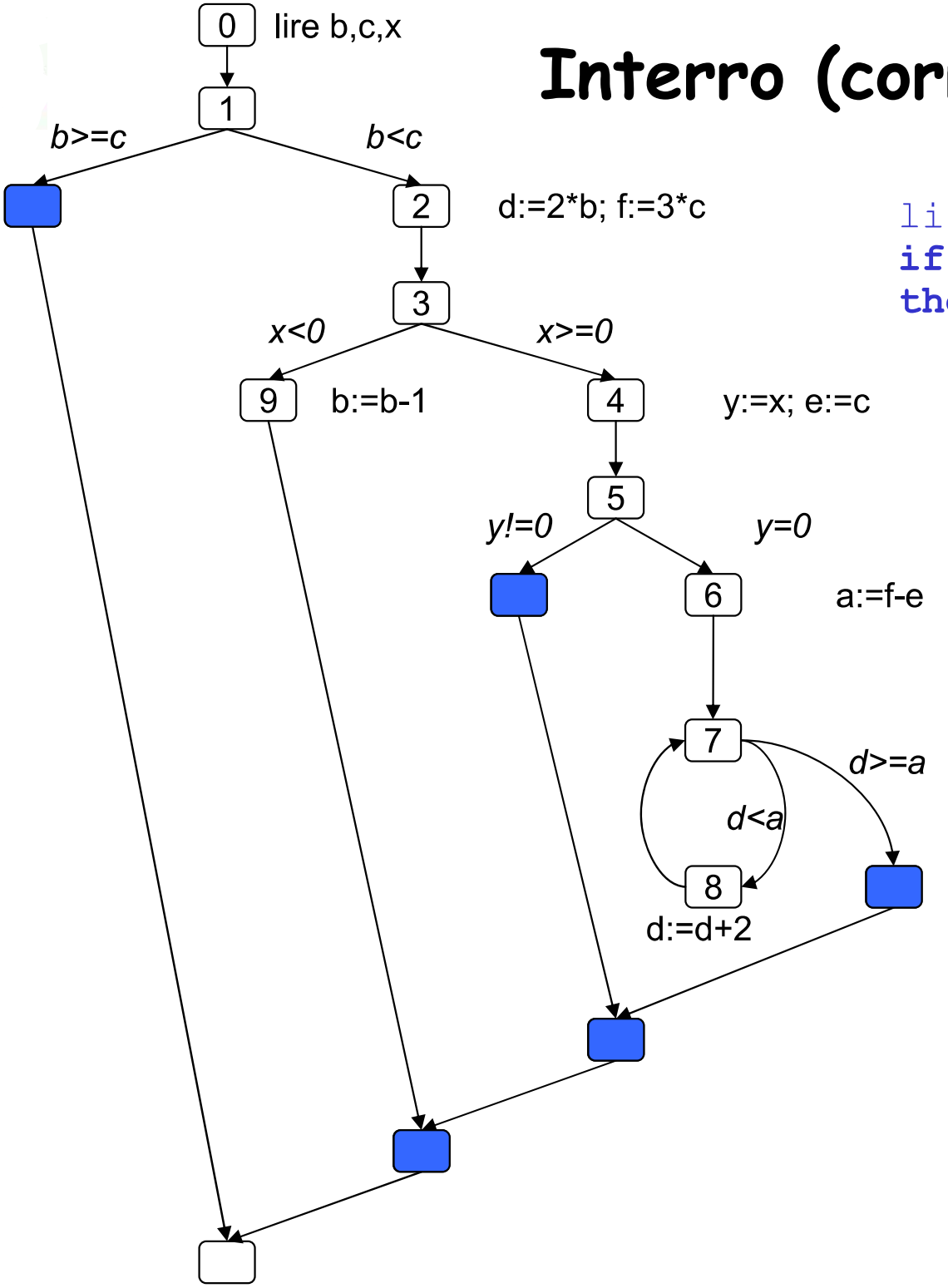
1. Écrivez un algorithme de recherche de l'emplacement d'un élément e dans un tableau T.
2. Donner le graphe de contrôle associé.
3. Donner l'expression des chemins.
4. Dans le cas où le tableau a une taille de 3, donner le nombre de chemins de contrôle.

Interro

```
lire(b,c,x)
if b<c
then begin
     d :=2*b
     f :=3*c
     if x>=0
     then begin
          y := x
          e := c
          if (y=0)
          then begin
               a :=f-e
               while d<a
                    begin
                    d:=d+2
                    end
               end
          end
     else begin
          b:=b-1
          end
     end
```

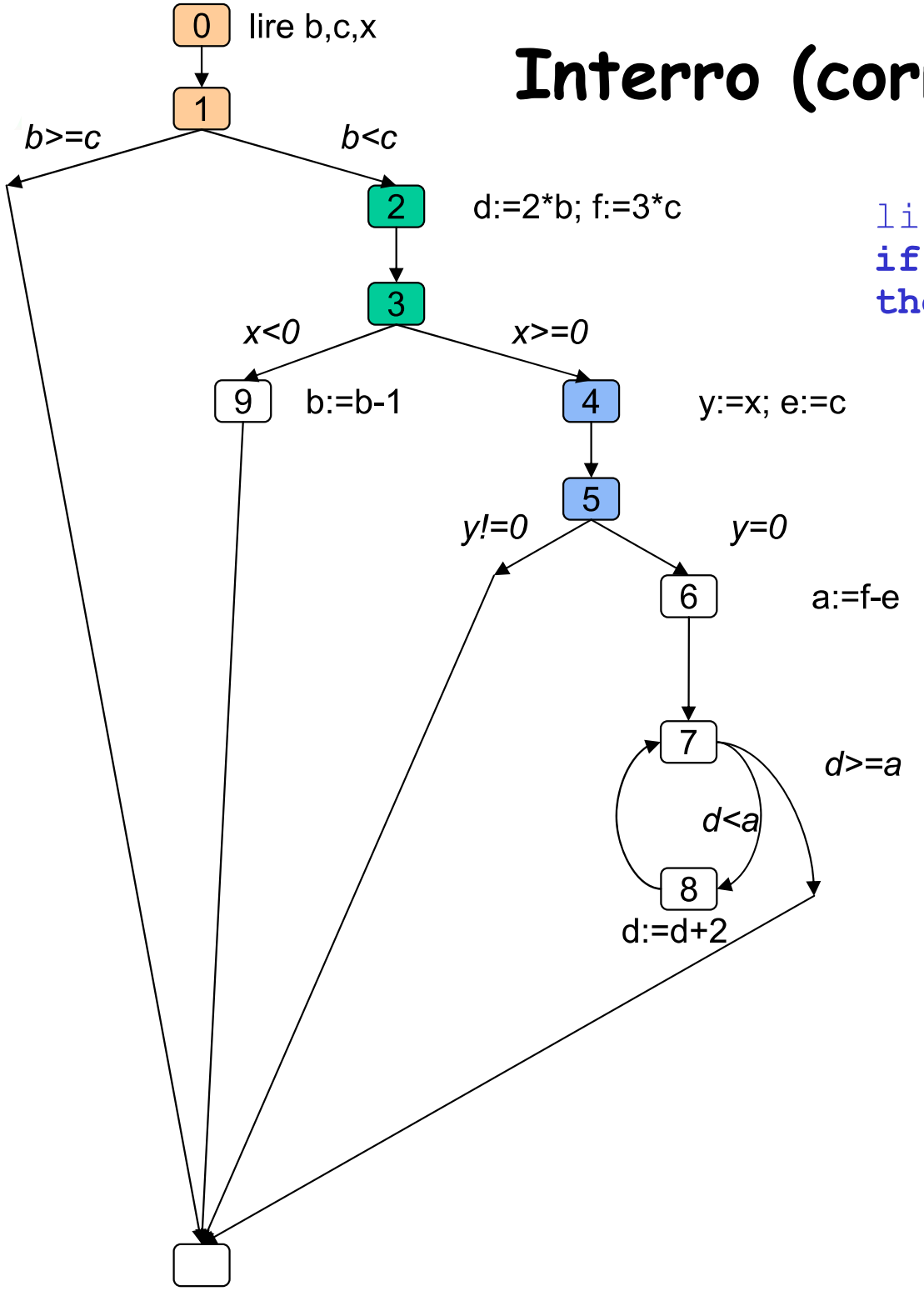
1. Donnez un graphe de contrôle associé au code source fourni.
2. Votre graphe de contrôle a-t-il des possibilités de réduction ?
3. Si oui, réduisez votre graphe de contrôle.

Interro (correction)



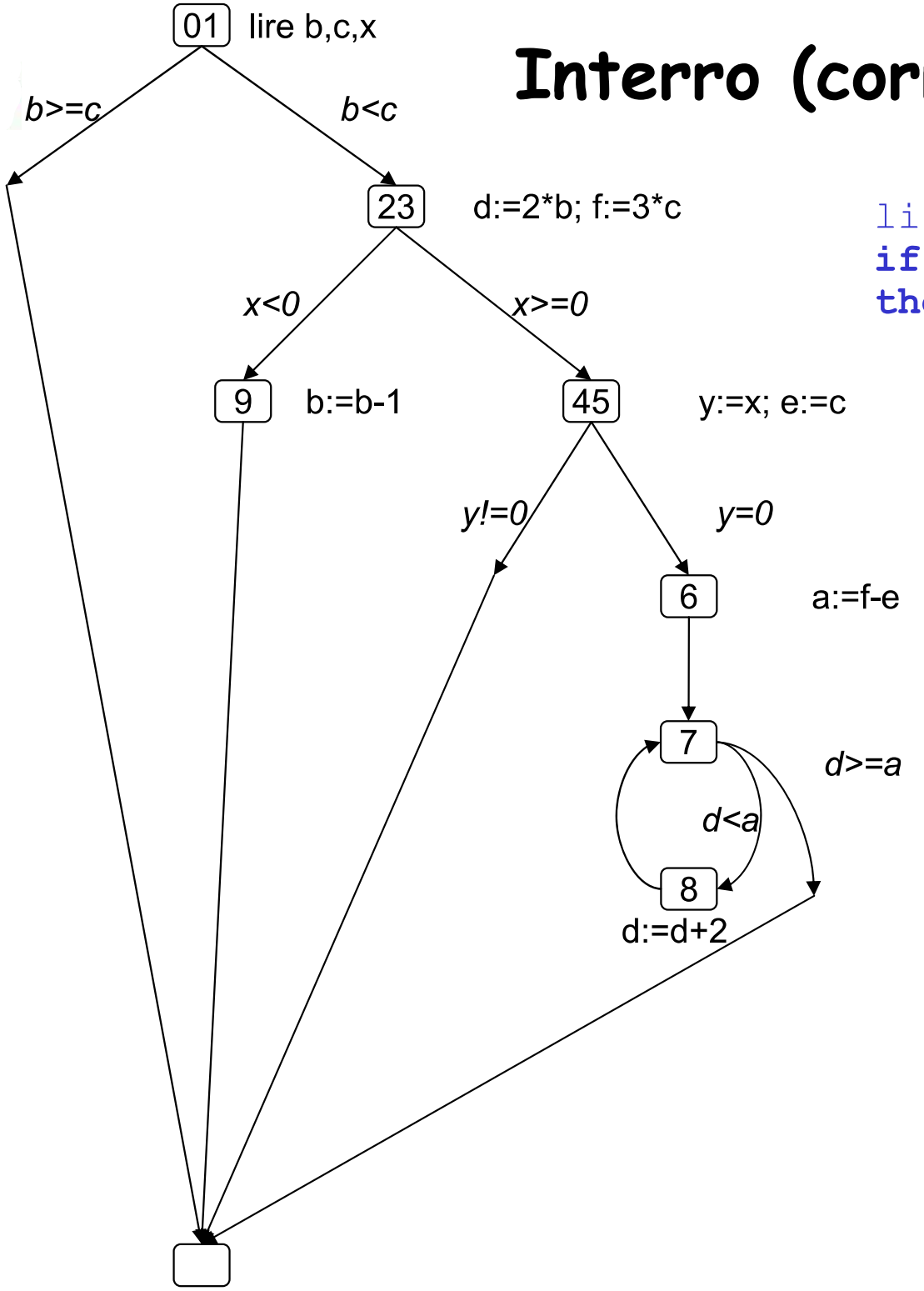

```
lire(b,c,x)/*0*/
if b<c /*1*/
then begin
      d :=2*b /*2*/
      f :=3*c
      if x>=0 /*3*/
      then begin
            y := x /*4*/
            e := c
            if (y=0) /*5*/
            then begin
                  a :=f-e /*6*/
                  while d<a /*7*/
                        begin
                        d:=d+2 /*8*/
                        end
                  end
            end
      else begin
            b:=b-1/*9*/
            end
end
```

Interro (correction)




```
lire(b,c,x)/*0*/
if b<c /*1*/
then begin
      d :=2*b /*2*/
      f :=3*c
      if x>=0 /*3*/
      then begin
            y := x /*4*/
            e := c
            if (y=0) /*5*/
            then begin
                  a :=f-e /*6*/
                  while d<a /*7*/
                        begin
                        d:=d+2 /*8*/
                        end
                  end
            end
else begin
      b:=b-1/*9*/
      end
end
```

Interro (correction)




```
lire(b,c,x)/*0*/
if b<c /*1*/
then begin
      d :=2*b /*2*/
      f :=3*c
      if x>=0 /*3*/
      then begin
            y := x /*4*/
            e := c
            if (y=0) /*5*/
            then begin
                  a :=f-e /*6*/
                  while d<a /*7*/
                        begin
                        d:=d+2 /*8*/
                        end
                  end
            end
else begin
      b:=b-1/*9*/
      end
end
```

Satisfaction d'un test structurel avec couverture

Soit T un test structurel qui nécessite la couverture d'un ensemble de chemins $\{\delta_1, \ldots, \delta_k\}$ du graphe de contrôle.

On notera : T= $\{\delta_1, \ldots, \delta_k\}$

Soit DT une donnée de test qui sensibilise le chemin de contrôle C.

- Définition: DT **satisfait** T ssi C couvre tous les chemins de T.
- Exemple : Soient le graphe de contrôle G5, δ_1=cdebcde, δ_2=ce et T1= $\{\delta_1, \delta_2\}$.

DT1 ={a[1]=-2, a[2]=3, a[3]=17,i=1} satisfait-il T1 ?

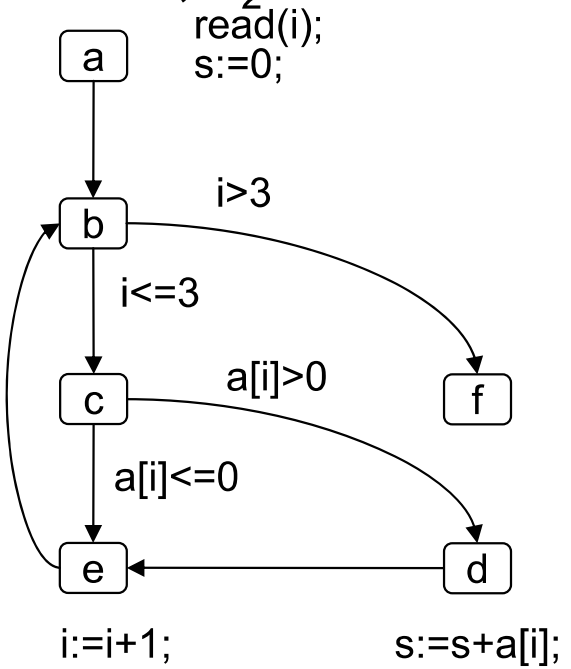
DT1 sensibilise M1=abcebcdebcdebf

M1= abcebcdebcdebf couvre δ_1=cdebcde

M1=abcebcdebcdebf couvre δ_2=ce

Donc DT1 satisfait T1

DT2 ={a[1]=-2, a[2]= 3, a[3]=-17,i=1} satisfait-il T1 ?



G5

Hiérarchie des techniques de test structurel

- Exemple : considérons le graphe de contrôle G5 et les 2 tests structurels avec couverture T1 et T2 définis par :

δ_1=cdebcde, δ_2=ce et T1= {δ_1, δ_2}

δ_3=de, δ_4=b, δ_5=cd et T2= {δ_3, δ_4 , δ_5}.

Lorsque T1 est satisfait, T2 l'est aussi : pourquoi ?

T1 est un test plus **fiable** (ie. 'fort') que T2 et on notera :

$T1 \Rightarrow T2$

- $\Rightarrow$ est une relation d'ordre partielle (réflexive, antisymétrique, transitive)
- $\Rightarrow$ permet de définir une hiérarchie entre les différentes techniques structurelles de test (relation d'ordre partielle)