2: Le test fonctionnel
2.2 : Le test fonctionnel de logiciel

Introduction

- Notre contexte : pas possible d'exprimer toutes les combinaisons de DT.
- Le test fonctionnel est basé sur la spécification/interface du programme : il n'examine pas le programme, mais son comportement.
- Chaque technique de test fonctionnel fournit une méthode avec un critère de sélection/production des DT (à partir des spécifications) :
  1. Analyse partitionnelle
  2. Test aux limites
  3. Graphe Cause-Effet
  4. Test syntaxique
  5. Etc…

Analyse partitionnelle : un exemple

Élaboration d'un logiciel de traitement des notes du BAC

- On distingue les différents profils de lycéens,
- On suppose que si le calcul est bon pour un lycéen A avec un certain profil, il sera exact pour tout lycéen B de même profil
- Exemple de profil : lycéen redoublant sa terminale ES option math, ne repassant pas l'épreuve anticipée de Français, mais repassant l'épreuve anticipée SVT.

Problème : comment construire ces différentes classes ?

- domaine des valeurs d'entrées

Analyse partitionnelle

- Un programme P traite des données en entrée :

  $P : D \rightarrow R$

- On partitionne D en différentes classes D_i telles que :
  1. $D = D_0 \cup D_1 \cup \ldots/\ldots \cup D_N$
  2. $D_i \cap D_j = \varnothing$ si $i \neq j$
  3. On sélectionne un seul cas de test par classe D_i

- Hypothèse de test :

  $\forall i \in [1,N]$, $\forall d,d' \in D_i$: P(d) correct $\Leftrightarrow$ P(d')correct

Analyse partitionnelle : exercices

Soit les spécifications suivantes :

1 - « Si la valeur n est négative : un message d'erreur est affichée. Si n est dans [1,20[on affiche la valeur exacte de Factoriel(n). Si n est dans [20,200] on affiche une approximation de Factoriel(n) en virgule flottante avec une précision de 0,1%. Si n>200 un message d'erreur est affichée. »

2 - « Écrire un programme qui calcule $F(x)=(1/x)^{1/2}$ »

*3 - « L'instruction FOR n'accepte qu'**un seul paramètre** en tant que variable auxiliaire. Son nom ne doit pas dépasser **2 caractères non blancs**. Une borne supérieure et une borne inférieure doivent être précisées: **la borne inférieure est précédée du mot-clé '='** et **la borne supérieure est précédée par le mot-clé 'TO'**. Les bornes sont des **entiers positifs**. »*

On se propose de tester des programmes correspondants à ces spécifications : donner un jeu de test associé à chaque spécification.

Test aux limites

Constat : erreurs souvent dues au fait que le programmeur n'avait pas prévu le comportement du logiciel pour des valeurs aux limites (aussi appelées valeurs de bord):

- Valeurs très grandes
- Valeur de boucle nulle, négative, maximale
- Données non valides
- Etc…

Remarque : Souvent utilisée avec la technique de partitionnement : les valeurs aux limites peuvent être des valeurs aux frontières des partitions

Exemple de choix des valeurs de test.

- Variable dans un intervalle de valeurs [a,b] :

a, b, a $\pm \Delta$, b $\pm \Delta$ (Δ : plus petite variation possible)

- Variable dans un ensemble de valeurs ordonnées {a_1, a_2, a_3,…/…,a_n} :

a_1, a_2, a_{n-1}, a_n (premier, second, avant dernier, dernier)

Test aux limites : des exemples

- Une variable a dans le domaine [**1**,**10**]
  Valeurs à tester : 0,**1**,2,9,**10**,11
- Deux variables a et b dans le domaine [**1**,**10**]^2
  6x6=36 valeurs à tester : {0,**1**,2,9,**10**,11}^{2}
- La notion de limite n'est pas toujours évidente ! Deux variables a et b dans le domaine [-10,10]^2
  - Rajouter les valeurs au voisinage de 0,
  - les valeurs où a=b,
  - etc. (selon son intuition… en fonction de la spécification, conception…)

Test aux limites : Exercice

- Un programme de classification de triangles prend en entrée un triplet de réels (a,b,c) correspondants aux longueurs des 3 côtés d'un triangle. Le programme doit préciser la nature du triangle (équilatéral, isocèle, scalène, impossible)

  Donner des exemples de valeurs aux limites.

Test syntaxique

But : déterminer des DT à partir d'une description formelle (sous forme BNF ou automate à états finis) des données

Type d'application visé : application nécessitant des données d'entrée respectant une syntaxe (analyseur d'expression, interpréteur, compilateur, etc.)

Méthode :

1. Définir une grammaire
2. Construire l'arbre de dérivation 'générique'
3. Produire des DT en se basant sur l'arbre de dérivation

   Couverture des nœuds Terminaux, des nœuds Non Terminaux

   Deux cas :

   1. Entrées valides
   2. Entrées non valides

Test syntaxique : exercice

Interpréteur qui reconnaît des commandes du type :

```
copy <fic1> to <fic2>
rename <fic1> to <fic2>
```

Où `<fic1>` et `<fic2>` sont des noms de fichiers formés sur une ou deux lettres prises dans l'ensemble {a,b}.

1-Donner une grammaire

2-Donner un arbre de dérivation 'générique'

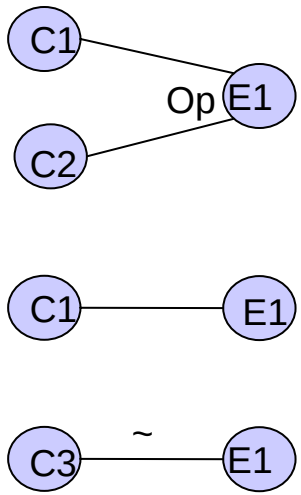
3-Produire des DT

Graphes Cause-Effet

Approche proposée par G.J Myers

Méthode : proposer un graphe/réseau qui relie les effets du programme (sorties) aux causes (entrées) qui sont à leur origine.

Syntaxe: Op::=And|Or|Nand|Nor



Graphes Cause-Effet : exercice

P prend en entrée une longueur (entier entre 1 et 20), une chaîne de caractères de cette longueur, et un caractère. P retourne sa position dans la chaîne ou un message d'erreur. Il est possible de cherche d'autres caractères.

Précises les causes et les effets

Donner le graphe Cause-Effet

En déduire des DT

Exercice

Une procédure de classification de triangles reçoit en entrées 3 réels a, b et c qui sont les longueurs des côtés d'un triangle. La procédure retourne comme résultat 0 si le triangle défini par a, b et c est invalide, 1 si le triangle est équilatéral, 2 si le triangle est isocèle et 3 pour un triangle valide quelconque (ni isocèle, ni équilatéral).

a) Proposez un partitionnement des données en entrée pour tester cette procédure.

b) En déduire un jeu d'essai pour cette procédure.