

I.2: Le test fonctionnel

I.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. $D = D_0 \cup D_1 \cup \dots / \dots \cup D_N$
 2. $D_i \cap D_j = \emptyset$ si $i \neq j$
 3. On sélectionne un seul cas de test par classe D_i
- Hypothèse de test :
 $\forall i \in [1, N], \forall d, d' \in D_i : P(d) \text{ correct} \Leftrightarrow P(d') \text{ correct}$

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 $\text{Factoriel}(n)$. Si n est dans $[20,200]$ on affiche une approximation de $\text{Factoriel}(n)$ 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 $\{a_1, a_2, a_3, \dots, a_n\}$:
 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 $[1,10]^2$
 $6 \times 6 = 36$ 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'

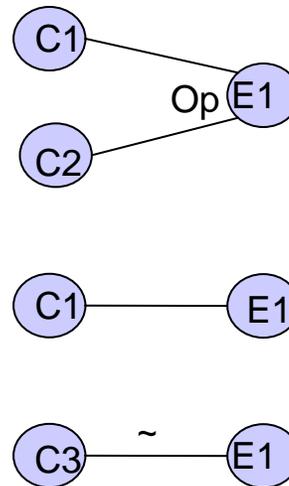
3-Produire des DT

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 chercher d'autres caractères.

Préciser 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.

Travaux Pratiques:

Ascenseur

On vous demande de construire un simulateur d'ascenseur (en Java ou dans un autre langage de votre choix).

Variables

étage : l'étage de la porte

Comportement

- Attendre que l'ascenseur soit à l'arrêt à l'étage de la porte
- Ouvrir la porte
- Attendre un certain laps de temps
- Fermer la porte
- Signaler à l'ascenseur qu'il peut redémarrer

Variables

étage : l'étage courant de l'utilisateur

direction : la direction que l'utilisateur veut emprunter

destination : la destination de l'utilisateur

Comportement

- Si un appel a été signalé au même étage en direction opposée :
 - Attendre
 - Sinon, appeler l'ascenseur
- Attendre que la porte s'ouvre
- Décider d'entrer ou non (l'utilisateur peut être distrait)
- Si la porte est encore ouverte, entrer dans l'ascenseur
- Entrer la destination
- Attendre que la porte se ferme
- Attendre que l'ascenseur soit à destination
- Attendre que la porte s'ouvre et sortir

Spécification : l'ascenseur

Variables

- `étage` : l'étage courant de l'ascenseur
- `direction` : la direction courante de l'ascenseur
- `destinations` : un vecteur des destinations entrées par les usagers
- `appels` : un vecteur des appels effectués par les usagers

Comportement

- Choisir la direction
- Monter ou descendre d'un étage selon la direction
- Renverser la direction si l'ascenseur atteint l'étage le plus haut (resp. le plus bas)
- Si le nouvel étage correspond à un appel ou une destination
- Effacer tout appel ou destination pour l'étage courant
- Signaler d'ouvrir la porte
- Attendre la fermeture de la porte

Comportement

Choisir la direction :

- Selon la direction courante, chercher un appel ou une destination vers le haut ou le bas
 - S'il n'y a aucune direction courante, commencer à chercher vers le haut
- S'il existe un appel à l'étage courant, indiquer qu'il n'y a pas de direction courante
- S'il existe un appel ou une destination vers la direction courante et que l'ascenseur n'est pas à l'étage le plus haut (resp. le plus bas)
 - Maintenir la direction courante
 - Sinon, chercher pour un appel ou une destination dans la direction opposée
- S'il existe un appel ou une destination dans la direction opposée et que l'ascenseur n'est pas à l'étage le plus bas (resp. le plus haut)
 - Changer la direction à la direction opposée
 - Sinon, indiquer qu'il n'y a pas de direction courante

Comportement global

- Lorsque l'ascenseur est en mouvement, aucune porte n'est ouverte.
- Il est toujours vrai qu'un usager qui demande l'ascenseur y entrera fatalement
- Il n'y a jamais plus d'une porte d'ouverte à la fois.
- La distance parcourue par un usager est toujours égale à $|\text{source} - \text{destination}|$.

Spécification : Simplification

Exemple de simplification :

- Un seul ascenseur,
- Deux usagers au même étage demandent l'ascenseur : si leurs directions sont différentes, un usager attend.

Fin ascenseur

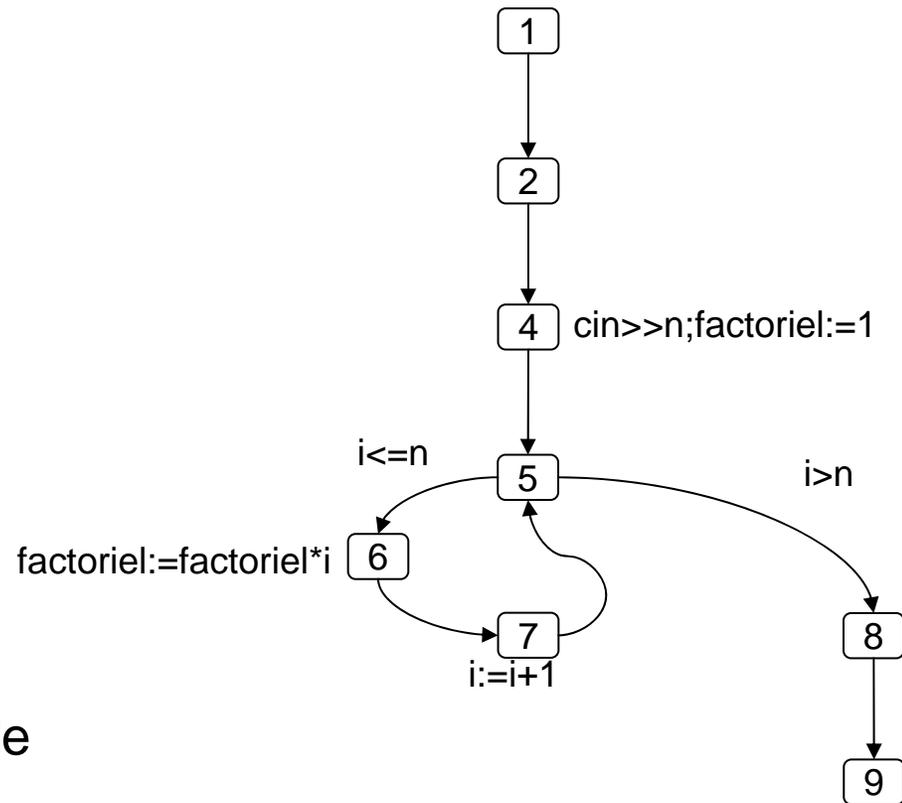
Exercice

Soit le programme suivant:

```

1 main( )
2 {
3   int i, factoriel;
4   cin >> n; factoriel = 1;
5   for( i = 1; i <= n; i++ )
6     { factoriel = factoriel * i;
7     }
8   printf( "%d\n", factoriel );
9 }

```



Question : donner des exemples de couverture de flot de données.

