# TD de Test Logiciel

## Sélection de tests boîte blanche

**Exercice 1** (Mutations)**.**

*Soit le programme C suivant :*

```
1  /* Outputs result = 0+1+...+|value|
2   * if result > maxint then error
3   */
4  void maxsum(int maxint, int value) {
5    int result =0;
6    int i =0;
7    if (value < 0)
8      value = -value;
9    while (i< value && result <= maxint) {
10     i++;
11     result = result+i;
12   }
13   if (result <= maxint)
14       println(result);
15   else
16       println("error");
17 }
```

1. *Générer 5 mutants (ordre 1) distinguables au moyen de ROR et ABS.*
2. *Générer les DT pour tuer ces mutants, calculer le score de mutation obtenu, calculer les couvertures I, C, D obtenues.*
3. *Prenez les DT que votre voisin a trouvé pour la question précédente, calculer le score de mutation de ces DTs sur vos mutants.*
4. *Générer 1 mutant non distinguable avec ROR et 1 mutant non distinguable avec ABS.*
5. *Reprendre chacun de vos jeux de tests de l'exo 5, et calculer son score de mutation.*
6. *Définissez un opérateur de mutations O tel que un jeu de tests O-adéquat couvre toutes les instructions. Idem avec décisions et conditions.*

Correction. Voici quelques éléments de correction. On note $P$ le programme initial, et $M$ un de ces mutants. Le résultat retourné par un programme $C$ sur des entrées $d$ se note $C(d)$. Soit une variable $v$ d'un programme. On note $v_l$ la valeur de $v$ à la ligne $l$ du programme.

1. pour les mutants : ordre 1 = obtenu du programme initial par une seule mutation ; distinguable = il existe effectivement une DT $d$ telle que $P(d) \neq M(d)$. Voici par exemple des mutants possibles (le caractère distinguable sera prouvé à la question 2).

- mutant M1 : (ABS) ligne 8 : $value = -|value|$
- mutant M2 : (ROR) ligne 7 : $value > 0$

2. Voici un test T qui permet de tuer M1 et M2 (ce qui montre qu'ils sont distinguables).
   T : entrée (10,1). $P(T) = 1$ , $M1(T) = 0$, $M2(T) = 0$.
   Comme sur l'entrée T, $P(T) \neq M1(T)$, $T$ permet de tuer (ou distinguer) M1. Idem pour M2. Le score de mutation de T pour $\{M1, M2\}$ est de :
   score(T) = (nombre de mutants tués) / nombre de mutants = $2/2 = 1$. (le meilleur score possible)
   Remarque : Si notre jeu de tests comptait plusieurs tests $T_1, \ldots, T_n$, on compterait les mutants tués par au moins un des $T_i$.
   Couverture
   - instructions : il y a 11 instructions à couvrir (lignes 5 6 7 8 9 10 11 13 14 15 16), le test T en couvre 8.
   - décisions : il y a 6 décisions (ligne 7 avec vrai / faux, ligne 9 avec vrai/faux, ligne 13 avec vrai faux ), le test T en couvre 4
   - conditions : il y a 8 conditions (par rapport aux décisions, il faut compter les conditions atomiques de la ligne 9), le test T en couvre 5

   Remarque : les couvertures structurelles atteintes par ce jeu de test n'est pas très bonne car on ne considère que deux mutants (et qu'un seul test suffit). En pratique, on génère des milliers de mutants, et les jeux de tests associés ont de bonnes couvertures structurelles.
3. 
4. Deux mutants non distinguables avec ABS : ligne 8 $value = |-value|$, et ligne 11 $result = |result| + i$
   La première est bien indistinguable : à l'entrée de la ligne 8, $value$ est négatif donc $-value$ est positif, donc $|-value| = value$. Malgré la mutation, les valeurs des variables à la fin de la ligne 8 sont identiques dans $P$ et dans $M$, donc pour toute entrée $d$, $P(d) = M(d)$. Le deuxième mutant est indistinguable avec le même type d'argument.
   Un mutant non distinguable avec ROR : ligne 7 $value \leq 0$. Montrons que ce mutant est bien indistinguable. Soit $d$ une entrée du programme. Si avec cette entrée $d$, $value$ est différent de $0$ à l'entrée de la ligne 7, rien ne change donc $M(d) = P(d)$. Si $value$ vaut $0$ à l'entrée de la ligne 7 alors à l'entrée de la ligne 9 : dans $P$, $value_9 = value_7$, donc $value_9 = 0$ ; dans $M$, $value_9 = -value_7$, donc $value_9 = -0 = 0$. Là encore, pour toute entrée $d$ à partir de la ligne 9 toutes les variables ont même valeur pour $P$ et $M$, donc pour toute entrée $d$, $P(d) = M(d)$.
5. point à regarder : bien souvent, des jeux de tests conçus pour une couverture structurelle réalisent un faible score de mutation.
6. pour les instructions : on définit l'opérateur $O$ par : à chaque instruction à la ligne k, créer un mutant en ajoutant en fin d'instruction "`; assert(false,"ligne k'')`".
   ($\Rightarrow$) Soit un programme $P$ et $TS$ un jeu de test $O$-adéquat, alors tous les mutants $M$ de $P$ sont tués par $TS$. On remarque d'abord qu'il y a un mutant par instruction du programme. On note $M_k$ le mutant obtenu en modifiant la ligne k. Pour tuer le mutant $M_k$, il faut au moins qu'un test de $TS$ passe par l'instruction k. Comme $TS$ tue tous les $M_k$, $TS$ passe donc par chaque instruction.
   ($\Leftarrow$) Le point clé ici est qu'un test atteignant la ligne k distingue forcément le mutant $M_k$, puisque le programme $P$ continue l'exécution normalement tandis que le mutant $M_k$ s'arrête en situation d'erreur (par construction du mutant). Soit $TS$ telque $TS$ couvre chaque instruction. Donc TS tue chaque $M_k$ puisque pour chaque $k$, il existe $T \in TS$ passant par k, et un tel test suffit à tuer $M_k$ (remarque précédente). Donc $TS$ est $O$-adéquat.

   Décision : avant chaque instruction branchante à la ligne k utilisant une expression booléenne $C$, créer deux mutants en ajoutant avant le test : soit "`; assert(C,"ligne k'')`", soit "`; assert(not C,"ligne k'')`"

   Condition : idem que les décisions, mais en détaillant les conditions atomiques.

□