Génie Logiciel Avancé

Année 2021-2022

Prof. Burkhart Wolff
wolff@lri.fr

Hugo Mlodecki, Carmelo Vaccaro
hugo.mlodecki@universite-paris-saclay.fr
carmelo.vaccaro@universite-paris-saclay.fr

TD 7 - Test structurel

Semaine du 20 novembre 2017

Exercice 1

On considère une fonction qui prend un tableau de caractères s en argument et renvoie vrai si et seulement si un caractère a apparaît strictement plus de fois qu'un caractère b dans s.

```
boolean comp_occurrences(char s[], char a, char b) {
    int res = 0;
    int i = 0;
    while(i < s.length) {
        if(s[i] == a) { res++; }
        if(s[i] == b) { res--; }
        i++;
    }
    return (res > 0);
}
```

1. Donner une spécification formelle de `comp_occurences`. **Astuce** : En MOAL, on considère `s` comme liste de caractères, donc strings.
2. Donner le graphe de flot de contrôle de cette fonction.
3. Donner le chemin le plus court permettant de satisfaire le critère « toutes les instructions ». On notera ce chemin ch1.
4. En détaillant l'exécution symbolique, déterminer la condition de chemin de ch1. S'il est faisable, donner un test pour ce chemin (c'est-à-dire des valeurs concrètes pour les arguments et le résultat attendu), sinon expliquer pourquoi il n'est pas faisable.
5. Donner le plus court chemin ch2 tel que l'ensemble $\{\text{ch1}, \text{ch2}\}$ satisfasse le critère « toutes les transitions ».
6. En détaillant l'exécution symbolique, déterminer la condition de chemin de ch2. S'il est faisable, donner un test pour ce chemin, sinon expliquer pourquoi il n'est pas faisable.
7. On considère tous les chemins qui passent deux fois par la boucle tel que, au premier tour, la condition du premier *if* est vraie et la condition du deuxième *if* est fausse. Sans détailler l'exécution symbolique, donner la condition de chemin pour chacun de ces chemins, puis donner un test s'il est faisable. S'il ne l'est pas, expliquer pourquoi.
8. Expliquer en quelques lignes la forme que doit avoir un chemin du graphe de cette fonction pour être faisable.

Exercice 2

La fonction `palindrome` prend en paramètre un tableau de caractères s et sa taille n, et renvoie *true* si et seulement si le mot s est un palindrome (on peut le lire indifféremment de gauche à droite et de droite à gauche).

On considère la réalisation suivante de la fonction, où `n div 2` est la division entière de `n` par 2.

```
boolean palindrome(char[] s, int n) {
  int i = 0;
  boolean b = true;
  while (i < (n div 2) && b)  {
    if (s[i] == s[n-i-1]) {
      i = i+1;
    } else {
      b = false;
    }
  }
  return b;
}
```

1. Donner une spécification formelle de cette fonction sous la forme de pré et post-conditions.
2. Donner le graphe de flot de contrôle de cette fonction.
3. Donner un ensemble de (petits) chemins permettant de satisfaire le critère « toutes les transitions ».
4. Calculer par exécution symbolique les conditions de chemin associées à ces chemins. Pour chaque chemin, s'il est faisable, donner un test pour ce chemin (c'est-à-dire des valeurs concrètes pour les arguments et le résultat attendu), sinon expliquer pourquoi il n'est pas faisable.
5. Les tests de la question précédente permettent-ils de satisfaire le critère « toutes les conditions multiples » ? Compléter par le(s) test(s) nécessaire(s).
6. Sans faire l'exécution symbolique, donner des tests pour couvrir tous les chemins passant au plus 2 fois dans la boucle.
7. Donner une condition suffisante pour qu'un chemin de ce graphe de contrôle soit infaisable.