LOG3430 - Méthodes de test et de validation du logiciel

Laboratoire 2

Fonctions logiques

Département de génie informatique et de génie logiciel
École Polytechnique de Montréal

Hiver 2022

1 Introduction

Dans ce laboratoire, vous allez implémenter deux fonctions logiques pour classifier les messages. Ensuite, vous allez créer des jeux de tests en fonction de plusieurs critères de couverture.

2 Objectifs

Les objectifs généraux de ce laboratoire sont :

1. Pratiquer la conception des tests pour les fonctions logiques.

3 Fonctions logiques

Une fonction logique prend en entrée un nombre n de clauses et sort une seule sortie booléenne. Les expressions logiques peuvent être dérivées de divers artefacts logiciels, y compris le code, les spécifications et les automates finis. Dans le cas de notre système de filtrage de Spam, nous allons utiliser une fonction logique pour classifier les messages.

Pour la classification, nous nous intéressons aux questions suivantes :

— Le message courant est-il un Spam ?
— L'utilisateur a-t-il un long historique de communication ?
— Quel est le niveau de confiance de l'utilisateur ?
— Quel est le niveau de confiance du groupe de l'utilisateur ?

En considérant ces questions on peut proposer la fonction logique suivante. Le message va être classifié comme spam, si :

— (le message courant est un spam)
ET
— (
  — (
    — (l'historique de communication est inférieur à 20 jours)
    ET
    — (le niveau de confiance de l'utilisateur est inférieur à 50)
  —)
  OU
  — (
    — (le niveau de confiance de l'utilisateur est inférieur à 50)
    ET
    — (le niveau de confiance du groupe de l'utilisateur est inférieur à 50)
  —)
—)

On définit les variables suivantes :
— S = vrai, si le message est classifié comme Spam.
— P : vrai, si le message courant est classifié comme Spam selon le vocabulaire créé.
— H : vrai, si le temps entre la date du premier message vu et la date du dernier message vu est est inférieur à 20 jours.
— U : vrai, si le niveau de Trust de l'utilisateur est inférieur à 50.
— G : vrai, si le niveau de Trust du groupe de l'utilisateur est **supérieur ou égal** à 50.

Finalement, l'équation logique pour classifier les messages est :

$$S = P * ((H * U) + (U * \neg G)) \tag{1}$$

En forme DNF, on va utiliser l'équation :

$$S = P * H * U + P * U * \neg G \tag{2}$$

3.1 Récapitulation des notions vues en classe

Votre tâche sera d'écrire les jeux des tests pour les fonctions logiques (1) et (2) pour différents critères de couvertures.

Pour les tests ACC, notre but est de construire un jeu de tests en fonction des clauses majeures et mineures. Dans un prédicat ou une fonction logique, les clauses sont les variables de la fonction. Tour à tour, chaque clause deviendra la clause majeure et il faudra trouver ses clauses mineures et leurs valeurs.

$$S = A(B + C) \tag{3}$$

Ici, nous avons 3 clauses : A, B, C. Pour chaque clause majeure, les clauses mineures correspondent à un ensemble de clauses. Lorsque ces clauses mineures ont des valeurs précises, un changement de la valeur de la clause majeure change la valeur du prédicat. Revenons à notre exemple pour clarifier les choses :

Commençons par A qui est la clause majeure. Si B = Vrai et C = Vrai, la valeur de A détermine le prédicat. Autrement dit, si A est Vrai, S est Vrai et si A est Faux, S est Faux. Les clauses B et C sont donc les clauses mineures de A.

Au tour de B maintenant ! Si A = Vrai et C = Faux, la valeur de B détermine le prédicat. En effet, si B est Vrai, S est Vrai et si B est Faux, S est Faux. Les clauses A et C sont donc les clauses mineures de B !

Un raisonnement similaire peut être appliqué à C pour trouver ses clauses mineures : A et B.

Dans un test ACC, on s'intéresse aux valeurs des clauses majeures et mineures. Le principe de base est que, pour chaque clause majeure, il y ait un test pour lequel la clause majeure est Fausse et un test pour lequel elle est Vraie. De plus, pour ces deux tests, il faut aussi choisir les valeurs des clauses mineures pour qu'un changement de la clause majeure modifie la sortie de la fonction logique. Ainsi, on se retrouve avec deux tests pour chaque clause majeure. La grande distinction entre les trois type d'ACC, c'est si les clauses mineures doivent garder les mêmes valeurs dans les deux tests. Le test GACC ne donne aucune restriction, le test

RACC dit que les clauses mineures doivent avoir les mêmes valeurs et le critère CACC donne comme seul restriction que dans un des deux tests, le prédicat soit Vrai, et dans l'autre il soit Faux. Veuillez vous référer aux notes de cours pour plus de détails.

Les critères de couverture des clauses actives (ACC) visent à s'assurer que les clauses majeures affectent leurs prédicats. Un critère complémentaire à ACC, l'ICC vérifie les modifications d'une clause majeure qui ne devrait pas affecter le prédicat. Ici, on choisira d'autres valeurs pour les clauses mineures. Le critère de choix sera que lorsque la clause majeure est modifiée, le prédicat reste le même. Au lieu d'avoir deux tests, les tests ICC requièrent 4 tests pour chaque clause majeure :

1. Clause majeure est vraie et prédicat est vrai
2. Clause majeure est vraie et prédicat est faux
3. Clause majeure est fausse et prédicat est vrai
4. Clause majeure est fausse et prédicat est faux

Dans le cas des tests GICC, on ne fait aucune restrictions sur les clauses mineures. Pour RICC, les clauses mineures doivent être identiques pour une même valeur du prédicat (1. = 3. et 2. = 4.).

Certains critères de couverture sont basés sur des prédicats écrits dans la forme DNF. Notamment les critères IC, PIC et VNS.

3.2 Les tâches

3.2.1 Implémentation

Vous devez d'abord implémenter les deux fonctions qui retournent S. Ces fonctions prennent en paramètre :

— Pour P : un booléen,
— Pour H : un entier indiquant la durée de l'historique
— Pour U : un entier indiquant le niveau de confiance de l'utilisateur
— Pour G : un entier indiquant le niveau de confiance du groupe

Vous devez ensuite **implémenter par un code** la manière de trouver les critères *CACC, GICC et IC*. Vos fonctions doivent produire un jeu de tests pour chaque critère. Il n'est pas nécessaire d'afficher tous les jeux de tests possibles. **Vous devez produire deux fichiers de code** :

— *criteres.py* : 2 fonctions S + 3 fonctions de critères
— *main.py/sh* : appels et commandes pour afficher à l'écran un jeu de tests pour chacun des 3 critères

3.2.2 Jeux de tests

Un jeu de tests pour chacun des 8 critères doit être présenté dans le rapport. Vous devez d'abord montrer la table de vérité pour le prédicat (celle-ci sera la même pour tous les critères).

— Pour le prédicat (1) proposez les jeux de tests qui satisfont les critères GACC, CACC et RACC.

    - GACC : Faites une démarche complète comme vu en classe
    - CACC : Reportez le jeu de tests fourni par votre code et montrez que le jeu de tests respecte bien le critère.
    - RACC : Faites une démarche complète comme vu en classe
- Pour le prédicat (1) proposez les jeux de tests qui satisfont les critères GICC et RICC.
    - GICC : Reportez le jeu de tests fourni par votre code et montrez que le jeu de tests respecte bien le critère.
    - RICC : Faites une démarche complète comme vu en classe
- Pour le prédicat (2) proposez les jeux de tests qui satisfont les critères IC, PIC et VNS.
    - IC : Reportez le jeu de tests fourni par votre code et montrez que le jeu de tests respecte bien le critère.
    - PIC : Faites une démarche complète comme vu en classe
    - VNS : Faites une démarche complète comme vu en classe

4 Remarques

- Vous n'avez donc que deux fichiers de code à produire : *criteres.py* et *main.py/sh*, le reste se passe dans le rapport.
- Pour tous les critères, expliquez en quoi vos jeux de tests couvrent ces critères (comme vu en cours).

5 Livrables attendus

Les livrables suivants sont attendus :

- Un rapport pour le laboratoire : 12 points. Ce rapport doit contenir :
    - Jeux de tests demandés pour les fonctions logiques. Il faut inclure toutes les étapes intermédiaires pour la création des tests (consulter les exemples donnés en cours).
- Les fichiers *criteres.py* et *main.py/sh* : 8 points

Le tout à remettre dans une seule archive **zip** avec le titre matricule1_matricule2_matricule3_lab1.zip sur Moodle. Seulement une personne par équipe doit remettre le travail.

Le rapport doit contenir le titre et numéro du laboratoire, les noms et matricules des coéquipiers ainsi que le numéro du groupe.

6 Information importante

1. Consultez le site Moodle du cours pour la date et l'heure limites de remise des fichiers (deux semaines après la première séance du laboratoire).
2. Un retard de [0,24h] sera pénalisé de 10%, de [24h, 48h] de 20% et de plus de 48h de 50%.

3. Aucun plagiat n'est toléré. Vous pouvez uniquement soumettre le travail fait par les membres de votre équipe.