Université Joseph Fourier — UFR IM2AG
Master 2 Génie Informatique — année 2013-2014

JML - TD 5

Yves LEDRU

Septembre 2013

Les fichiers décrits dans ce TP sont déposés sur le site Moodle du cours:
`http://imag-moodle.e.ujf-grenoble.fr/course/view.php?id=93`

Test combinatoire

Le test combinatoire est une technique de génération de tests qui produit toutes les combinaisons de valeurs définies pour un test donné. Pour ce TD, nous utiliserons l'outil Tobias, disponible en ligne à l'adresse `http://tobias.liglab.fr`.

Vous trouverez sur ce site sous l'entrée "Documentation/Tobias Notes" le document UsingTobiasOnLine.pdf qui décrit les principales fonctionnalités de l'outil. Veuillez noter que la taille des suites de test produites par le site web est limitée (environ deux mille tests dans les exemples vus lors de ce TD).

Un exemple

Reprenons l'étude de cas du stockage d'explosifs. Il est conseillé de créer deux répertoires. L'un comprendra la version originale de `Explosives.java` et l'autre comprendra la version complétée avec des pré-conditions.

Le fichier `TestExplosives00.txt` (disponible dans le moodle), définit un groupe de deux bâtiments et un groupe de trois produits. Il définit également le groupe `storeProd` qui réalise l'opération d'assignation d'un produit à un bâtiment pour toutes les combinaisons des deux bâtiments et des trois produits.

Le groupe `TestAssign001` instancie la classe `Explosives`, définit une incompatibilité entre les deux premiers produits, et effectue un ou deux appels au groupe `storeProd`. Le groupe `TestAssign001` est marqué `[us=true]` ce qui signifie que c'est ce groupe qui donnera lieu à une suite de test.

```
group TestAssign001[us=true] {
        Explosives e = new Explosives();
        e.add_incomp("Prod1","Prod2");
        @storeProd{1,2};
}

group storeProd{
  e.add_assign(@bats,@prods);
}

group bats {
values = ["Bat1", "Bat2"];
}

group prods {
values = ["Prod1", "Prod2", "Prod3" ];
}
```

Question 1 - Dépliage du groupe TestAssign001

Utilisez le site web de Tobias pour déplier le groupe `TestAssign001`. Choisissez l'option "JUnit4/JML". Combien de tests sont produits par ce dépliage? Récupérez le résultat sur votre messagerie, compilez le fichier de test et exécutez-le sur la classe originale `Explosives.java` (c'est-à-dire sans ajout de préconditions). Quelle est la propriété invariante qui est violée par certains de ces tests?

Remplacez
`org.junit.runner.JUnitCore.runClasses(TS_TestAssign001.class);`
par
`org.junit.runner.JUnitCore.main("TS_TestAssign001");`
pour des messages d'erreur plus complets.

Essayez ensuite d'exécuter cette suite de test sur la version avec préconditions de `Explosives.java` et constatez que tous les tests réussissent ou sont inconclusifs.

Question 2 - Test des propriétés 3 et 4

Construisez un fichier d'entrée pour Tobias qui teste la propriété 3 (les produits commencent par le String `"Prod"`). Définissez un groupe de noms de produits, qui comprenne des noms valides et des noms invalides et testez l'opération `add_incomp` avec diverses combinaisons de ces noms. Générez une suite de test et essayez-la sur les deux versions de la classe `Explosives.java`. Vérifiez que certains tests violent la propriété 3.

Faites de même avec la propriété 4.

Question 3 - Itération

Construisez un fichier d'entrée pour Tobias qui teste la propriété 1 (nombre maximum d'incompatibilités), en utilisant la construction d'itération de Tobias. Prenez garde à l'explosion combinatoire : répéter 50 fois une opération qui fait appel à un groupe génère un très grand nombre de tests! Générez votre fichier et utilisez le pour tester les deux versions de la classe `Explosives.java`.

Faites de même pour la propriété 2.

Question 4 - Autres propriétés

Construisez un fichier d'entrée pour Tobias qui teste la propriété 5 (deux produits identiques ne sont pas incompatibles). Votre fichier comprendra des cas qui invalident la propriété, et des cas normaux qui ne l'invalident pas. Veillez à ce que vos tests ne concernent que la propriété 5. Utilisez votre suite de test sur les deux versions de la classe `Explosives.java`.

Construisez un fichier d'entrée pour Tobias qui teste la propriété 7 (deux produits incompatibles ne sont pas dans le même bâtiment). Votre fichier comprendra des cas qui invalident la propriété, et des cas normaux qui ne l'invalident pas. Veillez à ce que vos tests ne concernent que la propriété 7. Utilisez votre suite de test sur les deux versions de la classe `Explosives.java`.

Question 5 - Mélange de propriétés

Construisez une suite de test plus complexe dont les tests déclarent quelques incompatibilités puis stockent divers produits dans les bâtiments.

Utilisez votre suite de test pour tester les deux versions de la classe `Explosives.java`.

Vous constaterez que (a) il est facile de générer un très (trop) grand nombre de tests, et (b) qu'il est difficile de faire le tri dans les messages d'erreur et de vérifier quelles propriétés ont effectivement été violées. Cette expérience suggère de construire des suites de test indépendantes pour chaque propriété visée. Par contre, vous constaterez que tous les fichiers produits pendant ce TD permettent de tester de façon massive la classe `Explosives.java` et augmentent la confiance que l'on peut avoir dans l'identification correcte des préconditions.