ANNEXE A : PLAN DE TESTS
Projet de systèmes intergiciels

Édouard Lumet <edouard.lumet@etu.enseeiht.fr>
Bastien Pelissier <bastien.pelissier@etu.enseeiht.fr>

17 avril 2019

Description sommaire

L'idée est de couvrir, à travers de nouveaux tests, l'ensemble des spécifications et des contraintes des primitives manipulant les tuples. Il faut donc tester le verrouillage, les opérations bloquantes ou non et bien sûr si les résultats attendus sont corrects comme l'écriture du tuple (4,5) dans le *tuplespace* à la suite d'un *write* de ce tuple.

A noter que par défaut, les tests se déroulent sur la version en mémoire partagée. Pour les exécuter sur la version client-serveur, il faut modifier l'attribut Linda.

Tests rédigés

CustomTest1

Ce test a pour but de tester si la primitive *take* est bien une opération bloquante et qu'elle efface bien le tuple de l'espace. Pour cela, on crée 3 *threads* démarrant les uns après les autres. Dans le premier, on effectue deux opérations *take* sur deux motifs différents. Dans le deuxième *thread*, on ajoute un tuple répondant au deuxième motif et non au premier motif. On doit s'attendre à ce que la première trace soit celle du premier *write* puis du second avant d'obtenir le retour du premier *take* ainsi que du second (dans cet ordre exact). De plus, la dernière trace de debug doit présenter le *tuplespace* vide.

CustomTest2

Ce test est dans la même philosophie que le précédent en vérifiant que la primitive *read* est également bloquante. La seule différence étant que le *tuplespace* ne doit pas être vide à la fin et doit toujours contenir les deux tuples ajoutés lors du test.

CustomTest3

Ce troisième test vérifie que la primitive *readAll* n'est pas bloquante. Lorsque qu'un *readAll* ne trouve pas de tuple correspondant au motif donné, on doit donc pouvoir poursuivre le *thread* sans blocage. Pour vérifier cela, le test rédigé consiste en l'écriture d'un motif 1 dans le *tuplespace*. Ensuite, un autre *thread* démarre et exécute deux *readAll* : le premier ne correspond pas au motif 1 tandis que l'autre correspond. On s'attend alors à voir apparaître une trace vide pour le premier et un retour immédiatement après pour le second, avant de passer aux *threads* suivants.

CustomTest4

Ce test est identique au précédent mais pour la primitive *takeAll*. On doit seulement s'attendre à ce que le *tuplespace* ne contienne plus les tuples retournés.

CustomTest5

À travers ce test on vérifie que la primitive *readAll* renvoie tous les tuples correspondant à un motif donné. Pour cela, on commence par ajouter plusieurs tuples dont quelqu'uns ayant un motif précis commun puis on effectue le *readAll*. Une collection de tuples doit être retournée et ceux-ci doivent toujours être présents dans le *tuplespace*.

CustomTest6

Ce test est le même que précédemment pour la primitive *takeAll*. La seule différence en termes d'attentes est que les tuples retournés ne doivent plus être présents dans l'espace partagé.

CustomTest7

Ce test est dans la même philosophie que le test 3 pour la primitive *tryRead*. Cependant, on doit s'attendre à ce que le retour ne contienne qu'un tuple correspondant au motif donné ou la valeur *null* en cas d'absence du tuple. L'espace partagé reste inchangé.

CustomTest8

Même chose que précédemment pour la primitive *tryTake*. En revanche, le tuple retourné ne doit plus être présent dans l'espace partagé.