LOG8470

Méthodes formelles en fiabilité et en sécurité

TP 3 : *Model checking*

Session	Automne 2021
Pondération	10 % de la note finale
Taille des équipes	2 étudiants
Date de remise	9 novembre (23 h 55 au plus tard)
Directives particulières	Soumission des fichiers Promela par moodle dans un dossier zip sous la forme **tp3-matricule1_matricule2.zip**.
	Toute soumission du livrable en retard est pénalisée à raison de 10 % par jour de retard.
Moyen de communication	- Serveur discord - Courriel : basil.capar@polymtl.ca

1. Notion du cours à connaître

- Automate non-déterministe
- Logique temporelle linéaire (LTL)
- Propriétés de sûreté et de vivacité

2. Objectif

Ce laboratoire est séparé en 3 exercices. L'objectif du premier est de représenter la description de l'énoncé en un automate non-déterministe. Le second exercice consiste à implémenter l'automate non-déterministe en Promela. Le dernier exercice a pour objectif de rédiger la LTL du système.

3. Description du problème

Comme énoncé dans le TP1, l'école Polytechnique développe une application qui permet aux étudiants ainsi qu'au personnel de l'école de commander de la nourriture. En effet, certaines opérations devront être possible à l'utilisateur. Bien entendu, certaines étapes devront être réalisées avant d'autres.

Lorsque l'utilisateur ouvre l'application, il devra s'authentifier à l'aide d'un nom d'utilisateur ainsi qu'un mot de passe. Tant et aussi longtemps que le nom d'utilisateur est invalide, le système demandera à l'utilisateur d'entrer un nom d'utilisateur valide et le message *WrongUsername* sera retourné. Ce choix est fait d'une façon non-déterministe. Une fois que le nom d'utilisateur est

valide, le message *ValidUsername* sera retourné et l'utilisateur devra entrer son mot de passe. Il possède 3 essais avant de bloquer son compte. Encore une fois, la vérification de la validité du mot de passe sera effectuée d'une façon non-déterministe et le message ValidPassword sera retourné à l'utilisateur lorsque le mot de passe est valide et WrongPassword lorsque c'est le contraire.

Une fois que l'utilisateur est connecté à son compte, il peut effectuer différentes opérations. Ces dernières sont réalisées d'une façon non-déterministe. L'utilisateur peut :

- Ajouter 5 $ dans son compte
- Retirer tout l'argent contenu dans son compte
- Acheter le plat du jour à 10 $
- Se déconnecter

Le message qui sera envoyé au serveur lorsque l'utilisateur ajoute de l'argent dans son compte est *AddFunds*. Quant au serveur, il retournera un message *FundsAdded*.

Le message qui sera envoyé lorsque l'utilisateur retire tout l'argent de son compte est *WithdrawFunds* et le serveur retournera le message *FundsWithdrew* lorsque l'opération s'effectue avec succès. Dans le cas contraire, le serveur retournera le message *NoMoreFunds.*

Lorsqu'il n'y a pas suffisamment d'argent dans le compte de crédit de l'utilisateur, le système devra retourner un message du type *NotEnoughFunds*. S'il y a assez d'argent, la transaction peut être complétée et le message *BoughtMeal* sera retourné.

Le message envoyé lorsque l'utilisateur se déconnecte est *Logout* et le système devra retourner un message *LoggedOut*.

Il est très important de noter que l'utilisateur utilise l'application pour la première fois. Ainsi, il n'y a pas d'argent dans le compte de sa carte de crédit d'école.

Il faut savoir que le système est séparé en 3 processus : *Client, Server* et *Database*. Bien entendu, toutes les vérifications seront réalisées dans le serveur et les informations seront conservées dans la base de données.

Aussi, l'application devra rouler à l'infini. En effet, même si le compte est bloqué, l'application devra s'assurer de se réinitialiser afin de repartir une séquence d'authentification.

Sachez que vous êtes libres d'ajouter d'autres types de messages afin de rendre le système plus structuré et réaliste. Les types de messages qui vous sont donnés sont les plus importants.

4. Automate non-déterministe **(3 pts)**

Représentez le système décrit ci-dessus avec un automate non-déterministe.

5. Implémentation **(2 pts)**

Vous devez maintenant implémenter l'automate non déterministe que vous avez conçu à l'exercice précédent. Assurez-vous de faire des affichages dans la console pour voir la quantité d'argent dans le compte lors d'un ajout ou d'un retrait, par exemple.

6. LTL **(5 pts)**

Traduisez-en LTL les propriétés suivantes. Pour chacune d'entre elles, précisez si c'est une propriété de *safety*, de *liveness*, *invariant* ou aucun des trois.

a) Si le mot de passe est invalide et le nombre de tentatives est supérieur à 3, le compte finira par être verrouillé par le système.
b) Un client ne peut pas s'endetter.
c) Un client ne peut pas commander tant qu'il n'est pas connecté.
d) Le client fini toujours par retirer de l'argent.
e) Un client peut toujours déposer de l'argent dans son compte.

Assurez-vous que votre système prend en considération ces propriétés LTL. Prenez le temps de vérifier ces propriétés avec Spin. Vous pouvez consulter l'annexe pour connaître les commandes à exécuter. Il est important de savoir que si spin génère un fichier qui s'appelle **.pml.trail*, cela signifie que votre propriété n'est pas vérifiée.

7. Contenu du rapport

Votre rapport doit contenir une page couverture, une image de votre automate, vos propriétés LTL. Au niveau de votre automate, prenez le temps de le décrire. Quant aux propriétés LTL, prenez le temps de décrire les variables que vous utilisez, de dire s'il y a un fichier **.pml.trail* qui est généré ainsi que de dire si les propriétés représentent la *safety*, la *liveness* ou l'invariance.

8. Modalité de remise

Vous devez remettre un fichier *zip* qui contient le fichier *.pml* ainsi qu'un rapport. Le rapport devra être nommé comme suit : tp3-matricule1_matricule2.pdf. Pour le fichier promela, il devra être nommé comme suit : tp3-matricule1_matricule2.pml.

9. Critères d'évaluation

Qualité du code	2 points
Qualité de la LTL	3 points
Propriétés LTL fonctionnelles	2 points
Qualité de l'automate.	3 points
Code non exécutable	-2 points
Mauvais format de remise	-1 point
Retard	**-10 % par jour de retard**

Annexe 1

Voici les commandes à exécuter qui vous permettront de vérifier vos propriétés LTL.

```
1- spin -a foo.pml
2- gcc -o pan pan.c
3- ./pan -a -N p1
4- spin -t -k foo.pml.trail -p foo.pml
```

```
//p1 représente le nom de la propriété LTL qui sera dans le code promela
// La dernière commande permet d'éxécuter le fichier *.pml.trail pour voir comment
l'application se comporte avant d'arriver à un état d'erreur
```