LOG8470: Vérification de la fiabilité et sécurité
Travail Pratique 1: Model Checking iSpin

Présenté à:
John Mullins

Par:
Kenny Nguyen (1794914)
Yujia Ding (1801923)
Ka Hin Kevin Chan (1802812)

École Polytechnique de Montréal
Département de génie Logiciel
Le 1er novembre 2018

Introduction

Dans le cadre du premier travail pratique dans le cours de fiabilité et sécurité, l'objectif de ce travail est de modéliser des systèmes avioniques simples et d'utiliser iSPIN pour en vérifier les propriétés. On propose de modéliser une version simplifiée des modules IMA connectés entre eux via le réseau qui fait la gestion des systèmes embarqués des avions modernes.

Ainsi, en effectuant plusieurs exercices en langage Promela avec le logiciel iSPIN, on peut modéliser les modules IMA de la logique de commande des avions ainsi que la communication réseau qui les relient. De cette façon, le *fuel control system (FCS), l'entertainment control system* (ECS) et le *landing gear system* (LGS) ne créent pas d'interférences entre-eux lors de l'affichage de leurs informations sur le *multi-functional display* (MFD). Cela est essentiel pour assurer la sécurité et le bien être des passagers.

Présentation des travaux

5. Les modèles IMA

2. (a) Les deux fonctions ne peuvent pas accéder aux ressources du module en même temps ;

La propriété est une propriété de safety. Empêcher les deux fonctions d'accéder aux mêmes ressources en même temps est reliée à l'empêchement d'une situation néfaste.

$\square(\neg a1 \vee \neg a2)$

Où a1 et a2 représentent les processus actifs 1 et 2.

(b) Une fonction active (non en panne) doit pouvoir continuellement accéder aux ressources du module ;

Donner accès aux ressources de manière continue crée un environnement favorisant l'avènement d'un comportement souhaite, donc (b) est une propriété de liveness.

$\square(\diamond a1 \vee p2)$
$\square(\diamond a2 \vee p1)$

Où p1 et p2 représentent les processus en panne 1 et 2.

(c) L'une des propriétés importantes de l'approche IMA est d'assurer la non-propagation d'une faille d'une fonction à l'autre. En d'autres, une faille dans une fonction ne doit pas modifier le comportement normal des autres fonctions du module. Traduisez ce requis en LTL.

En empêchant les failles d'une fonction de se propager aux autres fonctions, on diminue les chances qu'un comportement non-souhaité se manifeste, donc c'est une propriété de safety.

$p2 \rightarrow \square(\diamond a1 \vee p1)$

.

3. Specifez et verifiez (si possible) ces proprietes dans Spin. Si une propriete est fausse, associez
une trace d'execution qui l'invalide et expliquez le probleme.

regleA (pas d'erreur) : State-vector 64 byte, depth reached 53, errors: 0
[]((!actif[0]) || (!actif[1]))

regleB1 (pas d'erreur) : State-vector 64 byte, depth reached 53, errors: 0
[](<>actif[0] || panne[1])

regleB2 (pas d'erreur) : State-vector 64 byte, depth reached 53, errors: 0
[](<>actif[1] || panne[0])

regleC (pas d'erreur) : State-vector 36 byte, depth reached 0, errors: 0
panne[1] -> [](<>actif[0] || panne[0])

4. Proposez une amélioration de votre modèle de la gestion d'accès aux ressources du module qui prend en compte la possibilité d'une faille de fonctionnement. Vérifiez avec Spin que votre nouveau modèle assure ces requis.

Un timeout a été ajouté à la **ligne 79 de tp1_5.pml** pour tenir compte de la possibilité d'une faille de fonctionnement. Le timeout permet de sortir d'un état bloquant qui est provoqué par un break. Un break survient de manière non-déterministe et cause une panne. Le timeout reprend l'exécution lorsqu'un état est resté bloquant trop longtemps. Les sorties sont les mêmes qu'en 5.3.

6. La communication réseau

2. Traduisez en LTL les propriétés suivantes. Pour chacune d'elles, précisez si c'est une propriété de safety ou de liveness. (a) Tout message reçu par la destination est nécessairement produit par l'un des sources.

D = destination, E = buffer ECS, F = buffer FCS, L = buffer LGS, M = buffer MDS

$(! m \in D \ U \ ((m \in E) \vee (m \in F) \vee (m \in L)))$

La propriété 6a est une propriété de liveness. Elle vérifie que la source du message est correcte, puisqu'un message est envoyé par une fonction, dans des cas d'utilisation normaux.

(b) Un des requis importants des systèmes avioniques est que le réseau ne doit jamais perdre des messages de haut niveau de criticité. Exprimez le en LTL

Dans la théorie, on voit que la spécification pour empêcher la perte de message est exprimable en fonction du message et des canaux source et receveur :
$\Box[mc \in S.out \Rightarrow \Diamond(mc \in R.in)]$

En adaptant cette règle à notre cas, on se sert des canaux E, F et L comme sources et D comme receveur :

$\Box \ (\ (\ (mf \in E) \ (me \in F) \ (me \in L) \ ) \Rightarrow \Diamond \ (m \in D) \ )$

Cette règle est une propriété safety, car elle vise à empêcher une perte de message, qui est un scénario indésirable.

3. Implémentez et vérifiez (si possible) ces propriétés dans Spin (une vérification par type de messge). Si une propriété est fausse, associez une trace d'exécution qui l'invalide et expliquez le problème.

regleA (pas d'erreur) : State-vector 160 byte, depth reached 18, errors: 0
(! buff_destination?[m] U (buff_ECS??[m] || buff_FCS??[m] || buff_LGS??[m]))

regleB (pas d'erreur) : State-vector 160 byte, depth reached 176, errors: 0
[] ((buff_ECS?[m, niv] || buff_FCS?[m, niv] || buff_LGS?[m, niv]) -> <>
buff_destination??[m, niv])

4. Si la non perte des messages de haut niveau de criticité n'est pas assurée, modifiez votre ordonnanceur au niveau du commutateur (seulement) pour donner priorité aux envois de messages de haut niveau de criticité. Vérifiez votre modèle amélioré.

La non perte de message de criticité élevée est assurée à l'aide d'un timeout à la (**ligne 79 de tp1_6.pml**). Le timeout permet de quitter un état bloquant, qui survient lorsqu'un processus n'arrive pas à envoyer un message critique sur un des canaux. Le switch transmet les messages de criticité élevée en premier. À l'aide de sentinelles (**ligne 59 et 64 de tp1_6.pml**), la fonction switch envoie des messages d'ECS uniquement si aucun message de FCS ni de LGS n'est en attente.

5. (bonus) Le problème est-il résolu ? Sinon, pensez à ordonnancer les modules en donnant priorité ceux ayant que des applications de criticité elévée (ici l'envoi des messages par le CC de M 2). Notes : — Pensez à utiliser le timeout pour débloquer le système lorsque le cycle finit ; — Pensez à utiliser les canaux de capacité finie pour modéliser les tampons.

Le problème d'ordonnancement est résolu. Le timeout (**ligne 79 de tp1_6.pml**) définit un temps maximal après lequel le système quitte la fin de cycle. Le temps maximal est atteint lorsqu'on ne peut plus écrire sur les canaux. Nous avons utilisé des canaux de capacité finie (par exemple buffer_destination) afin de stocker les messages des fonctions sources. Ceci permet de simuler un envoi vers un module destination. En conséquence, lorsque la capacité maximale de buffer_destination (MSG_DEST) est atteinte, on atteint la fin de cycle et le timeout sort du cycle courant.

Difficultés rencontrées

Le langage Promela n'est pas évident ni à apprendre ni à maîtriser, à cause de la rareté de la documentation en ligne relié à ce langage. De plus, coder en cette langage n'est pas très intuitif, ajoutée à la difficulté d'apprendre les nouvelles formalités du langage. Cela a fait en sorte que les parties en Promela ont pris beaucoup plus de temps que prévu. Il faudrait peut-être des tutoriels moins basiques, ou carrément des tutoriels pratiques pour apprendre la langue avant de demander de coder avec.

De plus, nous avons disposé d'uniquement 3 heures en classe pour faire le TP, et pendant ce temps nous n'avions pas pu avancer le TP. Ceci est assez mal planifié, et le temps passé hors des heures de cours est supérieur à une charge de travail attendue. Il faudrait trouver une façon de balancer cela.

Conclusion

Ce travail a été utile pour approfondir nos capacités de vérification de la sécurité et de la fiabilité des systèmes. De plus, nous avons un exemple concret de l'utilisation de ces mesures vues en classe dans un aspect réel de la vie dans lequel la sécurité est un aspect essentiel, car la vie des passagers peut dépendre d'une faille du système.

De plus, nous avons pu nous familiariser avec Promela avec ces exercices. Cela sera un atout pour les prochains laboratoires si on a à utiliser cette langue encore dans les prochains travaux pratiques.