LOG 8470

Méthodes formelles en fiabilité et sécurité

TP1 : MODEL CHECKING - iSPIN

Session	Automne 2018
Pondération	5 % de la note finale
Taille des équipes	2 ou 3 étudiants
Date de remise du projet	1er novembre 2018 (23h55 au plus tard) pour tous les groupes
Directives particulières	Soumission du livrable par moodle uniquement (https ://moodle.polymtl.ca).
	Toute soumission du livrable en retard est pénalisée à raison de 10% par jour de retard.
Les questions sont les bienvenues et peuvent être envoyées à : Oswald Pichot (oswald.pichot@polymtl.ca), Sardaouna Hamadou (sardaouna.hamadou@gmail.com), John Mullins (john.mullins@polymtl.ca).	

1 Connaissances requises

— Notions de modélisation formelle des systèmes concurrents

— Logique Temporelle Linéaire (LTL).

2 Objectif

L'objectif de ce travail pratique est de modéliser de façon formelle des systèmes avioniques simples et d'utiliser iSPIN pour vérifier des propriétés de model checking. Dans ce but, on propose ici de modéliser une version simplifiée des modules IMA interconnectés via le réseau de communication AFDX qui gèrent les systèmes embarqués dans les avions civiles modernes tels que les Airbus A380 et les Boings B787.

3 Mise en situation

Les avions modernes, de par l'inclusion de fonctions avioniques de plus en plus nombreuses et complexes, requièrent des puissances de calcul et des ressources de communication de plus en plus importantes. Ainsi, les avionneurs tendent à construire des avions alliant des niveaux de performance et de sécurité accrus, tout en cherchant à en améliorer l'efficacité via notamment la réduction de la masse et de la consommation de l'appareil. Dans ce cadre, l'industrie avionique a majoritairement adopté l'approche dite Integrated Modular Architecture (IMA). En opposition à l'approche traditionnelle qui consistait en une architecture fédérée des systèmes avioniques, l'architecture IMA permet à des fonctions liées à différents sous-systèmes de l'appareil de partager la même plateforme de calcul communément appelée module. Ainsi, une plateforme peut héberger une multitude de fonctions avioniques présentant des niveaux de criticité différents. Cette approche vise à utiliser de manière plus efficace le hardware, le software, ainsi que les ressources de communication.

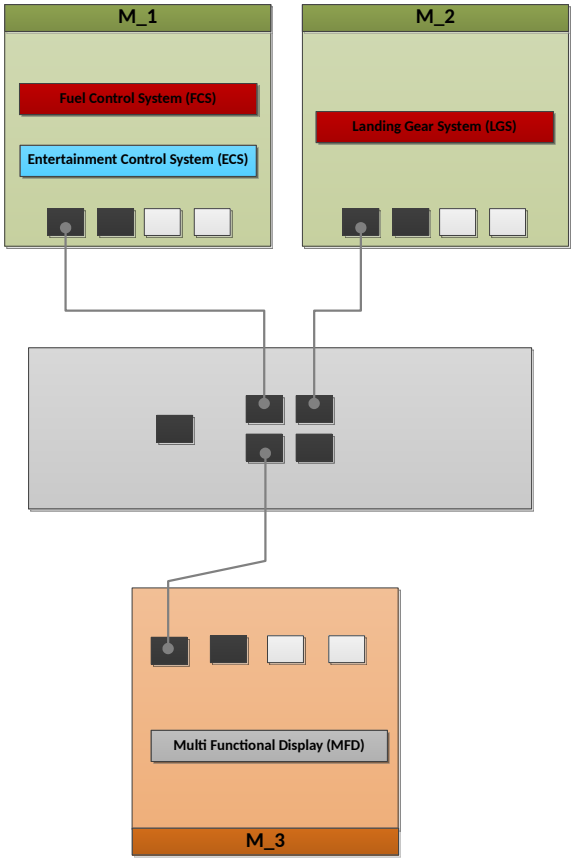

FIGURE 1 – Example d'un système avionique

4 Description

La figure 1 illustre un système avionique composé de trois modules interconnectés via un commutateur de réseau (switch). Sur chaque module est implémentée une ou plusieurs fonctions avioniques de différents niveaux de criticité. Par exemple sur cette figure, on a une fonction de la gestion de divertissement (ECS : Entertainment Control System) de faible niveau de criticité implantée sur le même module M_1 qu'une fonction de gestion de fuel (FCS : Fuel Control System) de niveau de criticité elévée. Le module M_2 contient une fonction de la gestion d'atterrissage (LGS : Landing Gear System) et le module M_3 implémente une fonction d'affichage dans le cockpit (MFD : Multi Functional Display).

5 Les modules IMA

Dans cette partie, nous nous penchons sur une modélisation simple d'un module IMA, avec une fonction de faible niveau de criticité et une de haut niveau de criticité. Pour faire simple, nous partons du principe que :

— Chaque fonction émet un signal unique ;

— Pour assurer le bon fonctionnement d'un module, nous devons garantir l'exclusion mutuelle (par un ordonnencement) pour l'accès aux ressources du module.

— Une fonction peut tomber en panne à tout moment.

— Une fonction qui n'est pas en panne, dépose continuellement le même message (signal) dans un canal de capacité illimitée, par example, pour signaler le bon fonctionnement du sous-système dont elle gère.

1. Modélisez en PROMELA un module ayant les deux types des fonctions en gérant l'ordonnancement (ici l'accès au canal) via une variable globale partagée en se passant à tour de rôle le droit d'accès. Chaque fonction choisit de manière non-déterministe s'il tombe en panne ou non. Un processus en panne ne peut pas remettre le tour à l'autre.
2. Traduisez en LTL les propriétés suivantes. Pour chacune d'elles, précisez si c'est une propriété de *safety* ou de *liveness*.
   (a) Les deux fonctions ne peuvent pas accéder aux ressources du module en même temps ;
   (b) Une fonction active (non en panne) doit pouvoir continuellement accéder aux ressources du module ;
   (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.
3. Spécifiez et vérifiez (si possible) ces propriétés dans Spin. Si une propriété est fausse, associez une trace d'exécution qui l'invalide et expliquez le problème.
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.

Notes :

— Pensez à utiliser le timeout pour débloquer le système lorsque tous les deux processus tombent en panne pour entrer dans un tat acceptant (bouclant sur lui-même) ;

— Une application ne tombe pas en panne en tant dans sa section critique. Pensez à utiliser l'atomicité.

6 La communication réseau

Les messages produits par des functions implémentés sur un module sont transmis fonctions les consommant sous forme de trames. Ces trames de différents niveaux de criticité transitent via le même réseau de communication. Un contrôleur de communication (CC) gère l'envoie des trames produits sur le même module. Le CC transmet les trames au commutateur réseau (switch) qui les fait suivre à leurs destinations suivant un ordronnancement qui lui est propre. Nous partons du principe que :

— Les systèmes avioniques sont cycliques.

— l'intérieur d'un cycle du système, chaque fonction s'exécute un nombre fini de fois ;

— Le commutateur ne fait que forwarder les messages ; Il reçoit les messages des CCs et stocke chaque type de message dans son propre tampon (buffer).

1. Modélisez le réseau en PROMELA (on ne modélise pas les modules IMA). On modélise juste les CC et le commutateur. On suppose 10 envois de messages ECS, 5 messages FCS et 3 messages LGS, tous transmis au MDS. Le MDS ne pouvant que s'exécuter au maximum 10 fois à l'intérieur d'un cycle. On suppose des ordonnanceurs équitables. Un ordonnanceur CC passe à tour de rôle l'envoi des différents types de message qu'il gère. Idem pour un commutateur avec ses tempons de reception.
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.

(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

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.
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.
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 cannaux de capacité finie pour modéliser les tampons.

7 Livrable

Le livrable attendu est constitué des sources et du rapport de laboratoire. Le livrable est une archive (ZIP ou RAR) dont le nom est formé des numéros de matricule des membres de l'équipe, séparés par un trait de soulignement (_). L'archive contiendra les fichiers suivants :

— le rapport au format PDF ;

— tout autre fichier source ou sortie créé par iSpin, jugé pertinent, et correctement référencé dans le rapport.

7.1 Rapport

Un rapport de laboratoire rédigé avec soin est requis à la soumission (format .pdf, maximum 8 pages). Sinon, votre travail ne sera pas corrigé (aussi bien le code source que l'exécutable). Le rapport doit obligatoirement inclure les éléments ou sections suivantes :

1. Page présentation : elle doit contenir le libellé du cours, le numéro et l'identification du TP, la date de remise, les matricules et noms des membres de l'équipe.
2. Introduction avec vos propres mots pour mettre en évidence le contexte et les objectifs du TP.
3. Présentation de vos travaux : une explication de votre solution.
4. Difficultés rencontrées lors de l'élaboration du TP et les éventuelles solutions apportées.
5. Conclusion : expliquez en quoi ce laboratoire vous a été utile, ce que vous avez appris, vos attentes par rapport au prochain laboratoire, etc.

Notez que vous ne devez pas mettre le code source dans le rapport.

7.2 Soumission du livrable

La soumission doit se faire uniquement par Moodle.

8 Évaluation

Éléments évalués	Points
Qualité du rapport : respect des exigences du rapport, qualité de la présentation des solutions	10%
Qualité du programme : commentaires, documentation, clarté, etc.	10%
Composants implémentés : respect des requis, logique de développement, etc.	
Q5.1	10%
Q5.2.a	5%
Q5.2.b	5%
Q5.2.c	5%
Q5.3	10%
Q5.4	10%
Q6.1	10%
Q6.2.a	5%
Q6.2.b	5%
Q6.3	10%
Q6.4	10%
Q6.7	15%
Q6.5 (bonus)	10%
Total de points	100%

9 Documentation

— Tutoriel de Promela, par Laure Petrucci.

— Site officiel de Spin : http ://spinroot.com/spin/whatispin.html.

— Les notes de cours.