Notes de Cours INF 3610

Olivier Sirois

2018-10-10

Contents

Chapter 1

Chapitre 1 - Introduction

On peu définir la vérification formelle comme étant l'utilisation de techniques théorique pour prouver le fonctionnement d'un système. Selon Sommerville, la vérification est la réponse à la question:
Are we building the product right

Pour etre vérifié, un système doit satisfaire:

- Vérification formele complète : exploration de façon exhaustive de tous les états possible du système
- Vérification formelle incomplète : vérification des bornes

INSÉRÉ PHOTO PROCESSUS DE DÉVELOPPEMENT
On vérifie formellement pour renforcer le processus de développement. Les techniques de vérification peuvent s'appliquer à différent niveau, car on regarde seulement le fonctionne du module sous test, pas tous le système.

INSÉRÉ PHOTO DES TESTS (MICROSCOPE)

On voit sur la photo la différence entre les deux méthodes de vérification, une est applicable à un gros système vs. l'autre qui ne peut être fait que sur des éléments de petites tailles.

Évidemment, certains accidents notables peuvent être attribuer à un manque de vérification. sa peut aller de AT&T à la NASA. (problème d'inversion).

Lorsqu'on fait affaire dans des systèmes critiques, soit:

- Transport (avionique, ferroviaire, spatial)
- Energie (nuclearire...)
- Medical (dosage radioactif)

En raison de la criticalités de ces applications, certains documents ont été mis en place pour la production de logiciel pour les applications aéronautiques.

Les normes ED-12C et DO-178C précisent ces contraintes.

1.0.1 Comment vérifier formellement

- Modéliser
- Spécifier
- Prouver

INSÉRÉ PHOTOS SLIDE 22

1.0.2 Comment spécifier un comportement

Le but de la modélisation est d'expirmer, au moyen d'un formalisme la manière dont le système se comporte. Le formalisme doit être assez expressif, repose sur une sémantique rigoureuse et doit offrire des possibilités d'analyse systématique. On peut résumé sa en une représentation simplifiée d'un système.

Dans le cas des systèmes temps réelles, le fonctionnement est assujetti à l'évolution dynamique de l'environnement et la réaction aux stimuli est soumise à des contraintes temporelles. c-a-d, il faut que la réponse à l'environnement arrive dans un temps spécifier. Plusieurs modèle formels ont été développés puis adaptés aux systèmes temps réel. On peut les distingué par la séquentialité (concurrence), le non-déterminisme, la synchronisation, la communication, la compositionnalité, les contraintes temporelles. On a même certains language de spécifications fait pour pouvoir vérifier des sytèmes.

les classe de modèles proposé dans la litérature sont :

- langage de prog..

INSÉRÉ SLIDE 27

en générales, les modèles doivent spécifier les:

- actions
- événement,
- contraintes,
- conditions d'activation,
- situations anormale et

- les états significatifs

langage de prog -¿ séparer en deux partie:

- classique, JAVA, C, C++
- ceux basé sur un modele de transition, RT-LOTOS, PROMELA, ESTEREL, LUSTRE

1.0.3 JAVA

Java pathfinder, vérificateur (model-checker) pour la NASA. (babelfish).

INSÉRÉ PHOTO SLIDE 32

1.0.4 LLBMC - low level bounded model checker

travail sur un code bas niveau (assembleur?). il fait du bounded model-checking dépendament du domaine spécifier.

INSÉRÉ SLIDE 34

1.0.5 Promela, model-checker SPIN

Promela (protocol/process meta language) est utilisé par le model checker SPIN. cré dynamiquement des processus concurrents, la communication entre processus via des variables partagées et des canaux de messages

INSÉRÉ SLIDE 36

1.0.6 LUSTRE, (boite a outils SCADE)

language synchrone a flots de donnees. Sa permet d'exprimer un systeme sour forme d'équations qui définissent l'évolution des valeurs de ses varables

INSÉRÉS SLIDE 38

on peut spécifier un programme aussi comme sur la slide 39

1.0.7 Modele a base de transition

On simule une sorte de state machine mealy, ou on modélise les états discrèt par états ainsi que leur transitions par rapport à leurs actions

Ces modèles sont en générales graphique:

- Les Statecharts
- Les réseaux de Petri
- Les automates

Les automates offrent un bon compromis entre la puissance de modélisation et la complexité de vérification

INSÉRÉ SLIDE 44, 45, 46

1.0.8 Modeles Logiques

INSÉRÉ SLIDE 48

1.0.9 Modèles Algébrique

Description du comportement d'un système à l'aide d'une description algébrique. On combine des opérateurs et des actions.

INSÉRÉS SLIDE 50

1.0.10 Comment spécifier une proprité

On peut utiliser la logique (propositionelle, prédicat), pour spécifier des chose. Sauf que c'étais insuffisant pour décrire des systèmes à comportement temporels. On a alors étendu ces notions à logique temporelle pour pouvoir vérifier formellement des systèmes de natures temporelle.

INSÉRÉS SLIDE 54

1.0.11 Logique temporelle linéaire

LTL permet d'expliquer comme la logique évolue dans le temps

1.0.12 Logique temporelle arborescente (CTL)

Cette logique permet de quantifier les chemins d'exécutions

INSÉRÉS SLIDE 56

1.0.13 Vérification effective

La vérification est dite décidable pour un ensemble de modèles M et une classe de propriétés P si et seulement si il existe un programme qui prend en trné un modèle quelconque de M et une propriété quelconque de P et détermine au bout d'un temps fini, si la propriété est satisfaite ou non par le modèle. Pour être automatisable, la vérification doit être décidable et aussi de complexité acceptable. On différencier entre deux grandes catégories de méthodes: les méthodes syntaxiques et les méthodes sémantiques.

1.0.14 Méthodes syntaxiques

Ce sont des preuves au sens mathématique du terme. Elles cherchent à déterminer si une propriété peut être obtenus.. difficile a automatiser

INSÉRÉ SLIDES 61

Pour résoudre syntaxiquement, on utilise les méthodes vu dans le cours 8215 (logique prédicats/propositionelle)

1.0.15 Méthodes sémantiques

Dans les méthodes sémantiques, on se base sur l'exécution du modèle. L'approche populaire est le model-checking. Sa s'appuie sur deux formalisme:

- système de transition
- logique temporelle

INSÉRÉ SLIDES 65

1.0.16 Model-checking

Normalement automatique, et produit des contrexemple sous forme de traces qui sont utils à la compréhension des situations d'erreurs et àla correction. basé sur la sémantique d'entrelacement, avec sa, sa pourrait générer n! et plus de 2^n états