Projet de recherche

Antoine Forgerou Jérémy Bardon Nicolas Bourdin

Sommaire

1 Présentation du laboratoire

Acteur central du développement de l'informatique dans la région des Pays de La Loire, le LINA (Laboratoire d'Informatique de Nantes Atlantique) est un laboratoire de recherche en sciences et technologies du logiciel qui est dirigé par Pierre Cointe. Avec ses 180 membres, ce laboratoire est actuellement situé sur deux sites Nantais : la Lombarderie (Faculté des Sciences et Techniques) et la Chantrerie (Ecole des Mines et Polytech' Nantes).

1.1 Les différentes équipes

Le LINA est composé des équipes suivantes :
— AeLoS : **A**rchitecture et **Lo**giciel **S**ûrs
— ASCOLA : **AS**pect and **CO**mposition **LA**nguages
— AtlanMod : **Atlan**tic **Mod**eling
— COD : **CO**nnaissances et **D**écision
— ComBi : **Com**binatoire et **Bi**oinformatique
— DUKe : **D**ata **U**ser **K**nowledg**e**
— GDD : **G**estion de **D**onnées **D**istribuées
— GRIM : **G**estion, **R**ésumé, **I**nterrogation, et apprentissage sur les **M**asses de données
— OPTI : Optimisation globale, optimisation multi-objectifs
— TALN : **T**raitement **A**utomatique du **L**angage **N**aturel
— TASC : Programmation par contraintes

2 L'équipe AeLoS

L'équipe AeLoS, dirigé par Christian ATTIOGBE, est une équipe de recherche intégré au LINA résultant de la fusion de deux anciennes équipes COLOSS et MODAL.

Le projet central de cette équipe s'appuie sur les trois thématiques suivantes :

— Architecture :
— Composants logiciels corrects :
— Multiformalisme et analyse multifacette :

3 Présentation du sujet

3.1 Problématiques

Contexte : Approches formelle des systèmes embarqués communiquants. La construction rigoureuse d'un logiciel se fait à partir de son modèle. Pour des logiciels complexes (par exemple ceux qui ont de nombreux composants différents – hétérogènes– et qui communiquent), on a affaire à plusieurs modèles (hétérogènes aussi) qui doivent intérargir de façon cohérente, pour garantir l'interopérabilité sémantique entre les modèles puis les composants logiciels. Le domaine des systèmes embarqués (et aussi des objets connectés) regorge d'exemples. Description du travail : Etudier l'interopérabilité entre des modèles de composants (logiciels ou non). En nous appuyant dans un premier temps sur des modèles à base d'automates à états, il s'agit dans le cadre du stage d'initiation à la recherche, de contribuer à la conception et au développement d'outils passerelle entre modèles. Nous travaillons sur plusieurs aspects : - Lorsque cela est possible, en fonction de la sémantique des modèles, un modèle donné est encodé par un autre modèle choisi mais qui est sémantiquement équivalent. - Lorsque cela est possible, en fonction de la sémantique des modèles, un modèle peut interagir avec un autre avec des contrats (de type Assume/Guarantee) - Plug'N'Check (analyse systématique de modèles/composants) - etc

3.2 Solutions proposées par l'équipe AeLoS

Pour répondre à la problématique, l'équipe AeLoS nous a proposé 3 façons de faire tout en nous laissant la liberté d'en proposer d'autres.

Dans le but de simplifier les exemples, nous avons fait le choix de représenter les modèles comme des formes géométriques que l'on essayerai d'emboiter.

La première solution consiste à adapter un des modèles au deuxième quand

cela est possible. On peut prendre l'exemple du théorème de Kleene qui assure qu'un automate à états finis peut être écrit sous la forme d'une expression rationnelle et vice et versa.

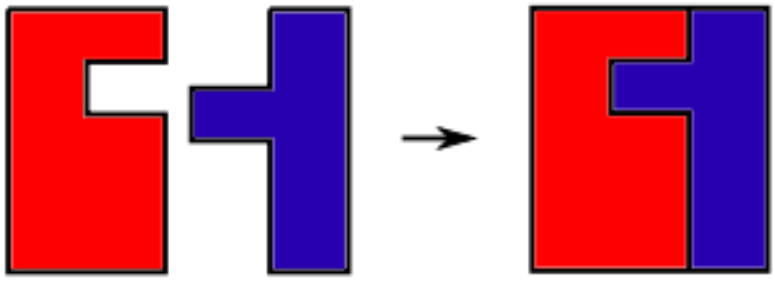


FIGURE 3.1 – Solution 1 - Adaptation d'un modèle

L'approche qui parait la plus évidente mais qui peut se révéler complexe consiste à intégrer un troisième modèle dans le système. Ce dernier va alors jouer le rôle de passerelle entre les deux modèles que l'on veut faire communiquer.

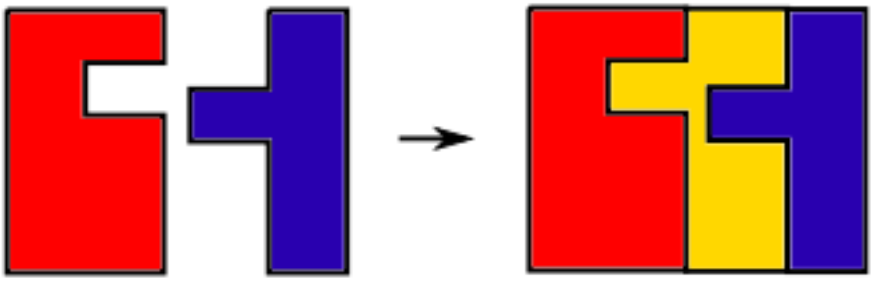


FIGURE 3.2 – Solution 2 - Interface entre les modèles

Plug'N'Check est le nom de la troisième solution que l'équipe nous a soumise. Le principe est de faire interagir les modèles ensemble puis d'effectuer des vérifications pour déterminer si les modèles peuvent communiquer à 100%, en partie ou pas du tout.

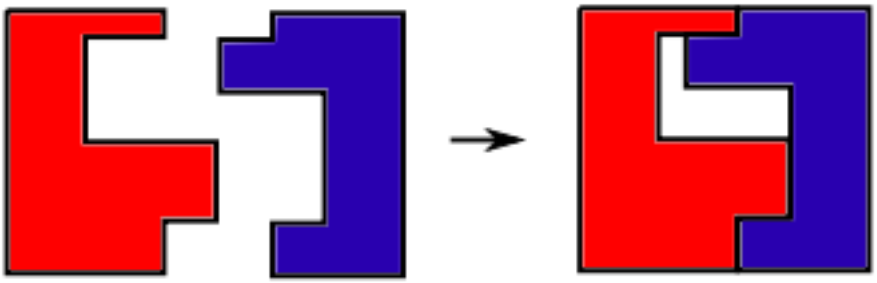


FIGURE 3.3 – Solution 3 - Plug'N'Check

Dans le cadre de notre stage d'initiation à la recherche, nous nous sommes orientés vers la première solution. Nous avons donc entrepris de transformer n'importe quel modèle dans un language commun afin de permettre de les interfacer.

3.3 UPPAAL : un outil de modelisation des automates

UPPAAL est un environnement intégré d'outils pour la modélisation , la validation et la vérification des systèmes temps réel modélisés comme des réseaux d' automates. Uppaal se compose de trois parties principales :

— un langage de description
— un simulateur
— un vérificateur de modèle

Le langage de description est un langage de commande non-déterministe avec des types de données (par exemple des entiers , des tableaux, etc.) . Il ce sert de la modélisation ou du langage de conception pour décrire le comportement du système grâce à des réseaux d'automates étendus avec des variables d' horloge et des données .

Le simulateur est un outil de validation qui permet l'examen de possibles exécutions d'un système au début de la conception(ou la modélisation) et fournit un moyen peu coûteux de détection de défaut avant la vérification par le vérificateur de modèle qui couvre l'ensemble des comportement dynamique du système.

Le modèle orthographique peut vérifier les propriétés invariantes et l'accessibilité en explorant l'espace et l'état d'un système, c'est à dire l'analyse de l'accessibilité en termes d'états symboliques représentés par des contraintes .

3.4 Le format DOT

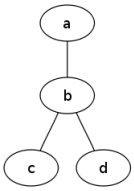
Le langage DOT est un langage de description de graphe dans un format texte. C'est une manière simple de décrire des graphiques que les humains et les programmes informatiques peuvent utiliser. Les graphes DOT sont généralement des fichiers avec pour extension un .gv (ou .dot).

La syntaxe DOT peut aussi bien décrire des graphes non-orientés que des graphes orientés, comme des automates finis. Il est possible de placer des labels sur les transistions pour par exemple leurs donner un nom ou une explication. Le language possède aussi des attributs permettant une description graphique plus poussée, par exemple choisir la couleur d'un noeud ou de dessiner une transition en pointillé.

3.4.1 Graphe non-orienté

```
graph monGraphe {
    a -- b -- c;
    b -- d;
}
```

Resultat :



3.4.2 Graphe orienté

```
graph monGraphe {
    a -> b -> c;
    b -> d;
}
```