Rapport de Stage
-
Traduction de composants Scade/Lustre vers des machines B

Florian Thibord

23 juillet 2013

Table des matières

Chapitre 1

Introduction

Ce stage s'est déroulé au sein d'un projet financé par l'Agence Nationale de la Recherche : CERCLES2. Ce projet a pour but la certification compositionnelle des logiciels embarqués critiques et sûrs. C'est la notion de composant réutilisable et assemblable pour former des logiciels critiques et sûrs qui est à la base du projet, l'intérêt étant à la fois pratique par le gain de temps et d'effort, et économique.

Pour assurer que ces composants sont sûrs et réutilisables, on utilise une méthode formelle, qui permet d'exprimer la signification d'un composant dans un formalisme mathématique. Il faut alors introduire le concept des contrats : un contrat est un composant associé à des conditions sur ses entrées (pré-conditions) et sur ses sorties (post-conditions). A partir d'un composant et de son contrat, il faut alors vérifier formellement que :

- (i) la définition du composant satisfait le contrat
- (ii) l'utilisation du composant satisfait les pré-condition, et en conséquence de (i) le résultat satisfait les post-conditions.

La validation est alors faite par une démonstration formelle.

Un acteur majeur du développement de systèmes embarquées critiques est Scade, un acronyme pour Safety Critical Application Developpement Environment. Cet environnement de développement est basé sur la programmation graphique, par schémas-blocs, permettant de définir des programmes faciles à lires et permettant d'engendrer directement du code compilable (C ou ADA). Il est notamment utilisé en aéronautique (grande partie du logiciel embarqué de l'A380), dans le domaine spatial ou dans le nucléaire. C'est donc avec Scade que sont écrits les composants, les contrats étants rédigés sous forme textuelle en accompagnement du composant.

INSERER CAPTURE SCADE

Concernant la méthode formelle, c'est la méthode B qui est utilisée. Introduite par J.R. Abrial dans les années 80, elle est basé sur le raffinement de spécifications formelles vers une spécification exécutable. La spécification formelle est rédigée dans un formalisme mathématique de haut niveau appelé machine abstraite. Le raffinement de cette machine abstraite consiste à la reformuler de façon plus concrète et à l'enrichir avec des substitutions correspondants aux instructions du composant. Ce raffinement de plus bas niveau est appelé implantation. Chaque étape de raffinement passe par une étape d'obligation de preuve, une validation par démonstration formelle, garantissant la fidélité de la spécification raffinée par rapport à la spécification originale.
L'avantage de la méthode B est qu'elle est permet la composition de programme, et ainsi permet de faire référence à des composants déjà certifiés. De plus cette méthode a déjà fait ses preuves industriel-

lement, elle a notamment été utilisée pour développer la ligne METEOR (ligne 14) du métro parisien, qui est entièrement automatisée.

Mon travail fut de développer un traducteur permettant de passer d'une méthode à l'autre. Le traducteur suit une ligne de compilation classique, prenant en entrée un code issu de Scade et produisant en sortie une machine abstraite correspondant aux spécification du contrat, ainsi que la machine raffinée qui implante le composants.

SCHEMA PRINCIPE GENERAL

Chapitre 2

Scade

Scade a été développé par le laboratoire Verimag à partir des travaux sur le langage synchrone Lustre, puis repris par Esterel-technologie. On retrouve ainsi les notions de Lustre dans le langage de Scade, un programme est découpé en noeuds. Ces noeuds sont les composants que nous voulons traduire. Les noeuds Scade considéré dans le cadre du projet Cercles2 sont soumis à quelques restrictions. En effet, il faut limiter le langage utilisé, car certains éléments du langage sont spécifiques aux langages synchrones et ne sont donc pas traductibles en B.

2.1 Le temps avec Scade

Le temps est un élément primordial dans ces systèmes dits réactifs, il est découpé en instants discrets.
Le langage de Scade est basé sur la notion d'instant,

Une horloge unique Ainsi, la première restriction à noter est qu'il n'y a qu'une seule horloge, celle de base. Avec les langages synchrones, on peut synchroniser des instructions sur des horloges différentes avec, par exemple, l'opérateur when :

EXEMPLE/RESULTAT

Mais dans les programmes que nous manipulerons, il n'y aura donc pas de définitions d'horloges, d'utilisation de when, ou current. Les instructions d'un noeud sont toutes calculées sur l'horloge de base uniquement.

Des registres La seconde restriction concerne l'utilisation des opérateurs pre et ->. Ce sont des opérateurs temporels :

- pre X donne la valeur de l'expression X à l'instant précédent. A l'instant 0 [1], la valeur de pre X n'est pas définie.
- A -> B donne au premier instant la valeur de l'expression A, puis la valeur de l'expression B pour les instants allant de 1 à n.

On ne pourra utiliser que la construction suivante utilisant ces deux opérateurs :

$$A \rightarrow (pre\ X)$$

1. On suppose que le premier instant est l'instant 0

Cette construction correspond au bloc SIMULINK 1/Z, où A représente un flux constant qui donnera la valeur de sortie à l'instant 0 du bloc. Puis pour les instants 1 à n, on aura la valeur de l'expression X à l'instant (1 à n)-1.
On appellera cette construction un registre, qui est initialisé avec la valeur A, et qui permet d'accéder à la valeur précédente de X à tout instant. Cette construction permet de donner un état à un composant.

LE COTE REGISTRE = ETAT A DEVELOPPER

+ Utilisation de l'opérateur FBY de scade ? (pas dans lustre... comment est-il traduit ?) + Si preuves de correction en aout, développer le problème langage synchrone vers langage formel/impératif ?

2.2 Contrat

Assertions On peut définir des assertions dans un noeud afin de poser des restrictions sur les valeurs d'entrée ou de sortie du composant. Avec Scade, ces assertions sont possibles avec :

- *assume x : expr*, où x est une des entrées du noeud, et expr un prédicat portant sur cette entrée.
- *guarrantee x : expr*, où x est une des sorties du noeud, et expr un prédicat portant sur cette sortie.

Ces assertions forment le contrat du composant, et seront obligatoires sauf pour la restriction sur les booléen qui est triviale (la valeur sera vraie ou fausse).

EXEMPLE

Concernant les types Ensuite, au niveau des types de données utilisées, on pourra manipuler des entiers, réels et booléens. Et on pourra également manipuler des tableaux de ces types. En revanche, les types définis par l'utilisateur tels que les types enregistrement ne seront pas gérés par le traducteur.

?HISTOIRES DE POLYMORPHISME POUR LES OPERATIONS SUR LES TABLEAUX A VOIR ?

2.3 Le langage d'entrée du traducteur

Scade étant un environnement de programmation par schémas-blocs, on développe avec des boîtes. Par exemple, une addition sur les fluxs A et B s'écrit :

EXEMPLE D'UN PROGRAMME SIMPLE EN SCADE

Ce langage en boîtes est basé sur le langage synchrone Lustre, et c'est le programme en Lustre que nous allons parser avec le traducteur. La représentation graphique du programme est ainsi réécrite en Lustre avant d'être traduite. C'est donc du code Lustre que l'on traduit.

REPRISE DE L'EXEMPLE EN LUSTRE

PUIS DETAILLER COMMENT OBTENIR LE LUSTRE ? (FICHIER SAOFD)

Expressions Les expressions disponibles sont toutes les expressions arithmétiques (+, -, /, *, mod), les expressions relationnelles (<, >, <=, >=, =, <>) et logiques (and, or, xor, not). Les expressions conditionelles sont également possibles (if .. then .. else ..),en revanche les constructions à base de l'opérateur case ne sont pas acceptée. (A AJOUTER ?)
Sont également disponibles les opérations sur les tableaux, telles que la définition, l'index, et la concaténation.

Ajouts d'opérations au langage A REFORMULER...

Dans la situation où un utilisateur veux ajouter une opération qui n'est pas reconnue par le traducteur, il peut définir cette opération dans une bibliothèque Scade, et définir cette même opération dans une machine B. Cette opération sera vue comme un appel de noeud. Une fois la traduction terminée, il suffit d'indiquer en haut du couple de machines B produites quelle machine à importer est utilisée dans ce programme.

SCHEMA A INCLURE

Chapitre 3

Machines B

raisonnement rigoureux, logiciel sur Raffinements de spécifications formelle vers une spécification exécutable. Chaque raffinement passe par une étape d'obligation de preuves

3.1 Machine B

3.1.1 Structure d'une machine

Clause, expression, predicats, substitutions
Puis développer clause et prédicats

3.1.2 Clause

REFINES

IMPORT

SEES

CONCRETE_VARIABLES

INVARIANT

INITIALISATION

OPERATIONS

3.1.3 Prédicats

Un prédicat est une formule mahtématique qui a une valeur booléenne.

3.2 Expressions

3.2.1 Arithmétiques et logiques

Expressions de B (rapidement : arithmétique, logique)

3.2.2 Fonction

Appel d'opération définie dans une autre machine

Détail sur les tableaux en B

3.3 Substitutions

[S]P : "la substitution S établit le prédicat P".

BLOC

SEQUENCE

VARIABLE LOCALE

PRECONDITION

DEVIENT EGAL

CONDITION

APPEL OPERATION

3.4 Raffinements

spécification = machine abstraite, puis raffinement vers implantation. Nous produisons les 2 machines sig/impl (abstrait/concret)

3.4.1 Signatures

Reprise du contrat, utilisation de la substitution precondition + EXEMPLE

3.4.2 Implémentation

Reprise du composant, détail de l'opération

3.5 Proof Obligation

Cf Proof obligation paper

A chaque raffinement, on passe par une étape de type checking et d'obligation de preuves. Démonstrations basée sur les principales substitutions. Montrer qu'on utilise ce qu'il faut pour assurer que les obligations de preuves sont possibles.

Principe :

Hypothèses (liste de prédicats) => But (doit être prouvé sous ces hypothèses)

Pour cela on applique des substitutions aux prédicats. Développer pour les différentes substitutions, exemples...

Chapitre 4

Schémas de traduction

... Reprendre les schémas de traduction du rapport scade to B.

4.1 Specification

Signature des noeuds -> specification en B Reprise des informations de typage Reprise du contrat et traduction en PRE .. THEN .. END

4.2 Implémentation

Ajout des variables d'etats + initialisation (pour les registres) Ajout des variables locales tri topologique pour la séquence de substitution

Chapitre 5

Exemples tests

Faire 2/3 exemples utilisants tous les traits utilisables de scade. Montrer le tri topologique, Utiliser les tableaux Cas critiques ? A TROUVER

Chapitre 6

Conclusion

Difficultés, Apport projet, Après projet, ...