La **Cer**tification **C**ompositionnelle des **L**ogiciels **E**mbarqués Critique**S** et **S**ûrs

Documentation du traducteur Scade2B

L_Tâche _préliminaire

Livrable	Documentation du traducteur Scade2B		
Projet	CERCLES2	**N° de contrat**	ANR-10-SEGI-017
Référence livrable	L_Tâche _préliminaire	**Date**	octobre 2013
Statut	préliminaire	**Version**	
Point de contact	Florian Thibord	**Organisation**	PPS
Tél/Fax	Tél: +33 6 85 14 54 89	**Courriel**	florian.thibord@pps.univ-paris-diderot.fr

Auteurs

Nom	Organisation	Courriel
Florian Thibord	PPS	florian.thibord@ pps.univ-paris-diderot.fr

Historique des changements

Version	Date	Modification	Pages
0.1	26 oct. 2013	Création	toutes

Table des matières

Cette documentation vise à décrire le fonctionnement et l'évolution du traducteur Scade2B.

I. Initialisation d'un registre à partir d'une entrée d'un composant

I.1 Exemple : `integrateur_sature`

Jusqu'à présent, les registres étaient initialisé uniquement avec des constantes, or nous aimerions pouvoir utiliser une entrée du composant pour initialiser un registre.

Considérons l'exemple `integrateur_sature` suivant :

Planche à venir...

Ce noeud contient une opération `fby`, dont l'initialisation correspond à l'entrée `P_INITIALISATION`. La version textuelle de ce noeud est la suivante :

FIGURE I.1: Version textuelle du composant `integrateur_sature`

```
node integrateur_sature(incr : int; P_INITIALISATION : int; P_SATURATION : int)
  returns (integr : int; integr_m1 : int)
var
  _L1 : int;
  _L2 : int;
  _L3 : int;
  _L7 : bool;
  _L6 : int;
  _L8 : int;
  _L9 : int;
let
  _L1= incr;
  integr= _L6;
  integr_m1= _L3;
  _L2= _L1 + _L3;
  _L3= fby(_L6; 1; _L8);
  _= _L7;
  _L6, _L7= #1 saturation(_L2, _L9);
  _L8= P_INITIALISATION;
  _L9= P_SATURATION;
tel
```

Si on traduit ce noeud dans l'état actuel du traducteur, on obtiendra le couple de machines B suivantes :

FIGURE I.2: Machines B de `integrateur_sature`

```
MACHINE M_integrateur_sature
SEES M_Consts
OPERATIONS

integr,integr_m1<--integrateur_sature(
  incr,P_INITIALISATION,P_SATURATION)=
 PRE
   P_SATURATION : INT &
   P_INITIALISATION : INT &
   incr : INT
 THEN
   integr_m1 :: { ii | ii : INT }||
   integr :: { ii | ii : INT }
 END
END
```

```
IMPLEMENTATION M_integrateur_sature_i
REFINES M_integrateur_sature
SEES M_Consts
IMPORTS M_saturation

CONCRETE_VARIABLES L3
INVARIANT
   L3 : INT
INITIALISATION
   L3 := L8

OPERATIONS

integr,integr_m1<--integrateur_sature(
  incr,P_INITIALISATION,P_SATURATION)=
 VAR L1, L2, L7, L6, L8, L9 IN
   integr_m1 := L3;
   L9 := P_SATURATION;
   L8 := P_INITIALISATION;
   L1 := incr;
   L2 := L1 + L3;
   L6, L7 <-- saturation(L2, L9);
   integr := L6;
   L3 := L6
 END
END
```

L'implémentation n'est pas correctement initialisée. Dans la clause `INITIALISATION`, la variable `L8` n'est pas définie, car c'est une variable locale de l'opération qui reçoit le flux d'entrée `P_INITIALISATION`. Plusieurs solutions permettent de résoudre ce problème :

- Solution 1 : Déclarer une nouvelle variable d'état `temps`, de type `INT`, initialisée à 0, et réaliser l'initialisation du registre non plus dans la clause `INITIALISATION`, mais en début d'opération.
- Solution 2 : A la place de la variable `L8` comme paramètre de la machine, utiliser l'entrée `P_INITIALISATION`, et supprimer cette entrée de la liste des paramètres d'entrées de l'opération.

I.1.1 Solution 1

En utilisant une substitution Condition en début d'opération, on peut tester la valeur de la variable d'état `temps` et affecter la valeur du registre en conséquence :

```
IF temps = 0 THEN L3 := L8 ELSE L3 := L3 END;
```

Or, en début d'opération, L8 n'a pas encore été affectée, il faut donc placer la substitution dans laquelle L8 est affectée d'une valeur avant la condition. Ainsi l'opération de cette machine est correcte.

Cependant, quelle valeur donner au registre L3 dans la clause INITIALISATION ? Cette valeur d'initialisation ne sera pas prise en compte par l'opération car la première L3 sera immédiatement affectée par la valeur d'initialisation dans la substitution condition. Mais pour les obligations de preuves, il faut que les substitutions de la clause INITIALISATION respectent l'INVARIANT de la machine. Si un invariant est lié au registre L3, on ne peut pas lui affecter une valeur d'initialisation aléatoire. Dans l'exemple, le registre est lié à la sortie integr_m1. Si on pose la condition suivante sur cette sortie :

```
guarantee G_1 : integr_m1 < 10 and integr_m1 != 0
```

Elle sera traduite par l'invariant B :

```
INVARIANT
    L3 : INT & L3 < 10 & L3 <> 0
```

Il faut donc initialiser le registre à une valeur respectant cet invariant, alors que cette valeur ne sera jamais utilisée. Une solution n'ayant pas ce genre de contrainte est donc préférable.

I.1.2 Solution 2

En utilisant des machines paramétrées, on peut alors directement initialiser les variables d'état de la machine avec les paramètres donnés. On a alors plus besoin d'une substitution condition en début d'opération. Cette solution consiste à paramétrer la machine avec la variable L8, qui ne sera plus une variable locale de l'opération mais un paramètre de la machine. Pour que la machine IMPLEMENTATION soit correcte, il faut ajouter une clause CONSTRAINTS dans laquelle on indique le type et les restrictions liées aux différents paramètres de la machine, sous forme de prédicat.

Dans la clause INITIALISATION, L8 sera alors défini, et les obligations de preuves s'appuieront sur le prédicat défini dans la clause CONSTRAINTS pour vérifier la correction de l'initialisation de la machine.

Dans l'opération, L8 est affectée par l'entrée de l'opération P_INITIALISATION, et les deux

I.1.3 Solution 2

I.2 Utilisation de machines paramétrées