Guillaume RIPOLL
L3 -> L3

Compte-rendu de réunion : 25/07/2014

Stage facultatif sur la génération de code parallèle pour les langages synchrones à l'IRIT avec l'équipe ACADIE

Présentation du travail effectué :

Présentation de l'implémentation non finie de signal_to_java
to do list :
gérer import Usable.* si usage de Call dans une tâche
gérer les blocs when où on retourne false sinon
rendre le codage automatique des fonctions add, mul et sub lorsque celles-ci sont utilisées

-> remarque :
hypothèse d'un code aplatit, c'est-à-dire pas plus d'une instruction par assignation

Contenu de la réunion :

1. La gestion du default : x := a default b

2 algorithmes proposés :

- 1 - on initialise N(x) = 1 (où N(_) retourne le nombre de dépendances de _)
Le fait de calculer a peut alors lancer x.
Calculer b ajoute x à la liste des tâches à lancer en second temps (liste à créer dans GlobalData).
-> 2 types de dépendances : directe pour a et indirecte pour b.
Quand il n'y a plus de tâche active,
Tant qu'il reste des tâches secondaires faire
Pour chaque tâche t secondaire faire
Si t NON dépendant d'une tâche secondaire
lancer t
avantage : performant
désavantage : preuve lourde

- 2 - on exécute x dès qu'on a accès à la valeur de a ou de b
si on accède à a après avoir pris en compte b on recalcule tout ce qui était dépendant de x
-> gestion de rollback
avantage : preuve plus facile, implémentation intéressante
désavantage : couteux en mémoire et en nombre d'instructions

2. Le problème de l'horloge :

Un process P est correct quand il est possible de déterminer l'horloge de chacune de ses tâches et une tâche ne se calcule pas forcément à chaque cycle. Le non calcul de l'horloge implique qu'on ne sait pas quoi lancer quand. Le calcul des valeurs est alors dépendant d'un booléen de présence. Comme il ne reste plus beaucoup de temps on suppose ces booléens fournis en entrée (lu dans le fichier In). L'horloge principale nous ait aussi fourni même si les booléens suffisent à savoir s'il faut ou pas calculer une valeur.
Grâce à cela, la gestion du default ne dépend plus que du booléen et donc pas d'algorithme à implémenter.

Solution :
implémentation de ŝ pour désigner l'horloge d'une variable s
toute variable dépend alors de son horloge.
N(s) = last N(s) +1

3. Exemple :

```
x1 = y1 $1 init 0
y1 = x1 + 1
x2 = y2 $1 init 0
y2 = x2 + 1
t = x2 when (call modulo(x2, 2) = 0)
z = x1 + t
```

Valeurs :

x2	0	1	2	3	4	5	6
y2	1	2	3	4	5	6	7
x1	0	*	1	*	2	*	3
y1	1	*	2	*	3	*	4
t	0	*	2	*	4	*	6
z	0	*	3	*	6	*	9

-> détection de x2 comme horloge principale non triviale.