Guillaume RIPOLL
fin de L3

Compte-rendu de réunion : 03/07/2014

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 la structure Ocaml imaginée pour générer le Java et du code Java qui serait généré dans le cadre d'un exemple.

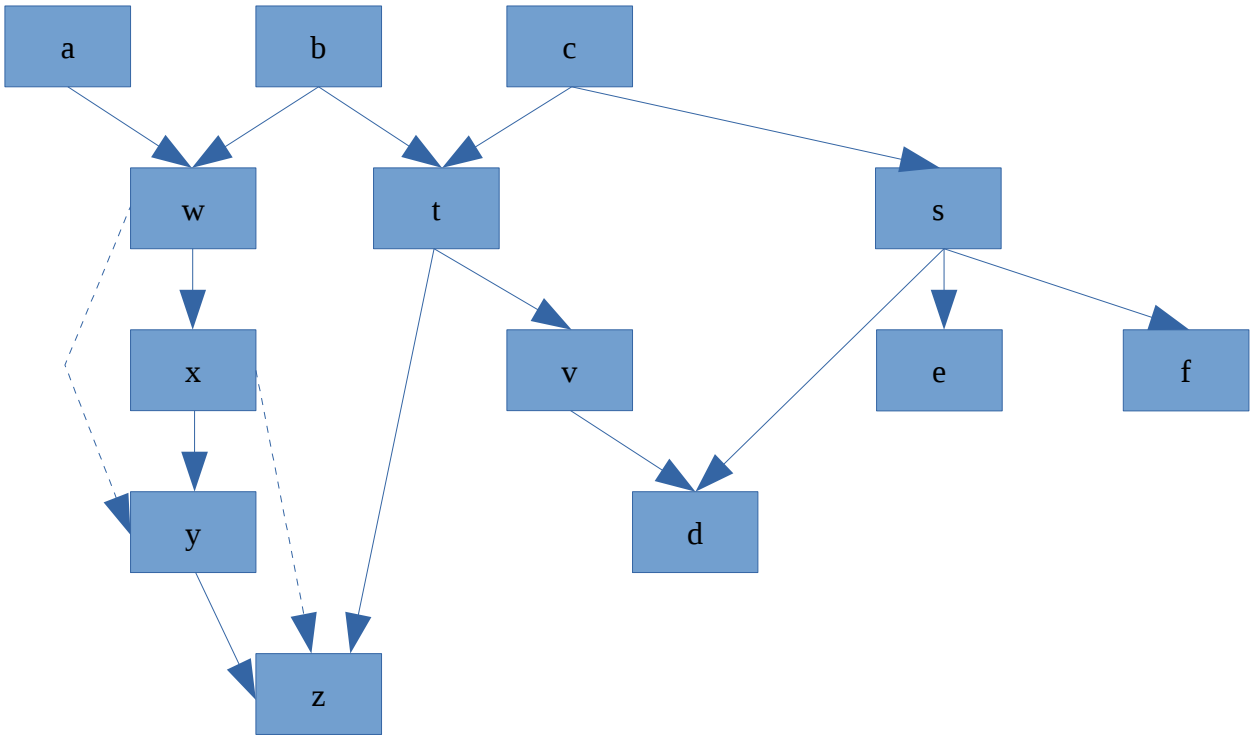
-> Remarques :

1. non parallélisme du fait du synchronized (obligatoire pour les wait et notify) sur une structure de données trop généraliste
   -> synchroniser sur les variables elles-même
2. prévoir des structures de données côté Ocaml adaptées aux schémas
3. prévoir l'évolution du code Ocaml :
   -> liste d'expressions dans un thread à garder
   -> fonction fusion(Thread, Thread) -> Thread
   -> fonction priorityChoice(Thread, Thread) -> Boolean
4. en ce qui concerne les tests en trop sur les wait, la documentation parle de spurious wakeup

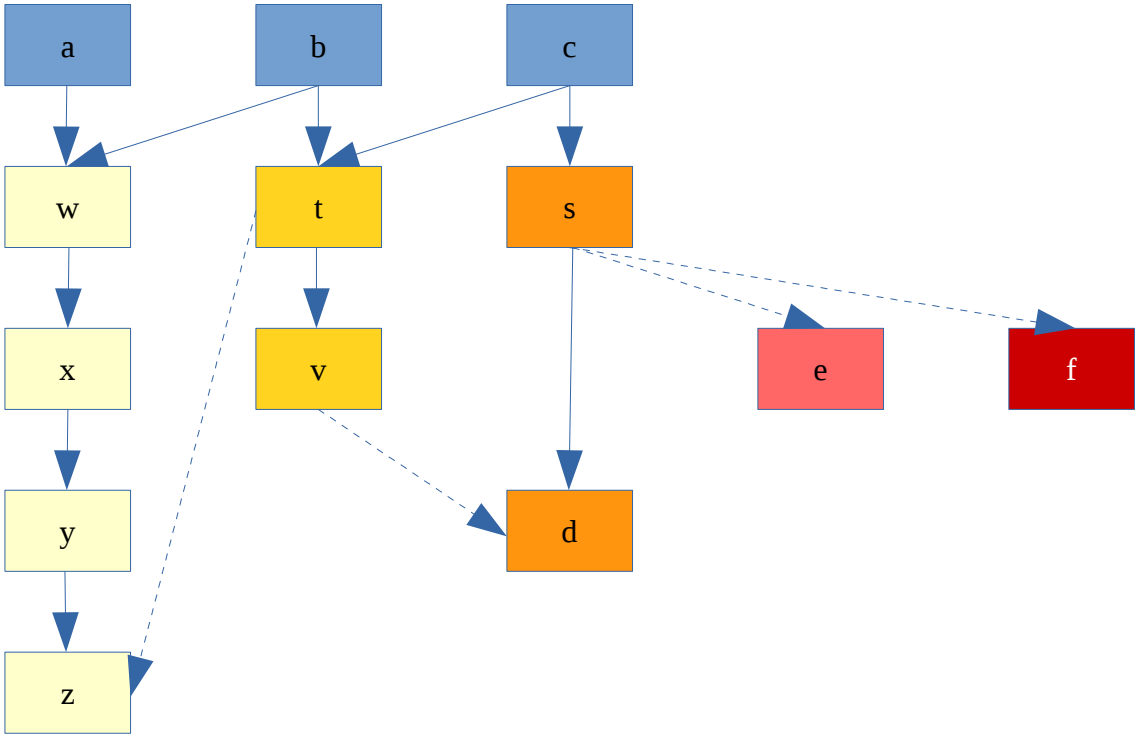
Définition des nouveaux objectifs :

Création de nouvelles structures de données :

1 - Graphe transitif :



2 - Graphe séquentiel :



signal calculé par le thread n°0

signal calculé par le thread n°1

signal calculé par le thread n°2

signal calculé par le thread n°3

signal calculé par le thread n°4

signal d'entrée

x → y y utilise x

x ⇢ y y a besoin d'être synchronisé avec x pour être définit

Pour un graphe séquentiel :

On voit que a, w, x et y pourraient ainsi être déclarés en local au thread n°0.
-> réfléchir à la portée des variables

On pourrait mettre e et f dans un même thread
-> Prévoir :

une fonction de fusion qui prend 2 threads et retourne leur union
une fonction de choix de fusion, qui prend 2 threads et retourne « faut-il fusionner ? »
une fonction de priorité qui prend 2 listes de 2 threads et les fusionne en choisissant la priorité de chaque élément de chaque liste.

Par exemple, pour la fusion du thread 0 et du thread 1, on pourrait calculer d'abord w, puis x, puis y, puis t ou préférer calculer t d'abord, puis w – x – y – z.