Compilation de programmes réactifs

Sujet de stage

La concurrence coopérative est un modèle de programmation très répandu. Il est par exemple utilisé pour programmer des interfaces graphiques, des serveurs web, ou des problèmes de simulation. Il a de nombreux avantages tels que l'absence de course critique et des implantations légères et efficaces.

La concurrence coopérative est souvent proposée sous forme de bibliothèque dans un langage généraliste. On peut par exemple l'utiliser en OCaml à travers des bibliothèques comme `Lwt`, `Async` ou `Equueue`.

ReactiveML [2] est un langage qui étend OCaml avec des constructions de concurrence coopérative. L'avantage de proposer un langage de programmation à la place d'une bibliothèque est de pouvoir implanter des analyses et optimisations dédiées.

Le but du stage est de faire gagner un facteur 5 à l'efficacité de l'exécution du langage !¹

Considérons l'extrait de programme ReactiveML suivant :

```
let process cascade s1 s2 =
  await immediate s2; print_int 2
  ||
  await immediate s1; print_int 1; emit s2
  ||
  print_int 0; emit s1
```

Ce programme exécute trois branches en parallèle. La dernière émet un signal qui va déclencher l'exécution de l'avant dernière qui elle même va déclencher l'exécution de la première branche. Ainsi ce programme va toujours afficher « 012 ».

Avec l'implantation actuelle du langage, l'ordonnancement des calculs est dynamique. Ainsi, dans l'exemple précédent, l'exécution peut commencer par tester la présence de `s2` et de `s1` avant d'émettre `s1` et retourner sur la branche qui testait `s1` puis celle qui testait `s2`. Une analyse statique du programme aurait pu déterminer les dépendances du programme et fixer lors de la compilation l'ordre des calculs.²

Lors du stage, vous devrez réaliser un compilateur pour un sous-ensemble du langage ReactiveML en vous inspirant des techniques de compilation du langage synchrone Esterel [1, 3]. Puis vous étudierez comment traiter les caractéristiques propres au langage : ordre supérieur, signaux comme valeurs de premières classes, etc.

Laboratoire d'accueil : Équipe Parkas du Département d'Informatique de l'École normale supérieure
Encadrant : Louis Mandel (louis.mandel@lri.fr)

Références

- [1] E. Closse, M. Poize, J. Pulou, P. Venier, and D. Weil. SAXO-RT : interpreting Esterel semantic on a sequential execution structure. In *Electronic Notes in Theoretical Computer Science*, volume 65, pages 864–878. Elsevier Science Publishers, 2002.

¹Évidemment, uniquement sur certains programmes.

²Sur l'exemple précédent, en augmentant le nombre de signaux mis en jeux, on peut simplement arriver à un facteur 5 dans l'efficacité du code généré en fonction de l'ordonnancement des calculs.

- [2] Louis Mandel and Marc Pouzet. ReactiveML : un langage fonctionnel pour la programmation réactive. *Technique et Science Informatiques (TSI)*, 2008.
- [3] Dumitru Potop-Butucaru, Stephen A. Edwards, and Gérard Berry. *Compiling Esterel*. Springer, 2007.