Langages réactifs synchrones

2019–2020 M. Pouzet

Projet : MiniLucy [1]

Le but de ce projet est d'écrire un compilateur vers du code séquentiel pour un langage synchrone proche de Lustre puis d'étendre le langage avec des automates hiérarchiques en les traduisant vers le langage noyau.

Nous vous proposons de suivre l'organisation du compilateur KCG de SCADE qui s'appuie sur les deux articles suivants :

— Darek Biernacki and Jean-Louis Colaco and Grégoire Hamon and Marc Pouzet. *Clock-directed Modular Code Generation of Synchronous Data-flow Languages.* ACM International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES). 2008. [2]
Cet article décrit comment traduire un langage noyau proche de Lustre vers un langage intermédiaire séquentiel. Les programmes de ce langage sont ensuites traduits vers du code C, par exemple.

— Jean-Louis Colaço and Bruno Pagano and Marc Pouzet. *A Conservative Extension of Synchronous Data-flow with State Machines.* ACM International Conference on Embedded Software (EMSOFT'05). 2015. [3]
Cet article montre comment traduire les automates hiérarchiques vers le langage noyau.

1 Travail demandé

Vous pourrez utiliser le parseur du sous-ensemble de Lustre utilisé dans le second projet proposé cette année :

`http://www.di.ens.fr/~pouzet/cours/comasic/projet/lmoch.tar.gz`

Les constructions `merge` et `reset` devront être ajoutées.

Vous avez également accès aux sources (en OCaml) du compilateur Heptagon qui suit l'organisation décrite dans les deux références données ci-dessus (le langage data-flow s'appelle *Minils*). N'hésitez-pas à reprendre des parties du parseur.

`http://heptagon.gforge.inria.fr`

L'objectif est d'écrire un compilateur, comprenant dans la partie "front-end" : parseur, typeur, vérification des horloges et l'absence de boucles de causalités et, pour la "middle-end", la génération d'un code séquentiel intermédiaire puis sa traduction vers du code impératif. Pour celui-ci, vous pourrez choisir C ou un autre langage de bas niveau, par exemple Rust.

1. Lucy était le nom interne au compilateur du langage Lucid Synchrone.
2. Article disponible à l'adresse : `http://www.di.ens.fr/~pouzet/bib/lctes08a.pdf`.
3. Article disponible `http://www.di.ens.fr/~pouzet/bib/emsoft05b.pdf`.

— Les types de données seront simples : `int`, `bool`, `real`.
— Vous devrez écrire un compilateur vers du code séquentiel. Pour chaque étape de compilation, vous devrez la compléter par une fonction indépendante de test. Précisément, si f_C est une fonction de compilation (e.g., la phase de normalisation, de scheduling ou de traduction vers du code Obc impératif) et telle que $y = f_C(x)$, vous proposerez une fonction de validation (ou test). Etant donnés x et y, la fonction (à valeur booléenne) f_T sera telle que si $f_T(x, y) = \mathit{true}$ alors x et y ont la même sémantique.
  L'objectif est d'écrire des fonction de test les plus simples, si possible dans un style purement fonctionnel, dans une perspective d'en faciliter la preuve formelle.
— Une fois réalisé le compilateur de "minils" (noyau Lustre avec `merge` et `reset`), vous étendrez le langage avec des automates et implémenterez la compilation par traduction dans le langage noyau. Pour ce projet, vous vous limiterez à des automates avec transitions faibles.
— Vous avez toute liberté pour le choix du langage d'implémentation. Heptagon (et KCG) sont implémentés en OCaml. Vous pouvez choisir OCaml, Coq ou Rust, par exemple.

Le projet est à faire seul ou en binôme si vous ne faites que la partie "minils". Si vous faites l'extension avec des automates, vous pouvez le faire à trois ou quatre.

Le projet doit être remis par email à `marc.pouzet@ens.fr` avant le **28 février 23h59**. Votre projet doit se présenter sous forme d'une archive `tar` compressée, appelée `vos_noms.tgz` qui doit contenir un répertoire appelé `vos_noms` (exemple : `dupont-durand.tgz`). Cette archive doit contenir le code du projet, des exemples et un rapport de deux ou trois pages expliquant les choix techniques, les extensions réalisées, les limitations du projet, etc. Le rendu du projet sera suivi d'une soutenance.