Une extension conservative du data-flow synchrone avec des machines à état

Marc Pouzet
LRI, Université Paris Sud, Orsay

Marc.Pouzet@lri.fr

Les langages de programmation synchrones, introduits dans les années 80, sont aujourd'hui utilisés dans divers domaines de l'informatique embarquée (avionique, transport terrestre, électronique grand public). Ils reposent sur l'idée de proposer des langages ayant un pouvoir expressif limité mais adaptés à la culture du domaine. En échange, ils offrent des garanties fortes de sûreté à la compilation (exécution en temps et mémoire bornée, absence de blocage ou déterminisme).

Ces langages se distinguent cependant par leurs styles de programmation. Les langages data-flow (e.g., SCADE/LUSTRE, SIGNAL) manipulent des suites infinies comme valeurs de base, les rendant naturellement adaptés à la description de systèmes réguliers (boucle de régulation, filtrage). Les langages impératifs (e.g., ESTEREL, SYNCCHARTS) au contraire, sont fondés sur un modèle de composition d'automates et s'avèrent donc mieux adaptés à la modélisation de systèmes ayant des changements de comportement fréquents. Cette distinction se retrouve également dans les outils de modélisation avec l'utilisation conjointe de SIMULINK pour les parties data-flow et de STATEFLOW pour la description du contrôle.

Dans cet exposé nous aborderons la question ancienne du rapprochement entre ces deux paradigmes en rappelant les solutions envisagées jusque là. Nous présenterons une solution permettant d'étendre un langage data-flow de type LUSTRE avec des structures impératives sous la forme d'automates hiérarchiques ou de conditions d'activation n-aires. L'extension proposée est *conservative* au sens où tous les programmes du langage de base sont valides et conservent la même sémantique. Il s'agit donc d'un alternative à l'utilisation conjointe d'outils différents tout en étant plus générale car elle autorise un mélange arbitrairement fin des deux styles de description au sein d'un langage unique.

L'extension proposée est fondée sur le mécanisme des *horloges* et une sémantique par traduction, les constructions impératives étant traduites vers des constructions avec horloges du noyau de base. Cette approche permet d'obtenir une sémantique uniforme pour le langage étendu et d'expliquer précisément l'interaction entre les contructions data-flow et les constructions impératives. Elle se révèle légère à mettre en oeuvre puisque le générateur de code existant peut être réutilisé en l'état. Les résultats expérimentaux montrent que le code obtenu est aussi efficace que les méthodes de compilation *ad-hoc*.

Nous discuterons des différents choix sémantiques envisagés et de l'extension des analyses de programmes existantes pour prendre en compte ces nouvelles constructions (e.g., calcul d'horloge, initialisation, causalité). Les principes proposés ont été mis en oeuvre dans le compilateur RELUC de SCADE/LUSTRE développé chez ESTEREL-TECHNOLOGIES et dans le compilateur de LUCID SYNCHRONE.

Ce travail a été réalisé en collaboration avec Jean-Louis Colaço, Grégoire Hamon et Bruno Pagano [2, 1, 3].

References

[1] Jean-Louis Colaço, Grégoire Hamon, and Marc Pouzet. Mixing Signals and Modes in Synchronous Data-flow Systems. In *ACM International Conference on Embedded Software (EMSOFT'06)*, Seoul, South Korea, October 2006.

[2] Jean-Louis Colaço, Bruno Pagano, and Marc Pouzet. A Conservative Extension of Synchronous Data-flow with State Machines. In *ACM International Conference on Embedded Software (EMSOFT'05)*, Jersey city, New Jersey, USA, September 2005.

[3] Grégoire Hamon and Marc Pouzet. Modular Resetting of Synchronous Data-flow Programs. In *ACM International conference on Principles of Declarative Programming (PPDP'00)*, Montreal, Canada, September 2000.