Compilation of Zero-crossing Detection in Languages for Hybrid Systems

Marc Pouzet
Marc.Pouzet@ens.fr

Timothy Bourke
Timothy.Bourke@ens.fr

Location: Département d'Informatique, École Normale Supérieure, 45 rue d'Ulm, 75230 Paris cedex 05.

Prerequisite: Interest in the design and implementation of programming languages, some experience with functional programming. Knowledge in numerical analysis is a plus but is not mandatory.

Collaboration: The team has close collaboration with the compiler group of SCADE at Esterel-Technologies and that of Modelica at Dassault-Systèmes.

Research Context

We address the compilation of a language for modeling hybrid systems that mix discrete time and continuous time behaviours. The most representative examples are Simulink/Stateflow,[1] a de-facto standard of embedded system industry with Modelica[2] as a new player. Despite their wide use, no comprehensive semantics nor formal description of their compilation exist. Several attempts have been made nonetheless in the recent years, either for their discrete subset [7, 8] or the mix of discrete and continuous time [5, 9, 4].

Our team is actively working on the topic with the purpose of defining a language and compiler based on strong semantical grounds and built above a synchronous language. Indeed, synchronous languages [1] are extensively used in the most critical control applications such as fly-by-wire control for planes, train tracking, or security system for energy plants. Several of these applications are programmed with SCADE.[3] Nonetheless, synchronous languages do not model continuous-time systems, expressed in particular by Ordinary Differential Equations (ODEs). Continuous-time is mandatory for modeling physical devices both faithfully and accurately (e.g., engine control).

In recent work, we have proposed a synchronous, Lustre-like language, extended with features for modeling continuous dynamics (ODEs and zero-crossings) [6]. Its semantics is based on non standard analysis [4]. The compilation is done through a source-to-source transformation into the synchronous subset which is then translated to sequential code using an existing compiler [3, 2]. This approach enables to reuse many existing techniques (e.g., compilation and static analysis) in order to generate efficient which is then paired with a black-box numerical solver (here SUNDIALS CVODE from LLNL [4]).

1 Project Description

The purpose of the intership is to work on the compilation problem raised by zero-crossing detection. In hybrid systems, a zero-crossing occurs when a continuous-time signal crosses zero (e.g.,

[1] http://www.mathworks.com/products/simulink with more than a million of licences
[2] https://www.modelica.org
[3] http://www.esterel-technologies.com
[4] https://computation.llnl.gov/casc/sundials/main.html

from negative to positive). This detection is performed by the numeric solver and is extremely costly. It is thus key to reduce the number of zero-crossings to detect to the minimum. This calls for the design of new compilation techniques. In particular, several zero-crossings can be merged together when they are proved to be exclusive from each other (e.g., when they appear in differents states of an automaton), or eliminated at an instant where a signal has a discontinuity. Moreover, this can be combined with the definition of a dedicated type system in the form of a *clock calculus* to record instants of discontinuities of a signal.

The intership will need both interest in language design, semantics, type systems and compilation. It should propose effective solutions that can be prototyped inside the Zelus compiler. The work can continue with a PhD. thesis, either in the group or funded by a CIFRE.

References

[1] A. Benveniste, P. Caspi, S.A. Edwards, N. Halbwachs, P. Le Guernic, and R. de Simone. The synchronous languages 12 years later. *Proceedings of the IEEE*, 91(1), January 2003.

[2] Albert Benveniste, Timothy Bourke, Benoit Caillaud, and Marc Pouzet. A Hybrid Synchronous Language with Hierarchical Automata: Static Typing and Translation to Synchronous Code. In *ACM SIGPLAN/SIGBED Conference on Embedded Software (EMSOFT'11)*, Taipei, Taiwan, October 2011.

[3] Albert Benveniste, Timothy Bourke, Benoit Caillaud, and Marc Pouzet. Divide and recycle: types and compilation for a hybrid synchronous language. In *ACM SIGPLAN/SIGBED Conference on Languages, Compilers, Tools and Theory for Embedded Systems (LCTES'11)*, Chicago, USA, April 2011.

[4] Albert Benveniste, Timothy Bourke, Benoit Caillaud, and Marc Pouzet. Non-Standard Semantics of Hybrid Systems Modelers. *Journal of Computer and System Sciences (JCSS)*, 78:877–910, May 2012. Special issue in honor of Amir Pnueli.

[5] Olivier Bouissou and Alexandre Chapoutot. An operational semantics for simulink's simulation engine. In *LCTES*, pages 129–138, 2012.

[6] Timothy Bourke and Marc Pouzet. Zélus, a Synchronous Language with ODEs, 2012 November. Submitted for publication.

[7] P. Caspi, A. Curic, A. Maignan, C. Sofronis, and S. Tripakis. Translating Discrete-Time Simulink to Lustre. *ACM Transactions on Embedded Computing Systems*, 2005. Special Issue on Embedded Software.

[8] Grégoire Hamon. A denotational semantics for stateflow. In *EMSOFT*, pages 164–172, 2005.

[9] Pieter J. Mosterman, Justyna Zander, Grégoire Hamon, and Ben Denckla. Towards computational hybrid system semantics for time-based block diagrams. In A. Giua, C. Mahulea, and M. Silva, editors, *3rd IFAC Conference on Analysis and Design of Hybrid Systems (ADHS'09)*, pages 376–385, Zaragoza, Spain, September 16-18 2009.