Design and Implementation of a Synchronous Language with ODEs

Master 2 Internship

November 2012

Level: M2R
Length: 5/6 months (spring 2012)
Advisors: 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: This internship is for a student with strong interest and skills in functional programming, the semantics and implementation of programming languages and reactive systems. Knowledge in numerical analysis is a plus.
Collaboration: The work can be continued with a PhD. thesis. The group has a close collaboration with several production teams, including the compilation group of Esterel-Technologies and the one of Modelica at Dassault-Systèmes.

Project description

Hybrid modeling tools [7] such as SIMULINK[1] are now *de facto* standard of embedded system design[2] and MODELICA[3] a new player. They allow to describe discrete time and continuous time systems so that the very same source code is used both for simulation/testing/formal verification and code generation. Despite the availability of such tools, there remain a number of issues related to their semantics and the lack of reproducibility of simulations. It is thus critical to place them on a firm semantical basis where it can be proven that the results of simulation, compilation and verification are mutually consistent.

Synchronous languages [1] are used for programming the most critical applications (e.g., fly-by-wire control of Airbus planes, braking systems for subways and trains). They have already addressed the above issues for a class of discrete real-time systems for which time- and resource-bounded code can be generated. Nonetheless, they cannot be used to model hybrid systems with both efficiency and precision. For that, it is important to be able to model, in the very same language, both discrete time systems and continuous times ones expressed by Ordinary Differential Equations (ODEs) or Differential Algebraic Equations (DAEs).

The problem we address here is the design and implementation of a hybrid synchronous language which combines the expressiveness of a synchronous language with ODEs for modeling physical elements evolving on a continuous-time basis. This extension must be conservative with respect to an existing synchronous language: it must preserve sequential code generation, an execution in bounded time and space for the discrete part as well as the guaranty on determinacy and the absence of deadlocks. Defining such a language has a tremendous number of applications: the design of control systems (planes, cars, etc); mixed (discrete/continuous) signals for modeling digital/analogous circuits; music (e.g., control/simulation of instruments), etc. Despite important developments on the verification of hybrid automata [8] (see [7] for a survey), there is little work on programming languages issues regarding, in particular, static typing, semantics and compilation.

[1] http://www.mathworks.com/products/simulink
[2] Over one million of licenses for SIMULINK/STATEFLOW.
[3] https://www.modelica.org

In recent work, we have proposed a synchronous, LUSTRE-like language, extended with features for modeling continuous dynamics (ODEs and zero-crossings). We based its semantics on non-standard analysis [5, 4]. This semantics clarifies interactions between continuous behaviors and discrete transitions in hybrid systems, like cascades of zero-crossing. 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]).

A first prototype of this language, called ZELUS, has been developed [6]. We are currently augmenting it with control structures, like hierarchical automata [2]. But, even so, the language lacks many features including, among others 1. a finer clock calculus and causality analysis to detect unbounded cascades of zero-crossing. 2. the ability to use multiple numerical solvers or to take execution time into account in order to compromise speed and accuracy (real-time simulation). 3. new optimization techniques in order to reduce the number of zero-crossing detections performed by the numeric solver.

The intership will investigate some of the above research problems. It will propose effective solutions and prototype them inside the ZELUS compiler. Interaction with the designers of MODELICA language will be done all along the intership.

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] Albert Benveniste, Benoit Caillaud, and Marc Pouzet. The Fundamentals of Hybrid Systems Modelers. In *49th IEEE International Conference on Decision and Control (CDC)*, Atlanta, Georgia, USA, December 15-17 2010.

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

[7] Luca Carloni, Maria D. Di Benedetto, Alessandro Pinto, and Alberto Sangiovanni-Vincentelli. Modeling Techniques, Programming Languages, Design Toolsets and Interchange Formats for Hybrid Systems. Technical report, IST-2001-38314 WPHS, Columbus Project, March 19 2004.

[8] Thomas Henzinger. The theory of hybrid automata. *NATO ASI Series F: Computer and Systems Sciences*, 170:265292, 2000. Springer-Verlag.

[4] https://computation.llnl.gov/casc/sundials/main.html