Static Analysis of Simulink Programs

Alexandre Chapoutot[1]

CEA, LIST - Laboratoire MeASI
Boîte courrier 94 F91191 Gif-sur-Yvette Cedex, France

Matthieu Martel[2]

Laboratoire ELIAUS-DALI
Univeristé de Perpignan Via Domitia
52, avenue Paul Alduy F66860 Perpignan Cedex, France

Design tools are needed to cope with the growth of complexity of embedded systems. Simulink^{TM}or Lustre/SCADE^{TM}[3] are the main industrial tools achieving this goal. But despite of the numerous features proposed by both tools, like simulation, test or code generation, Simulink is most of the time chosen because of its important system design expressiveness. It can model and simulate continuous time systems, discrete time systems or a mix of both. So in the case of embedded systems, it offers a convenient way to design an embedded software and its physical environment. The application of formal methods on such programs seems to be a good way to validate the high level specification of embedded systems.

We are working on defining a static analysis by abstract interpretation[2] of Simulink programs named *Abstract Simulation* (*A.S.*). In general, Simulink program executions are used to assess "good" program behaviors (called simulation phase in Simulink). The aim of *A.S.* is to provide a correctness criterion on the Simulink phase and then replace test activity by validation method. In order to handle continuous time parts and discrete time parts of Simulink programs, we have to design a static analysis manipulating different numerical domains: the domain of Taylor series for the continuous time one (inspired from [5]) and the domain of floating-points with errors [4] for the discrete time one. Moreover, in order to deal with dynamical systems, that is systems which evolve during time, we also use a close Kahn's sequence semantics [1] to represent the temporal evolution.

The property validated with our approach is depicted in Figure 1. We evaluate the *time sampling approximation* related to the error introduced by numerical integration algorithms (Simulink solver). These numerical algorithms are used in

[1] Email: alexandre.chapoutot@cea.fr
[2] Email: matthieu.martel@univ-perp

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

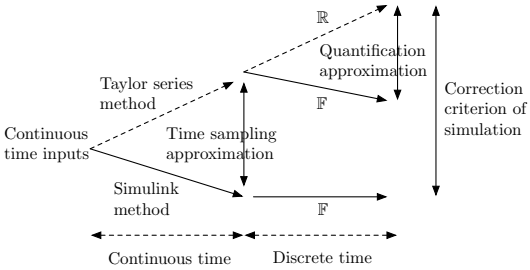
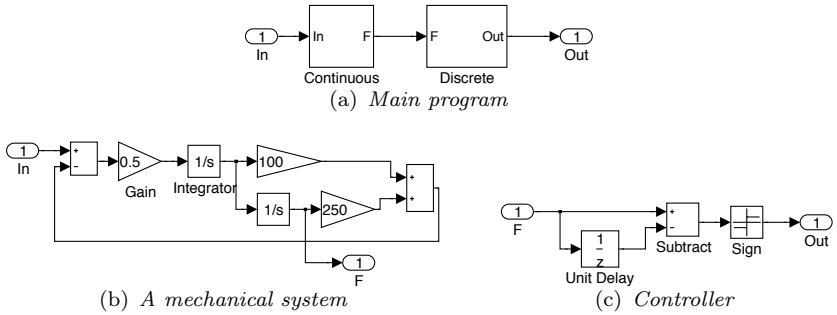


Fig. 1. *Static analysis property of Simulink programs*

the continuous time part of a Simulink program. This is done by comparing values given by the solver and values issued from a Taylor series method. Furthermore, the *quantification approximation* measures rounding errors introduced by floating point computations compared to real ones for the discrete time part. We use abstract interpretation in order to validate classes of Simulink programs and, consequently, a set of behaviors. We are currently implementing a prototype and experimental results should be available soon.

We represent in Figure 2 an example of Simulink program that we would like to analyze. The main program (Figure 2(a)) is a composition of one continuous time program representing a mechanical system: a mass-spring-damper (Figure 2(b)) and one discrete time program which computes the sign of the difference of two consecutive inputs (Figure 2(c)).



(a) *Main program*

(b) *A mechanical system*

(c) *Controller*

Fig. 2. *A Simulink program.*

References

[1] G. Kahn. The Semantics of a Simple Language for Parallel Programming. In *International Federation of Information Processing (IFIP'74)*, 1974.

[2] P. Cousot and R. Cousot. Abstract Interpretation: a Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Principles of Programming Languages (POPL'77)*, ACM, pages 238–252. ACM Press, 1977.

[3] P. Caspi, D. Pilaud, N. Halbwachs, and J. A. Plaice. LUSTRE: a Declarative Language for Real-Time Programming. In *Principles of Programming Languages (POPL'87)*, pages 178–188, New York, NY, USA, 1987. ACM Press.

[4] M. Martel. Semantics of Roundoff Error Propagation in Finite Precision Computations. *Journal of Higher Order and Symbolic Computation*, 19(1):7–30, 2004.

[5] A. Chapoutot and M. Martel. Différentiation automatique et formes de Taylor en analyse statique de programmes numériques (in French). In *AFADL'07, Approches Formelles dans l'Assitance au Développement de Logiciels*, 2007.