Proposition de recherche doctorale

A Programming Language for Cyber-Physical Systems

Mots clés :

- **Directeur de thèse** : marc POUZET
- **Co-encadrant(s)** :
- **Unité de recherche** : Laboratoire d'Informatique de l'École Normale Supérieure
- **Ecole doctorale** : École Doctorale Informatique, Télécommunications, Électronique de Paris
- **Domaine scientifique principal**: Divers

Résumé du projet de recherche (Langue 1)

Hybrid system modelers have become a corner stone of complex embedded system development. Embedded systems include not only control components and software, but also physical devices. In this area, Simulink/Stateflow is a de facto standard design framework and {Modelica} a new player. However, such tools raise several issues related to their semantics and the lack of determinacy and reproducibility of simulations (sensitivity to simulation parameters and to the choice of a simulation engine). 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 are used for programming the most critical applications (e.g., fly-by-wire control of Airbus planes, braking systems for subways and trains, designed and implemented with the language SCADE ([http://www.esterel-technologies.com/products/scade-suite/]). 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. The are founded on a precisely defined semantics with automatic code generation, formal verification and testing tools. Nonetheless, they lack the ability to model faithfully and efficiently hybrid systems that mix control software and a model of the plant. For that, it is important to express continuous-time dynamics (by mean of differential equations) interwined with control-software. The problem we address here is the design and implementation of a language for hybrid systems which combines the expressiveness and safety of a synchronous language such as {Lustre} or {Lucid Synchrone} for programming discrete time systems with ODEs for describing physical elements evolving on a continuous-time basis. We expect this extension to 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 simulation of a complete systems including physical devices and software; 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 (Henzinger et al., 2000) for a survey), there is less work on programming languages issues regarding, in particular, static typing, semantics and compilation. Pionneering works are those by Ramine Nikoukhah on Scicos ([http://www.scicos.org]) and Edward Lee on Ptolemy ([http://ptolemy.berkeley.edu/ptolemyII/]. In previous works, we have shown how to extend an existing {Lustre}-like language with ODEs and zero-crossing mechanism to model both discrete and continous behavior and relying on a numerical black box solver. We defined an ideal semantics on non-standard analysis (CDC 2010, JCSS 2012) as a natural extension of the data-flow semantics of Lustre. 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 (LCTES'11, EMSOFT'11). This approach enables to reuse and extend many existing techniques like the causality analysis (HSCC'14) and compilation techniques in order to generate efficient which is then paired with a black-box numerical solver (here SUNDIALS CVODE from LLNL ([https://computation.llnl.gov/casc/sundials/main.html]). These results has been the foundation of the language Zélus (HSCC'13) [http://zelus.di.ens.fr]. Zélus is functional with type inference and polymorphism, mix of data-flow, hierarchical automata and Ordinary Differential Equations (ODEs). Based on the principles experimented in Zélus, an extension of SCADE KCG 6.5 (KCG is the the {qualified code generator} of SCADE developed by ANSYS/Esterel-Technologies and used by more than 250 companies for programming the most critical control software in planes, trains, energy, etc.) We are now investigating the question of 1. A more expressive type system to separate continous-time from discrete-time signals and analyse causality relations between signals; 2. combining different solvers with the possibility of doing real-time simulation; 3. considering the more general case of semi-explicit DAEs --- ODEs with algebraic constraints --- instead of ODEs only. The purpose of the PhD. thesis is to address those questions, to propose theoretical solutions (language design and semantics, type systems, causality analysis, compilation techniques), to integrate them into a compiler and to validate them on real-size examples. It is expected that the solution are general enough to apply on a large subset of applications written with tools such as \simulink/\stateflow{} and \modelica. -#{{{Typing discrete and continuous-time signals}}} The mix of continous-time and discrete-time signals must be done such that the behavior does not depend on internal decisions made by the solver. A first solution has been introduced in LCTES'11. It essentially amount at separating expressions into three kinds: expressions are either combinatorial, discrete or continuous. An interesting question is the definition of a more expressive expressive type system able to distinguish piece-wise continuous, piece-wise constant, discrete time signals and periodically sampled signals. The existing clock calculus of synchronous languages (e.g., the one of Lucid Synchrone used in SCADE 6) is a good basis to start with. -#{{{Combination of solvers}}} Existing modelers such as Simulink and Modelica, use a single numerical solver for approximating continuous trajectories. Combining several solvers is the right way to achieve both precision and fast simulation for the whole system. The problem is reminiscent to the problem of {automatic code distribution} for synchronous languages. We propose to consider language annotations in order to define parts which are approximated by the same black-box solver, to study causality constraints that must be verified between them and to propose a semantics that deal with multi-solvers. -#{{Semi explicit DAEs}} Tools like {Modelica} can manage more general implicit (or acausal) models defined by Differential Algebraic Equations (DAEs). A particular class are semi-explicit DAE, i.e., a ODE ($\dot{x} = f(y, t, x)$) together with an algebraic constraint ($g(y,t,x) = 0$). The goal is to study the combination of semi-explicit DAE with control structures (hierarchical automata) to express modes. This raises semantical issues (what is the semantics of the whole language), compilation issues (how to prepare the code so that it can be linked with existing black-box solvers) and optimization issues.

Résumé du projet de recherche (Langue 2)

To define and implement a language to model systems with mixed (continuous/discrete) signals and based on strong semantical grounds. The goal is to be able to write with a unique source code both a (discrete) controller and its (continuous) physical environment, and be able to simulate, test it and generate correct-by-construction target code.

Informations complémentaires (Langue 1)

-* The PARKAS teams has several collaborations with teams developing models, languages and analysis techniques for embedded software (e.g., Mary Sheeran group at Chalmers Univ., Gotheborg; Stephen Edwards at Columbia Univ., NY; Ed Lee at Berkeley). -* All questions addressed in this PhD. thesis interest directly two of our close industrial partners, ANSYS/Esterel-Technologies (SCADE) and Dassault-Systèmes (Modelica).

Informations complémentaires (Langue 2)

This PhD. thesis is for a student with strong interest and skills in functional programming, the semantics and implementation of programming languages, formal methods, reactive systems.