Static Analysis of Embedded Programs with Continuous I/O.

HSCC07 Hybrid Systems Computation and Control

Olivier Bouissou **Matthieu Martel**

CEA, LIST, Gif-sur-Yvette, F-91191 France
{olivier.bouissou,matthieu.martel}@cea.fr

Abstract

The validation of embedded programs requires that we compute all their possible executions. However, such programs usually interact with their environment in two ways: their inputs come from the discretization of a continuous function via sensors and their outputs modify the dynamics of these functions via actuators. Thus, their executions strongly depend on the physical environment in which they are run. Therefore, good results can only be obtained if one considers the program as the discrete part of a more general system, in which the continuous dynamics is taken into account. This poster presents our work on the analysis of such hybrid systems. We chose to split our analysis into two parts: an analysis of the continuous system via validated integration and an analysis of the discrete system via abstract interpretation techniques. This approach may be used for industrial systems as it does not require big changes of the existing codes.

Embedded programs as Hybrid systems

We are interested in the **verification and validation** of **industrial** systems. These are generally composed of two distinct parts: a discrete embedded program and the continuous physical environment surrounding it. Each subsystem has its own particularities and modelling each requires a **good expertise of the domain** (see Figure 1). Therefore, we try to analyse these systems without using models where the discrete and continuous worlds are mixed.

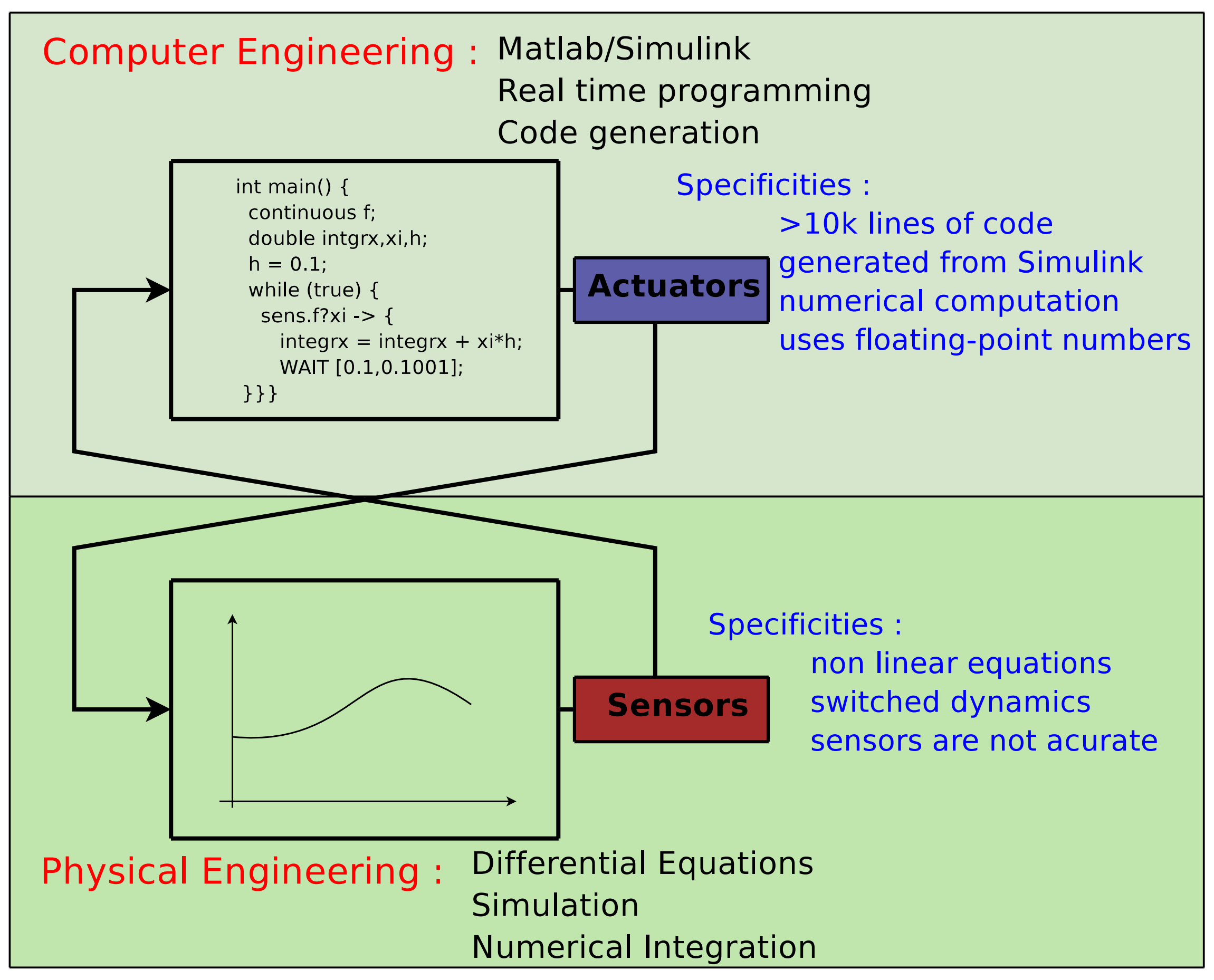

Figure 1: A hybrid system

We thus proposed [1] a model for hybrid systems in which we basically add hybrid actions (like sensors and actuators) to an imperative language. The program is then coupled to a description of the continuous physical environment. We base our analysis on this model.

Goal of the analysis

We want to prove that the **uncertainties** due to the implementation of the system do not affect its behaviour. These uncertainties are due first to the imprecision of the **numerical computations in floating-point numbers**, but also to the **imprecision of the sensors**. In addition, the time at which the sensors and/or actuators are activated is in practice unknown, which introduces new potential errors. Our analysis aims at proving that the differences between the implementation where all these errors occur and the ideal model with perfect sensors/actuators and real numbers computations do not modify the behaviour of the system (see Figure 2).

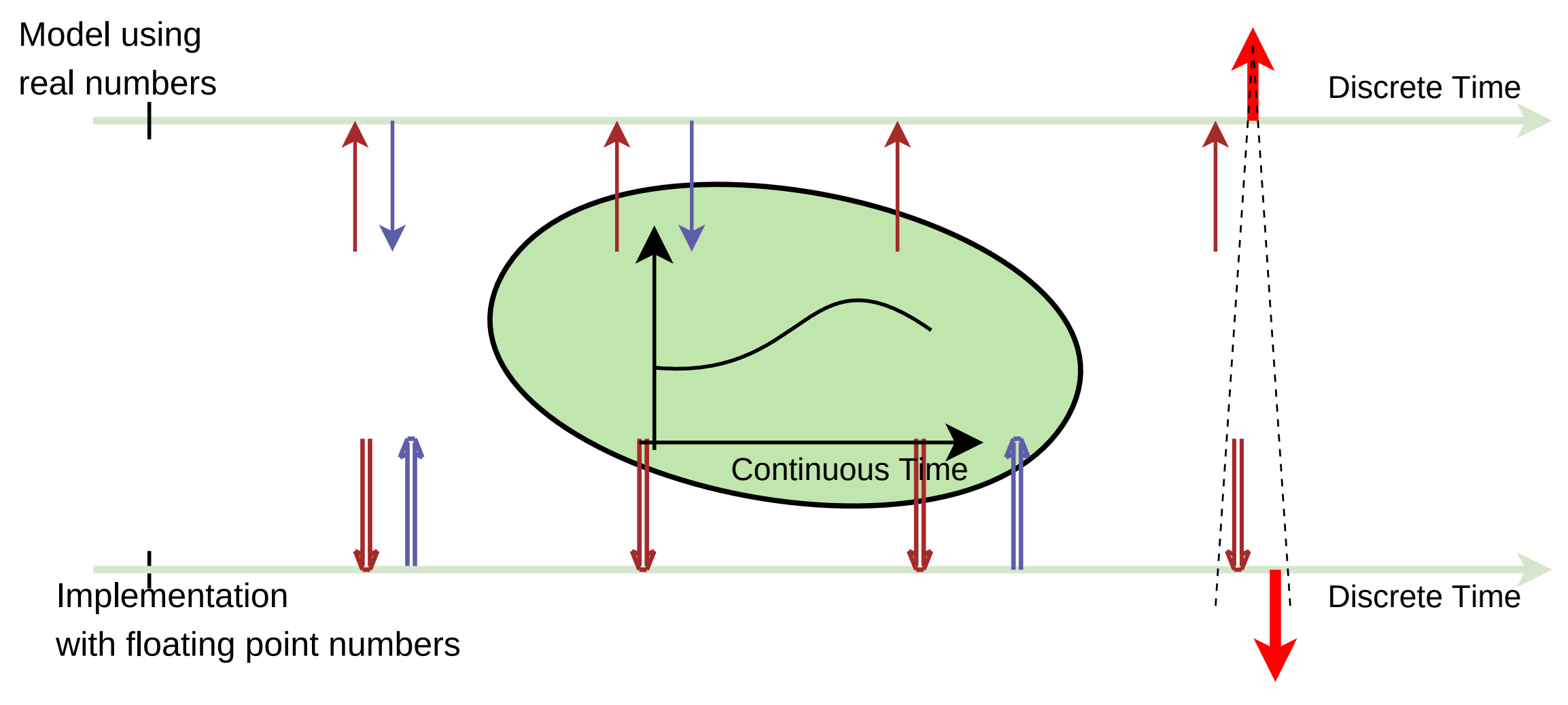

Figure 2: Actions taken by the ideal model (top) and by the implementation (bottom)

We thus only focus on some actions: sensors (dark red arrows) and actuators (blue) activations, as well as alarms (red), and we show that these actions occur almost at the same time in both cases.

Static analysis and Abstract interpretation

Validating programs means proving that they will **never enter unsafe regions**. However, as the inputs of most programs are unknown, it is generally not possible to test all the possible executions (Figure 3 shows some of them).

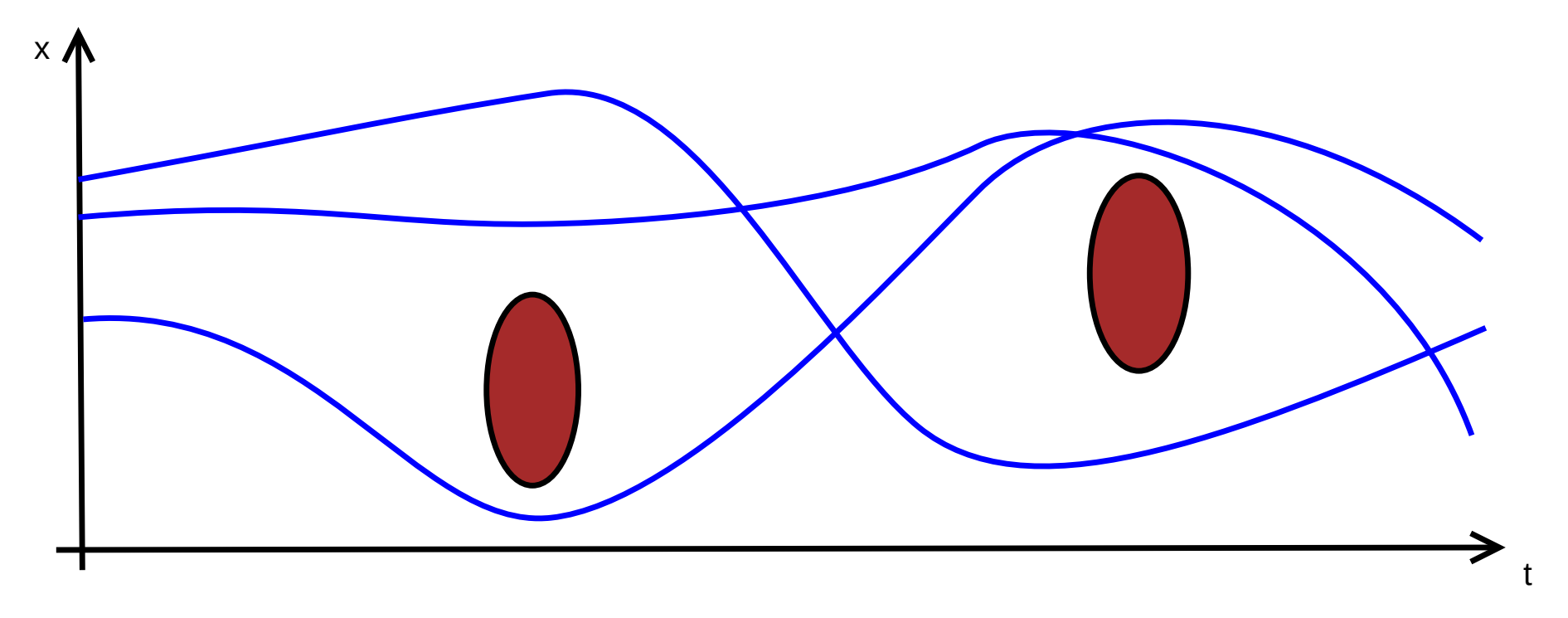

Figure 3: Some executions of a program.

The idea of the abstract interpretation theory [3] is to compute a set which is guaranteed to contain **all the possible values** taken by a variable during the execution of the program. Thus, if this set does not intersect with the unsafe regions, then so does the real execution of the program. However, if the set enters a forbidden state, we conclude whether the error is inherent to the program or if the computed set is too large.

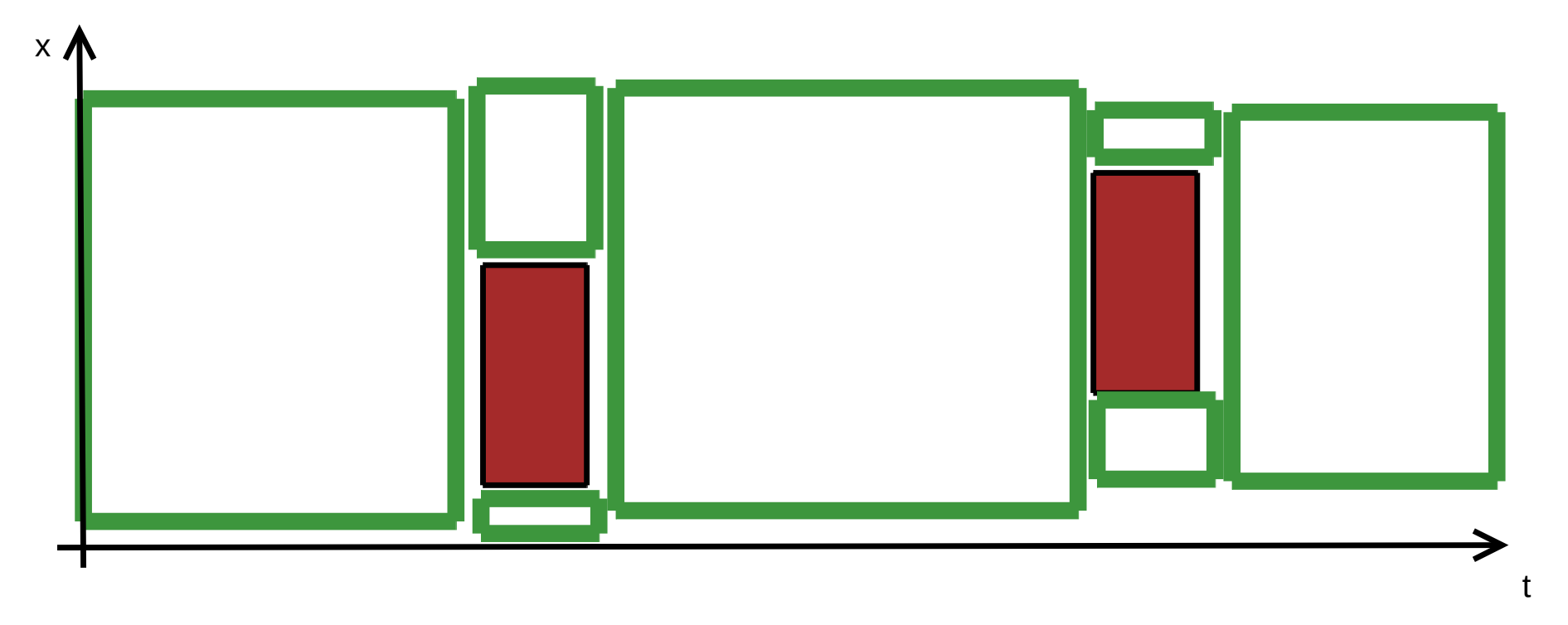

Figure 4: The abstract values of x along the program.

First results

Discrete World

The difficulty in the analysis of the discrete system is that we must take **time** into account. Thus, the semantics of the program will associate to every variable not a value but a **signal**, i.e. a function of time with non-continuous jumps (blue lines on Figure 5). A jump of such a signal may affect the continuous environment by changing its dynamics (blue arrow).

Our abstract domain unites all the equivalent paths into one where the time of the jump is no longer exactly known (green rectangles). The result of this time uncertainty is that the continuous dynamics changes at an unknown time, and we thus need to compute two functions that enclose all the possible curves (green arrows).

Continuous World

The continuous part of an hybrid system is modeled as a **family of differential equations**. These equations define functions which are the inputs of embedded programs. The abstract interpretation framework requires that we compute overapproximations of these functions. Therefore, we developed a library for the guaranteed integration [4] of autonomous differential equations. The method we propose is based on a classical Runge-Kutta integration scheme, where all kinds of errors (methodological and numerical) are safely overapproximated [2].

We then use this method to compute two step functions which over- and under-approximate a set of implicitly defined functions (see Figure 6).

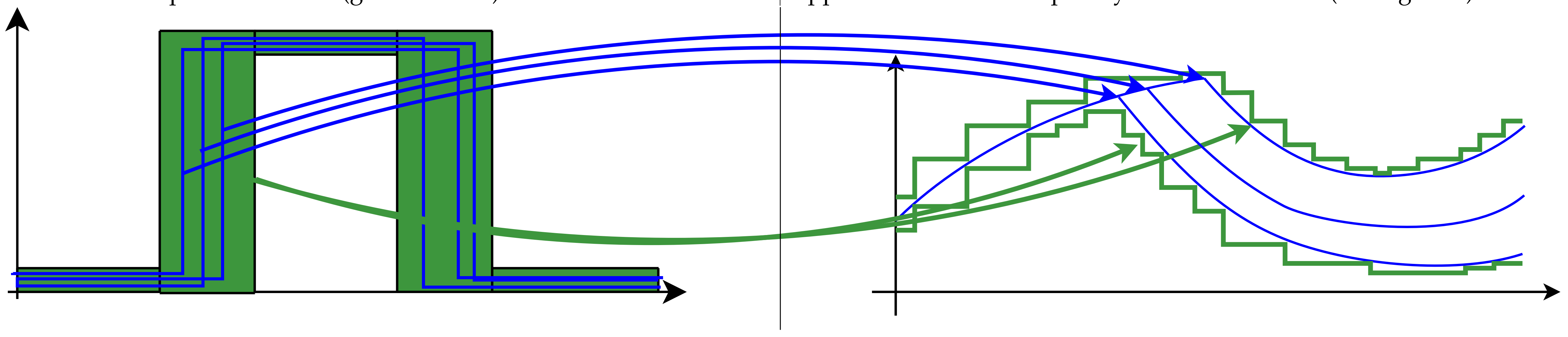

Figure 5: Concrete (blue) and abstract (green) signals.

Figure 6: The concrete continuous dynamics (blue) and its approximation by step functions (green).

References

[1] O. Bouissou. Analyse statique par interpretation abstraite de systèmes hybrides discrets-continus. Technical Report 05-301, CEA-LIST, 2005.

[2] O. Bouissou and M. Martel. A runge-kutta method for computing guaranteed solutions of odes. In *SCAN*, 2006.

[3] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, 1977.

[4] N. S. Nedialkov, K. R. Jackson1, and G. F. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105(1), 1999.

Work in Progress

- Continuous analysis: validated integration ✓
- Discrete analysis: we need better widening operators on our abstract domain ?
- Hybrid analysis:
  - coupling discrete and continuous analysis ✓
  - expressing validated integration as an application of abstract interpretation ✗
  - extending our analysis to a wider class of continuous functions ✗