Simulation of Hybrid Systems based on Hierarchical Interval Constraints

Poster Abstract

Daisuke Ishii, Kazunori Ueda
Waseda University
3-4-1, Okubo, Shinjuku-ku
Tokyo, Japan
{ishii, ueda}@ueda.info.waseda.ac.jp

Hiroshi Hosobe
National Institute of Informatics
2-1-2, Hitotsubashi, Chiyoda-ku
Tokyo, Japan
hosobe@nii.ac.jp

ABSTRACT

We propose a framework called HydLa for simple modeling and reliable simulation of *hybrid systems* which involve discrete and continuous changes over time. HydLa employs *interval constraints* as a central principle to express uncertainties in modeling, error bounds in the computation of nonlinear continuous changes, and reachable state sets that play key roles in verification. In this research, we propose a modeling language with *hierarchical* interval constraints to facilitate *well-defined* modeling, and its implementation which uses machine-representable interval constraints to enclose computation errors with intervals or boxes. The implementation is based on the integration of a consistency technique for nonlinear interval constraints and a technique for solving ordinary differential equations. We also present a method for solving constraint hierarchies among interval constraints.

Categories and Subject Descriptors

D.3.4 [**Programming Languages**]: Processors; I.6.4 [**Simulation and Modeling**]: Model Validation and Analysis

General Terms

Simulation, languages

Keywords

Hybrid systems, interval arithmetic, constraint hierarchies

1. INTRODUCTION

Hybrid systems are systems consisting of discrete and continuous changes over time. An example of hybrid systems is a "bouncing particle" illustrated in Figure 1, which falls down by gravity (i.e. continuous change) and bounces off a sinusoidal surface (i.e. discrete change). Various languages such as hybrid automata and Hybrid CC [4] have been used for modeling hybrid systems (See [3] for a survey). We aim to develop a language that allows us to describe large systems with complex laws concisely. From model descriptions, various model checkers for hybrid systems compute continuous states rigorously. However, those methods still have the limitations in the handling of nonlinear models.

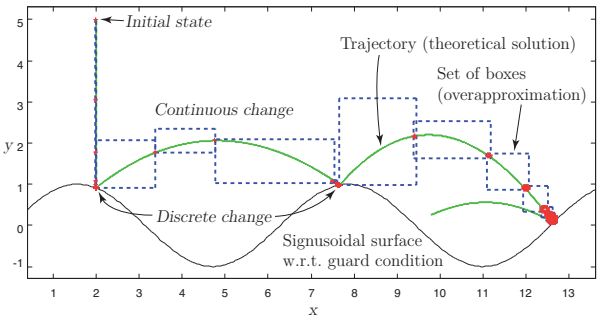

Figure 1: An execution result of a bouncing particle.

We propose a framework called HydLa for modeling and simulating hybrid systems. HydLa employs *interval constraints* as a central principle to express uncertainties in modeling, error bounds in the computation of nonlinear continuous changes, and reachable state sets that play key roles in verification. Another important feature is *constraint hierarchies* [2] to facilitate the construction of *well-defined* models, which we have found is often difficult. In HydLa, a model composes of constraints. In the particle example, simultaneously activating constraints corresponding to the gravity law and collisions will result in an over-constrained system implying an error. On the other hand, to model switching of several sets of constraints is often difficult; Modeling as a under-constrained system, a location of the particle may becomes undefined when a discrete change occurs.

This paper presents an implementation of HydLa for obtaining overapproximation of trajectories, that is, sets of intervals or boxes enclosing every possible execution of the model. The implementation employs techniques for solving nonlinear constraints and ordinary differential equations (ODEs) by enclosing solutions with tight intervals. These techniques reliably handle machine-representable intervals which have floating-point numbers as its boundaries; every computation error is enclosed with the intervals. Although

SIMUTools 2009 Rome, Italy

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5640
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5640

$$
\begin{aligned}
\text{INIT} &\Leftrightarrow x = (2,5) \wedge x' = (0,-5).\\
\text{LAWS} &\Leftrightarrow \Box(g \geq 9.8 \wedge g \leq 9.81 \wedge k = 0.3 \wedge e = 0.5).\\
\text{FALL} &\Leftrightarrow \Box(x_1'' = -g - k \cdot x_1').\\
\text{BOUNCE} &\Leftrightarrow \Box(x_{1-} = \sin(x_{0-}) \Rightarrow\\
&\qquad m = (-\sin'(x_{0-}), 1) \wedge n = m/\|m\|\\
&\qquad \wedge x' = e \cdot (x'_- - 2 \cdot n \cdot \langle n, x'_- \rangle)).\\
&\text{INIT},\ \text{LAWS},\ (\text{FALL} \ll \text{BOUNCE}).
\end{aligned}
$$

Figure 2: A bouncing particle model described by HydLa language.

symbolic computation techniques handle a limited class of hybrid systems, the proposed implementation allows models involving nonlinear constraints for both discrete and continuous changes. Furthermore, verification of various properties such as "will the particle fall into a hole?" and "how long will the particle stay in a designated area?" would be done by assigning intervals to variables in a model.

2. HIERARCHICAL INTERVAL CONSTRAINTS

Figure 2 is a model of the bouncing particle described in the HydLa language with *hierarchical interval constraints*. The model expresses a *trajectory* that is a vector of real-valued functions over time. Although the acceleration of the particle is governed by the gravity law most of the time, the velocity is determined by an equation of collision when the particle collides with the ground. HydLa manages hierarchies of constraints to determine which constraint to be activated in each point of time.

In Figure 2, x and x' are real vector-valued variables and x_1 denotes the first element of the vector x. The model consists of four user-defined modules and the composition of the modules using the binary composition operators with or without priority ("$\ll$" and "," respectively). Modules are described in terms of general mathematical notations (given an expression E, E' denotes the derivative, and E_- denotes the left-hand limit value at the time of a discrete change). The implication operator $\Rightarrow$ in the fourth statement gives a guard for a discrete change. The *always* operator $\Box$ activates a constraint to be satisfied on and after time 0.

3. IMPLEMENTATION

In the following, our implementation of HydLa for simulating models based on interval arithmetic is described. For the bouncing particle, the proposed implementation computes a set of boxes as shown in Figure 1 that encloses every possible trajectory of the model.

A trajectory is computed by alternating *point phases* (i.e. computation of an initial state or a discrete change) and *interval phases* (i.e. computation of continuous changes) as in previous methods [3, 5]. One of the difficulties in the computation of interval enclosures is the handling of discrete changes. An enclosure of a state where a discrete change will occur is computed by applying a method [5] that integrates a consistency technique for interval constraints [1] and an interval-based technique for solving ODEs [6]. In an interval phase, the method divides and prunes a continuous state space to search for a tight enclosure of a state that will trigger discrete change, i.e. the next point phase.

Constraint modules in a model form a partially ordered set. In each phase, a downward-closed set of modules (a module with higher priority comes lower), which is maximal and consistent, is computed as follows. First, a set M of downward closures of each module is created from the model. A constraint store $S := \emptyset$ is also prepared. Then, a maximal consistent set of modules in M is greedily searched. Computation is done by iteratively adding a constraint set in a closure in M to S and checking consistency. If a closure is inconsistent with S, then the closure is skipped (the algorithm will be optimized in various ways).

The check of consistency in the above is determined by checking if the existence of a unique solution is proved by the *interval Newton method*. Our modified version of the interval Newton method works for a problem consisting of nonlinear constraints and an ODE under certain conditions. The modification is based on the underlying solvers [1, 6], each proving that a box encloses a solution of a guard condition or a trajectory. If the above conditions do not hold for a problem, the implementation tries to avoid a singular point by dividing the initial domain.

The proposed implementation generally does not handle exact solutions. However, the implementation is able to compute intervals with a specified accuracy by repeatedly dividing and reducing intervals. When several guard conditions intersect within an interval enclosure, or when an initial interval value is too wide for a property to be verified, dividing of the intervals will be effective.

4. FUTURE WORK

Experiments on the implementation with large examples are needed. The examples will include a model with a complex law where dividing of intervals be effective, and a model composed of several sub-systems.

5. REFERENCES

[1] F. Benhamou, D. McAllester, and P. van Hentenryck. CLP(intervals) revisited. In *Proc. of ISLP*, pp. 124–138. MIT Press, 1994.

[2] A. Borning, B. Freeman-Benson, and M. Wilson. Constraint hierarchies. *Lisp and Symbolic Computation*, Vol. 5, No. 3, pp. 223–270, 1992.

[3] L. P. Carloni, R. Passerone, A. Pinto, and A. L. Angiovanni-Vincentelli. Languages and tools for hybrid systems design. In *Foundations and Trends in Electronic Design Automation*, Vol. 1, pp. 1–193. Now Publishers, 2006.

[4] B. Carlson and V. Gupta. Hybrid cc with interval constraints. In *Proc. of HSCC*, Vol. 1386 of *LNCS*, pp. 80–95. Springer-Verlag, 1998.

[5] D. Ishii, K. Ueda, and H. Hosobe. An interval-based approximation method for discrete changes in Hybrid cc. In *Trends in Constraint Programming*, pp. 245–255. ISTE, 2007.

[6] N. S. Nedialkov, K. R. Jackson, and G. F. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, Vol. 105, No. 1, pp. 21–68, Elsevier, 1999.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2009.5640
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5640