Computing the Evolution of Hybrid Systems using Rigorous Function Calculus

Pieter Collins * Davide Bresolin ** Luca Geretti ***
Tiziano Villa **

* Maastricht University, Maastricht, The Netherlands
(e-mail: pieter.collins@maastrichtuniversity.nl)
** Università degli Studi di Verona, Verona, Italy
(e-mail: davide.bresolin,tiziano.villa@univr.it)
*** Università degli Studi di Udine, Udine, Italy
(e-mail: luca.geretti@uniud.it)

Abstract: Hybrid systems exhibit all the complexities of finite automata, nonlinear dynamic systems and differential equations, and are extremely difficult to analyze. A rigorous mathematical approach is needed to achieve provable approximation bounds along the computations. In this paper we describe a rigorous numerical calculus for working with functions that can be used for computing the evolution of nonlinear hybrid systems, and the implementation in the tool Ariadne for reachability analysis of hybrid systems. The method is based around expressing the sets attained during the evolution in terms of functions, and computing approximations to these functions, and allows highly accurate approximations for the evolved sets to be computed. An example of the control of the water level in a tank is presented.

Keywords: Hybrid automata; Numerical Tools; Analysis; Reachability; Verification.

1. INTRODUCTION

Hybrid systems are dynamic systems in which continuous evolution is interspersed with discrete events triggered by conditions on the continuous state. They typically occur in applications in which digital sensors and actuators interact with the physical world, but can also be used to model purely physical phenomena such as impact systems or electrical circuits with diodes and switches. The analysis of hybrid systems is extremely difficult, since hybrid systems exhibit all the complexities of finite automata, nonlinear dynamic systems and differential equations, and additionally have discontinuities and singularities in the evolution due to the switching.

There are many software tools for reachability analysis and/or verification of hybrid systems, mostly restricting to a subclass of system. The Ariadne software package is an ambitious attempt to provide analysis and verification tools for general nonlinear hybrid systems. The original functionality, as described in Benvenuti et al. [2008], was based on affine approximations of the continuous dynamics, and while being fast and reliable, was not accurate enough for strongly nonlinear systems, especially when making discrete transitions. A new computational kernel was subsequently developed, based on rigorous numerical methods for working with real numbers, functions and sets in Euclidean space, using techniques such as interval arithmetic, automatic differentiation and polynomial function models as a foundation. The purpose of this paper is to describe the low-level operations on sets and functions that have been used to build up the high-level algorithms for the analysis of hybrid systems in Ariadne.

The paper is organised as follows. In Section 2, we describe the class of hybrid system we work with, and in Section 3 we describe the evolution in terms of functions and constraints. In Section 4, we describe relevant existing rigorous numerical methods, and in Section 5 we describe the function calculus as implemented in Ariadne. In Section 6 we describe how the function calculus is used in Ariadne to compute the evolution of hybrid systems. We give an example of controlling the level of water in a tank in Section 7, and in Section 8 we give a comparison with other tools for analysing hybrid systems. Concluding remarks are given in Section 9.

2. HYBRID SYSTEMS

In this section we describe a simple framework for nonlinear hybrid systems.

A hybrid system is a tuple

\[ H = (Q, E, T, (n_q, f_q)_{q \in Q}, (p_{q,e}, a_{q,e}, r_{q,e})_{q,e \in \text{dom}(T)}) \]

where:

- \( Q \) is a finite set of locations,
- \( E \) is a finite set of events,
- \( T : Q \times E \rightarrow Q \) is the transition map,
- \( f_q : \mathbb{R}^n_e \rightarrow \mathbb{R}^n_q \) is the continuous dynamics,
- \( p_{q,e} : \mathbb{R}^n_q \rightarrow \mathbb{R}^m_q \) is a progress function,
- \( a_{q,e} : \mathbb{R}^n_q \rightarrow \mathbb{R} \) is an activation function,
- \( r_{q,e} : \mathbb{R}^n_q \rightarrow \mathbb{R}^{\mathbb{Z}^+} \) is the reset map.

* This research was partially supported by the European Commission through the project FP7-ICT-223844 CON4COORD and by the Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO) Vidi grant 639.032.408.
The state space is the set $\bigcup_{q \in \mathbb{Q}} \{q\} \times \mathbb{R}^n$.

A trajectory of a hybrid system $H$ comprises continuous evolution interspersed with discrete events. The continuous dynamics in location $q$ is governed by the differential equation $\dot{x} = f_q(x)$. When an event $e$ occurs, the location transitions to $q' = T(q, e)$, and the continuous state is updated by the reset into $x' = r_{q,e}(x)$. The event $e$ may only occur if the activation condition $a_{q,e}(x) \geq 0$ holds, and continuous evolution may only occur while the progress conditions $p_{q,e}(x) \leq 0$ hold for all events $e$. If $p_{q,e} = a_{q,e}$, then the event is urgent (in location $q$), and we combine the progress condition and activation condition into a guard condition $g_{q,e}(x) = 0$.

We let $\Xi_H$ denote the set of trajectories of $H$, and $\Xi_H(X_0, T)$ the set of points attained by all trajectories starting from a point $x_0 \in X_0$ at a time $t \in T$. The fundamental problem of hybrid systems is to compute the set of trajectories $\Xi$, and the finite-time reachable set $R = \Xi_H(X_0, [0, t])$ and evolved set $E = \Xi_H(X_0, t)$.

Due to results of [Collins and Lygeros, 2005, Collins, 2011], these sets cannot always be computed to arbitrary accuracy, but only lower- or over-approximations may be computed, depending on the exact semantics of evolution.

3. EVOLUTION OF A HYBRID SYSTEM

In this section, we give a mathematical overview of the evolution of a hybrid system.

The continuous state reached at time $t_f$ starting at time $t_0$ from point $x_0 \in X_0$ with a single event $e_1$ occurring at time $t_1$ with reset $r_1$ is

$$x_f = \phi_1(x_0, t_1) = f_{q_1}(q_1(x_0, t_1), t_f - t_1).$$

Here $\phi_1 : \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^m$ is the flow of the differential equation $\dot{x} = f_q(x)$ for location $q_1$, and satisfies

$$\dot{\phi}_1(x, t) = f_q(\phi_1(x, t), t); \quad \phi_1(x, 0) = x.$$ 

If $a_1(x) \geq 0$ is the activation condition for the event $e_1$ in location $q_0$, then

$$a_1(\phi_0(x_0, t_1, t_l - t_0)) \geq 0,$$

yielding a constraint on the event time $t_l$. If $p(x) \leq 0$ is a progress condition in location $q_0$, then

$$\sup_{t \in [t_0, t_1]} p(\phi_0(x_0, t, t_l - t_0)) \leq 0.$$ 

The set of points attained at time $t_f$ starting from a point in $X_0$ after a non-urgent event at time $t_1$ with activation condition $a_1 \geq 0$ and progress conditions $p_1 \leq 0$ and $p_2 \leq 0$, assuming there are no other possible events, can be represented as

$$E = \{\phi_1(x_0, t_1, t_l - t_0), t_f - t_l \mid \phi_0(x_0, t, t_l - t_0) \in X_0 \times [0, t_f] \mid a_1(\phi_0(x_0, t_1, t_l - t_0)) \geq 0 \wedge \sup_{t \in [t_0, t_1]} p_1(\phi_0(x_0, t, t_l - t_0)) \leq 0 \wedge \sup_{t \in [t_0, t_1]} p_2(\phi_0(x_0, t, t_l - t_0)) \leq 0\}.$$ 

This is the most general form for the evolved set, but the presence of the conditions $\sup_{t \in [t_0, t_1]} p_1(\phi_0(x_0, t, t_l - t_0)) \leq 0$ makes this class of set very hard to work with. We therefore seek conditions under which we can simplify the representation of the evolved set $E$.

The rate of change of $p(x(t))$ along solutions of $\dot{x} = f(x)$ is given by the Lie derivative

$$L_f p := \nabla p \cdot f.$$ 

The crossing of the flow line $\phi(x_0, t)$ with the progress set boundary $p(x) = 0$ is transverse if

$$\frac{\partial}{\partial x} p(\phi(x_0, t)) \neq 0 \text{ when } p(\phi(x_0, t)) = 0.$$

We see that crossings are transverse if

$$\nabla p \cdot f \neq 0 \text{ whenever } p(y) = 0.$$

In this case, the progress constraint reduces to $p(\phi(x_0, t_1 - t_0)) \leq 0$.

If the event $e$ is urgent, then the guard constraint

$$g(\phi(x_0, t_1 - t_0)) = 0 \wedge \sup_{t \in [t_0, t_1]} g(\phi(x_0, t, t_l - t_0)) \leq 0$$

reduces to

$$g(\phi(x_0, t_1 - t_0)) = 0.$$ 

Further, the crossing time $t_1$ can be computed as a function $t_1 = \gamma(x_0)$. Note that the assumption that the event occurs at a time $t_2$ in $[t_0, t_f]$ implies range($\gamma$) $\subset [t_0, t_f]$.

If $e_1$ is urgent and crossings of $g_1$ and $p_2$ are transverse, we therefore obtain the following forms for the evolved set:

$$E = \{\phi_1(r(\phi_0(x_0, t_1 - t_0)), t_f - t_l) \mid (x_0, t_1) \in X_0 \times [0, t_f] \mid [g_1(\phi_0(x_0, t_1 - t_0)) = 0 \wedge p_2(\phi_0(x_0, t_1 - t_0)) \leq 0]\}
$$

$$= \{\phi_1(r(\phi_0(x_0, \gamma_1(x_0) - t_0)), t_f - \gamma_1(x_0)) \mid x_0 \in X_0 \mid p_2(\phi_0(x_0, \gamma_1(x_0) - t_0)) \leq 0\}.$$ 

A grazing contact is a quadratic tangency of the flow line with the progress set boundary, and is characterised by

$$L_f^2 p(y) < 0 \text{ whenever } L_f p(y) = p(y) = 0.$$ 

In this case, we can compute the critical time $\mu(x_0)$ at which the $p(\phi(x_0, t))$ reaches a maximum. The progress condition then reduces to

$$(t_1 \leq \mu(x_0) \wedge p(\phi(x_0, t_1 - t_0)) \leq 0) \vee (t_1 \geq \mu(x_0) \wedge p(\phi(x_0, \mu(x_0) - t_0)) \leq 0).$$

Note that the condition $t_1 \geq \mu(x_0)$ is unnecessary.

In the neighbourhood of a point where the flow has a cubic or higher-order tangency with the progress set boundary i.e. if $p(y) = 0$, $L_f p(y) = 0$ and $L_f^2 p(y) = 0$, it is possible in principal to find formulae for the evolved sets using higher-order singularity theory, but these quickly become unwieldy. However, when computing over-approximations to the evolution, we can always use the fallback conditions $p(\phi(x_0, t_1 - t_0)) \leq 0$.

The evolved set $E$, or in the presence of singular crossings, an under- or over-approximation of $E$, can therefore be represented as a constrained image set having the form

$$S = \{f(z) \mid z \in D \mid g(z) \leq 0 \wedge h(z) = 0\}.$$ 

Here, $z$ is the parameter, $D = [d_1, d_2] \times \ldots \times [d_i, d_j] \subset \mathbb{R}^i$ is a bounded coordinate-aligned box giving the parameter domain, $f : D \to \mathbb{R}^m$ give the spatial coordinates of the set, $g : D \to \mathbb{R}^m$ are the inequality constraints and $h : D \to \mathbb{R}^i$ are the equality constraints. An alternative form, which does not emphasise the distinction between inequality and equality constraints, is

$$S = \{f(z) \mid z \in D \mid g(z) \in C\} = f(D \cap g^{-1}(C))$$

where $D$ and $C$ are coordinate-aligned boxes.
The main advantage of constrained image sets is that they are sufficiently general to be able to represent reach and evolve sets arising in the evolution of a hybrid system directly, but sufficiently restrictive to be reasonably easy to work with.

The analysis above shows that in order to compute the evolution of a hybrid system, it suffices to implement operations for computing:

— the composition of functions;
— the range of a function;
— the flow of a differential equation;
— the solution of an algebraic equation.

In order to analyse the solution, including discretising reachable sets for global analysis, we also need to

— test feasibility of a constraint satisfaction problem.

Additionally, the following is very useful for user output:

— drawing a constrained image set in \( \mathbb{R}^2 \).

Notice that all these operations can be expressed in terms of boxes in \( \mathbb{R}^n \) and functions \( f : \mathbb{R}^n \to \mathbb{R}^m \). This suggests an approach to the analysis of nonlinear hybrid systems by the development of a function calculus, in which these operations are implemented rigorously and efficiently.

4. RIGOROUS NUMERICAL METHODS

The real numbers \( \mathbb{R} \) and the set of continuous functions \( \mathbb{R}^n \to \mathbb{R}^m \) have continuum cardinality, so there is no possible way of describing all elements exactly using a finite amount of data. The traditional approaches to handling uncountable sets computationally are either to restrict to a countable subset and use symbolic manipulation, or to work with approximations in a finite subset. For the purpose of verification of hybrid systems, neither approach is feasible; in the former, the computations take far too long, and in the latter, we lose mathematical rigour. Instead, we use rigorous numerical methods to perform the computations.

The main idea of rigorous numerics is to represent an element \( x \) of an uncountable set \( X \) by a set \( \tilde{x} \) containing \( x \). The set \( \tilde{x} \) is taken from a countable set \( \tilde{X} \) of subsets of \( X \), and has a concrete description given by a finite amount of data. An implementation of an operation \( \text{op} : X_1 \times \cdots \times X_n \to Y \) is a function \( \tilde{\text{op}} : \tilde{X}_1 \times \cdots \times \tilde{X}_n \to \tilde{Y} \) such that the following inclusion property holds:

\[
\tilde{x}_i \ni x_i \text{ for } i = 1, \ldots, n
\Rightarrow \tilde{\text{op}}(\tilde{x}_1, \ldots, \tilde{x}_n) \ni \text{op}(x_1, \ldots, x_n).
\]

The most well-known example of a rigorous numerical calculus is interval arithmetic [Moore, 1966]. A real number \( x \in \mathbb{R} \) is represented by an interval \( [x, \tilde{x}] \ni x \). Typically, the endpoints \( x, \tilde{x} \) are taken to lie in a set \( \mathcal{F} \) of floating-point numbers of a given precision. Arithmetical operations preserving the inclusion property can be implemented using rounded arithmetic, which are built-in for modern microprocessors. An interval extension of a function \( f : \mathbb{R} \to \mathbb{R} \) is a function \([f]\) on intervals satisfying the inclusion property

\[
x \in [x, \tilde{x}] \Rightarrow f(x) \in [f(x), [x, \tilde{x}]].
\]

A powerful rigorous calculus for continuous functions \( \mathbb{R}^n \to \mathbb{R} \) is based around the Taylor models of Makino and Berz [2003]. A Taylor model for a function \( f : \mathbb{R}^n \to \mathbb{R} \) is a pair \( \tilde{f} = (T, I) \) where \( T \) is a polynomial in \( n \) variables with coefficients in \( \mathbb{F} \) and \( I \) an interval satisfying

\[
\forall z \in [-1, +1]^n, f(z) - T(z) \in I.
\]

Taylor models for functions on a domain \( D \) other than \([-1, +1]^n \) can be constructed by pre-composing \( f \) with \( s^{-1} \), where \( s \) is a scaling function \([-1, +1]^n \to D \).

The usual operations on functions, including arithmetic, evaluation, composition, and antidifferentiation, can be extended to Taylor models, and satisfy the inclusion property. The interval \( I \) is used to capture the round-off errors introduced by floating-point arithmetic. The efficiency of Taylor models relies on sweeping terms of \( T \) with small coefficients into the interval \( I \).

Taylor models can be computed using a Taylor series with remainder term; in one-dimension we have

\[
T(z) = \sum_{k=0}^{n-1} \frac{1}{k!} f^{(k)}(0) z^k; \quad I = \frac{1}{n!} f^{(n)}([-1, +1])
\]

The derivative of \( f \) can be computed using automatic differentiation [Griewank, 2000], which is useful in many areas of rigorous numerics.

Algebraic equations \( f(x) = 0 \) for \( f : \mathbb{R}^n \to \mathbb{R}^n \) can be solved using the interval Newton operator [Moore, 1966], defined as

\[
N(f, \tilde{\gamma}, \tilde{\gamma}) = \tilde{\gamma} - [Df(\tilde{\gamma})]^{-1} f(\tilde{\gamma}),
\]

where \( \tilde{\gamma} \) is a box and \( \tilde{\gamma} \in \gamma \). If \( N(f, \tilde{\gamma}, \tilde{\gamma}) \subset \gamma \), then the equation \( f(y) = 0 \) has a unique solution in \( \tilde{\gamma} \), whereas if
There are many methods in the literature for solving differential equations [Lohner, 1987; Berz and Makino, 1998; Nedialkov et al., 1999; Zglliczynski, 2002]. However, most use some variation on the following procedure to compute the flow $\phi$ of $\dot{x} = f(x)$ starting in an initial set $X_0$ at time 0 for a time step $h$. First a bound $B$ is computed such that $\phi(X_0, [0, h]) \subset B$. A sufficient condition for $B$ to be such a bound is that $X_0 + [0, h]f(B) \subset B$. Then the Taylor coefficients of $\phi$ are computed using automatic differentiation, both at the point $(x_0, 0)$ and over the set $(B, [0, h])$. Finally, the results are combined to give a set $X_1$ containing $\phi(X_0, h)$. As an additional step, some packages have a reconditioning step to control the errors.

Constraint propagation [Kearfott, 1996, Jaulin et al., 2001, Hansen, 2003] is a collection of methods for determining solutions of $\{x \in D \mid f(x) \in C\}$, where $C$ and $D$ are coordinate-aligned boxes and $D$ is bounded. The result is given as a collection of boxes $X = \bigcup_{i=1}^{n} X_i$ such that $D \cap f^{-1}(C) \subset X$. The main technique is to use contractors to reduce the size of a box $X$, containing points of $g^{-1}(C)$, starting with $X = D$. Given a symbolic representation of $g$ as a directed acyclic graph, the revising hull consistency algorithm [Benhamou et al., 1999] uses interval arithmetic to propagate constraints; other contractors use monotonicity properties of $g$. A particularly powerful approach [Hansen, 2003] is to introduce Lagrange multipliers to the constraints and solve a related optimization problem. If no reduction in the solution set $X$ can be made, then it is split into two pieces, each of which are considered separately. In order to validate a candidate solution, methods based on applying the interval Newton test to the Karush-Kuhn-Tucker conditions can be used.

5. SET AND FUNCTION CALCULI IN ARIADNE

The computational kernel of Ariadne is written in C++, and provides complete support for the operations of rigorous numerics described in Section 4.

The main functionality for functions and sets is based around standard abstract interfaces which can be implemented in various ways. For technical reasons due to the handling of polymorphic data types in C++, for each interface class, a handle class is provided for users. Interfaces for various evaluators are also defined, with each evaluators being required to implement a closely-related set of operations.

In order to make the package as self-contained as possible, and because existing software libraries did not meet requirements on functionality, it was decided to implement all necessary operations within the package itself, including the core operations of interval arithmetic, linear algebra, automatic differentiation, and polynomial arithmetic.

The main abstract function types are ExactFunction for exact scalar functions described symbolically, and ValidatedFunctionModel for interval functions on a box domain.

The main class of validated function model implemented are scaled polynomial function models, similar to Taylor models; affine function models are also provided. Since the only existing package implementing Taylor models, COSY Infinity [Makino and Berz, 2006] is not open-source, an implementation within Ariadne itself is provided. The scaled polynomial models of Ariadne use a floating-point error bound $\epsilon$ rather than an interval error, and are defined defined by a tuple $(s, p, c)$ where

(i) $s : [-1, +1] \to \prod_{i=1}^{n} [d_i, \bar{d}_i]$ is a scaling function,
(ii) $p : [-1, +1] \to D$ is a polynomial $z \mapsto \sum c_i z^i$ with coefficients $c_i \in \mathbb{R}$, and
(iii) $\epsilon \in \mathbb{R}^+$ is an error bound.

A polynomial model $(s, p, c)$ represents $f : \mathbb{R}^n \to \mathbb{R}$ if $\sup_{x \in D} (f(x) - p \circ s^{-1}(x)) \leq \epsilon$ where $D = \prod_{i=1}^{n} [d_i, \bar{d}_i]$ is the domain of the model.

The coefficients $c_i$ of the polynomial $p$ are stored in a memory-efficient sparse array format, and standard operations of scalar and arithmetic, range, differentiation, and antidifferentiation, are efficiently implemented. For example, addition is performed using

$$p_1 + p_2 = \sum c_{1,\alpha} + c_{2,\alpha} x^\alpha$$

assuming the domains are equal, where $\star_1$, $\star_2$, and $\star_n$ denote operations with rounding upwards, downwards and to nearest, and $\sum$ is sum with upwards rounding. Different methods are provided to evaluate and compose polynomial models, including direct evaluation and evaluation based on Horner’s rule. Various sweepers allow the accuracy versus efficiency tradeoff of the polynomial to be controlled.

In order to compute the evolution of a hybrid system, we need to keep track of the elapsed time as well as the current state. A TimedEnclosure interface is provided, allowing different concrete set types to be used to keep track of the reach and evolved states. The most powerful timed enclosure type implemented is based on constrained image sets of the form

$$E = \{ x(z), \tau(z) \mid z \in D \mid \rho(z) \in C \}$$

where $\xi(z)$ is the state $x$ and $\tau(z)$ the elapsed time $t$ for the parameter $z$, and $\rho(z) \in C$ are the constraints.

An integrator is an evaluator for computing the flow of a differential equation. Ariadne currently provides three integrators, an Affine integrator for computing the flow of an affine system $\dot{x} = Ax + b$, a PicardIntegrator, which uses Picard’s iteration

$$\phi_{n+1}(x_0, t) = x_0 + \int_0^t \phi_n(x_0, \tau)) d\tau,$$

and a TaylorIntegrator, which computes the flow using a Taylor series expansion. In practise, the Taylor integrator outperforms the Picard integrator, since it generates sharper error bounds after fewer iterations.

A solver is an evaluator for computing the solution of a parameterised algebraic equation $f(x, h(x)) = 0$. Ariadne implements an IntervalNewtonSolver based on the interval Newton operator, and a KrawczykSolver based on the related Krawczyk operator, which is more reliable but slower. The crossing time $\gamma(x)$ of the flow $\phi(x, t)$ with the
guard $g(y) = 0$ is computed by solving $g \circ \phi(x, t) = 0$ with the interval Newton iteration

$$\gamma_{n+1}(x) = \gamma_n(x) - \frac{g(\phi(x, \gamma_n(x)))}{(\nabla g \cdot f)(\phi(x, \gamma_n(x)))}.$$  

The convergence rate depends on the size of $(\nabla g \cdot f)(\phi(x, \gamma(x)))$.

A propagator is an evaluator for testing feasibility of the constraint satisfaction problem $z \in D \land f(z) \in C$. Ariadne implements a propagator based on applying contractors based on hull consistency and monotonicity properties. Interior-point techniques from nonlinear programming are used to find Lagrange multipliers to construct linear combinations of constraints which can be more efficiently contracted. Splitting of the domain is performed based on an estimation of the Jacobian to try to decrease the nonlinearity of the function over the subdomains.

A drawer is an evaluator for drawing a set on a canvas. The problem of efficiently and accurately drawing a constrained image set $S = f(D \cap g^{-1}(C)) \subset \mathbb{R}^2$ is nontrivial since the domain $D$ may have a very high dimension. Ariadne implements a simple BoxDrawer, in which $D$ is subdivided into boxes $D_i$ and $|f|(D_i)$ is drawn whenever $|g|(D_i) \cap C \neq \emptyset$. The more sophisticated AffineDrawer draws sets $|f|(D_i \cap (g^{-1}(C)))$, where $|f|$ and $|g|$ are affine approximations to $f$ and $g$, and usually achieves good accuracy with few subdivisions.

6. HYBRID SYSTEM EVOLUTION IN ARIADNE

The main algorithm for the computation of the evolution of a hybrid system in the current version of Ariadne follows the procedure outlined in Section 3. Integrators are used to compute the flow, and solvers to compute the crossing times. Propagators are used to compute discretisations of sets, to check safety and liveness conditions, and to test whether an enclosure is empty and can be removed from consideration of the future evolution.

There are three major practical difficulties when using the set and function calculus for computing the evolution of a hybrid system. The first difficulty is how to handle degenerate crossings, and is currently tackled by computing under- or over-approximations. The second difficulty is how to avoid splitting the enclosure sets as much as possible, since once split, sets cannot easily be recombined. The third difficulty is how to manipulate the enclosure sets to maintain accurate over-approximations without too heavy a storage or computational burden.

6.1 Event scheduling

When two events are possibly active, then the evolution splits into two pieces, one taking each event. Assuming the events are urgent and crossing times $\gamma_{1,2}$ can be computed,

$S_1 = \{t_1(\phi(x, \gamma_1(x))) \mid x \in D \land \gamma_1(x) \leq \gamma_2(x)\}$

$S_2 = \{t_2(\phi(x, \gamma_2(x))) \mid x \in D \land \gamma_2(x) \leq \gamma_1(x)\}$.

Otherwise, we can write

$S_1 = \{t_1(\phi(x, t_1)) \mid x \in D, t_1 \in [0, t_h] \land g_1(\phi(x, t_1)) = 0 \land g_2(\phi(x, t_1)) \leq 0\}$.

With a single active event, care must be taken to avoid splitting the evolved set unnecessarily. Suppose the flow $\phi$ over an initial domain $D$ can be computed over a time step $t_h$, and that $\phi(D, t_h)$ straddles a guard set $g(x) = 0$. Then the crossing time $\gamma(x)$ has $t_h \in \gamma(D)$.

6.2 Reconditioning

The accuracy of computation on polynomial models $p(z) = e$ with $e$ is highly sensitive to the value of $e$; if this becomes too large, then subsequent computations tend to lose accuracy very quickly. In Ariadne, we implement the reconditioning procedure of [Kuhn, 1998]

$$\{p(z) \pm e \mid z \in [-1, +1]^{n+1}\} = \{p(z_1) + c_2 \pm 0 \mid (z_1, z_2) \in [-1, +1]^{n+1}\}$$

to eliminate the error terms $e$ at the expense of introducing a new variable. Conversely, we can reduce the number of variables (at the expense of increasing the error terms) since

$$\sum_{\alpha} a_\alpha z^\alpha \pm e = z \in [-1, +1]^{n+1}$$

$$\subset \{\sum_{\alpha=0} a_\alpha z^\alpha \pm \sum_{\alpha > 0} |a_\alpha| + u_\epsilon \mid z \in [-1, +1]^{n+1}\}.$$

If enclosure sets become ill-conditioned or have too many parameters, then reconditioning can help control the size of the polynomial and the errors. We have implemented the reduction process of Lohner [1987] for affine enclosures without constraints. If $A = QDR$ with $Q$ orthogonal, $D$ diagonal and $\|R\|_{\infty} \leq 1$, then

$$\{Dz + b \mid z \in [-1, +1]^{m}\} \subset \{QDz + b \mid z \in [-1, +1]^{m}\}.$$

If enclosure sets become too big, then splitting is needed to avoid blow-up of errors. The simplest way of splitting is to subdivide the parameter domain along one of its coordinate axes:

$S_i = f(D_i \cap g^{-1}(C))$ for $i = 1, 2$ with $D = D_1 \cup D_2$. 
In Ariadne, the splitting coordinate for the purpose of evolution is chosen to reduce the sizes of the bounding boxes of the sets. Splitting also has the beneficial effect of decreasing the coefficients of the scaled polynomial models, and may result in redundant constraints which can be removed from the representation.

6.3 Future improvements

The current implementation of the evolution routines in Ariadne provides a complete functionality for computing the evolution of hybrid systems, but there are many areas with potential improvements to the accuracy or efficiency of the calculations.

To simplify constraint propagation and event scheduling, and reduce ill-conditioning of the functions defining the enclosures, it may be more efficient to explicitly consider intermediate variables in the description of the evolved sets. For example, by writing enclosure sets as

\[
\{ (y, t_1, y_1, x_1) \in D \mid y_1 = \phi_0(x_0, t_1) \land g_1(y_1) = 0 \\
\land x_1 = \rho_1(y_1) \land y = \phi_1(x_1, t_ - t_1) \},
\]

where \( x_1 \) and \( y_1 \) represent the states immediately before and immediately after the occurrence of some discrete event \( e_1 \), rather than

\[
\{ \phi_1(\rho_1(\phi_0(x_0, t_1)), t_ - t_1) \mid (x_0, t_1) \in X_0 \times [0, t_f] \\
\land g_1(\phi_0(x_0, t_1)) \}.
\]

Since the error of computing the flow \( \phi(x, t) \) over \( D \times [0, t_b] \) depends mostly on the bounding box for the range, where fairly large step sizes are used, it may be more efficient to precompute the flow over a larger domain, rather than compute the flow for each step. Similar techniques have been used in [Dang et al., 2009]. Here, the main problem to solve is how to find a good balance between using a small domain which allows higher accuracy for a lower order, and a large domain which requires fewer computations of the flow.

Furthermore, the accuracy of the computation of the flow may be improved by using a parallelepied rather than a box to bound the flow step, as this will yield a tighter bound. To support this, we are considering allowing a general invertible affine map for the scaling of a polynomial function model, \( f(x) = p(A^{-1}(x - c)) \pm e \).

Finally, since the reconditioning procedure for affine enclosures cannot be applied directly to nonlinear enclosures or enclosures with constraints, work is in progress to find reconditioning techniques which do work in these situations.

7. EXAMPLE—WATER TANK SYSTEM

In this example, the water height \( h \) in a tank with continuous outflow and a valve-restricted inflow (see Fig. 4) needs to be controlled between \( h_{\text{min}} \) and \( h_{\text{max}} \). By Torricelli’s law, the rate-of-change \( h \) is given by

\[
\dot{h} = -a\sqrt{h} + b\alpha
\]

where \( \alpha \in [0, 1] \) is the aperture of the inlet valve, \( a \) and \( b \) are two constants whose value depends on some physical quantities of the system. The valve can be opened or closed at a speed of \( 1/T \) and it is controlled so that it starts to

open as soon as \( h \leq h_{\text{open}} \) and starts to close as soon as \( h \geq h_{\text{close}} \). Fig. 5 shows the hybrid automaton modeling the water tank.

At the beginning the automaton is in location \( \text{opening} \), the value of \( \alpha \) increases with speed \( 1/T \) and the water level follows equation (1). As soon as \( \alpha = 1 \) (the valve is fully open) the urgent transition to \( \text{open} \) is taken. The valve is kept open until \( h = h_{\text{close}} \), when the automaton switches to location \( \text{closing} \) and the valve starts closing with speed \( -1/T \). The urgent transition to \( \text{closed} \) is activated when \( \alpha = 0 \), and the water level decreases following the dynamics \( \dot{h} = -a\sqrt{h} \) until \( h = h_{\text{open}} \) and the transition to \( \text{opening} \) is taken.

A computation of the finite-time reachable set of one evolution loop starting from \( (\text{opening}, 0, 0) \) using Ariadne is shown in Fig. 6.

8. COMPARISON WITH OTHER TOOLS

Most other tools for performing reachability analysis of hybrid systems use the same basic framework as Ariadne.
i.e. enclosure sets are propagated under the evolution of the flow. However, not all tools have the ability to handle nonlinear systems or compute infinite-time chain-reachable sets. Ariadne has the ability both to accurately compute long evolution traces or perform discretisations after relatively short time steps, and distinguishes between lower-semantics for proving existence of trajectories, and upper-semantics for proving non-existence, which have different computability properties.

Phaver [Frehse, 2008] uses zonotopes as enclosures, can only handle piecewise-affine-derivative systems and affine guards. Termination of an infinite-time reachability computation is performed by testing if no new enclosures are generated. d/dt [Asarin et al., 2002] is a tool for affine systems with affine guards, and uses polytopes as its enclosures. HyperTech [Henzinger et al., 2000] uses boxes as enclosures, and can compute the solution of differential inclusions with small noise terms. SpaceEx [Frehse et al., 2011] combines polyhedra and support function representations of the state space, and guarantees local error bounds on the computation of systems with piecewise affine dynamics and guards. Checkmate [Clarke et al., 2003] can handle nonlinear dynamics, but only affine guard sets. HSOLVER [Ratschan and She, 2007] can handle nonlinear hybrid systems, but can only prove safety properties.

Methods of [Kurzhanski and Varaiya, 2002a,b] use ellipsoids as enclosures, but cannot perform infinite-time reachability analysis. The level-set toolbox of [Tomlin et al., 2003] can be used for reachability analysis. It uses a global representation in terms of the frontier of the reached region, which can suffer from singularities occurring on the boundary during the computation.


9. CONCLUDING REMARKS

Ariadne is a software tool for reachability analysis of nonlinear hybrid systems based on a rigorous numerical calculus for manipulating functions and sets. The combined functionality allows rigorous and accurate computation of the evolution of nonlinear hybrid systems.

The functional calculus includes support for interval arithmetic, linear algebra, automatic differentiation, function models with evaluation and composition, solution of algebraic and differential equations, constraint propagation and non-linear programming. Although there are software packages supporting each of these functionalities individually, Ariadne is the first open-source package providing integrated support for all this functionality. Additionally, due to the modular structure, additional function and set types can be introduced, and different implementations of the algorithms provided. Indeed, the software can be seen as a general-purpose tool for verified numerics.

Work in the immediate future will focus on further improvements to the efficiency and accuracy of the tool. Partially implemented future extensions include evolution of nondeterministic hybrid systems described by differential inclusions [Zivanović and Collins, 2010] and verification of linear temporal logic formulae [Collins and Zapreev, 2009]. Theoretical work is in progress on the evolution of stiff continuous dynamics, the analysis of stochastic systems and dynamical games, the computation of optimal controllers, system reduction, including time-scale decomposition and assume-guarantee reasoning.

REFERENCES


W. Kühn. Rigorously computed orbits of dynamical systems without
A. B. Kurzhanski and P. Varaiya. On ellipsoidal techniques for
A. B. Kurzhanski and P. Varaiya. On ellipsoidal techniques for
reachability analysis. II. Internal approximations box-valued con-
R. J. Lohner. Enclosing the solutions of ordinary initial and bound-
ary value problems. In C. Ullrich E.W. Kaucher, U.W. Kulisch,
editor, *Computer Arithmetic: Scientific Computation and Pro-
K. Makino and M. Berz. Taylor models and other validated
K. Makino and M. Berz. COSY INFINITY Version 9. *Nuclear
N. S. Nedialkov, K. R. Jackson, and G. F. Corliss. Validated solutions
of initial value problems for ordinary differential equations. *J.
S. Ratschan and Z. She. Safety verification of hybrid systems
by constraint propagation based abstraction refinement. *ACM
O. Stauning. *Automatic Validation of Numerical Solutions*. PhD
C. J. Tomlin, I. Mitchell, A. M. Bayen, and M. Oishi. Computational
techniques for the verification of hybrid systems. *Proc. IEEE*, 91
S. Živanović and P. Collins. Numerical solutions to noisy systems.