Symbolic Verification of Hybrid Systems

MARTIN V. MOHRENSCHILDT
Department of Computing and Software,
Faculty of Engineering,
McMaster University,
Hamilton, Ontario, Canada L8S 4K1,
e-mail: mohrens@mcmaster.ca

Abstract In this paper we present a new symbolic approach to hybrid systems. Hybrid systems are systems containing both, continuous and discrete changing quantities. We model hybrid systems using hybrid automata: Hybrid automata extend the classical notion of finite state machines by combining differential equations to model the dynamic behavior of systems with a finite control. In contrast to other approaches we consider a hybrid automata as a generalization of differential equations and develop the notion of a symbolic "closed form" solution of a hybrid automata. A closed form solution is an expression which gives the value of the quantities in question as a function of design parameters and time. These solutions allow us to perform the verification of design properties. We were able to detect design constrains on control systems that other methods fail to detect. This paper gives the basic definitions, algorithms, and an example to demonstrate the advantage of the proposed approach.

Key-Words: Hybrid-Systems, Symbolic Computation, Differential Equations, Design Verification, Control Systems.

1 Introduction

Many engineered systems contain both traditional analog components and modern digital (finite state) components. Such mixed continuous-discrete systems are often called **Hybrid Systems**. It is of high interest to develop precise models for hybrid systems in order to develop and verify control software (hardware) for systems such as nuclear reactors, patrol chemical plants, or car engines.

In general, we observe two approaches to hybrid systems: *discretisation* and *continuation*; the systems are transformed either to a pure discrete or to a pure continuous system. There are few exceptions to this classification (e.g. [5]).

The discrete approach describes the system in terms of a finite set of *states* (assignments of values to the state variables). The hybrid system is transformed into a discrete system by partitioning the continuous state space into *modes*. The mode changes of a hybrid system are commonly triggered by some external events or by conditions on the state variable. The theory of *Hybrid Automata* [1][10],[5] follows this discrete methodology. Most analytical approaches, approaches based on logic, do not actually study the differential equations which describe the continuous state changes but restrict themselves to a specific, easily solved class of equations, most of the time equations of the form $y' = c$. [1], [2] In discretisation approaches, much of the dynamic behavior of the system is lost. For example, valves do not close instantaneously even though many discrete models assume so. Furthermore, by using discrete modeling techniques, some design constraints are lost in the process of modeling. For the well known water level control, discrete models failed to detect the valve closing speed constraints which were easily detected using my proposed approach (e.g. [1] [10]).

On the other hand, the continuation approach often found in control theory, transforms the discrete state changes into continuous ones by using approximation techniques, or integral transformations. The result is a set of nonlinear differential equations modeling the hybrid system. These differential equations are then solved using sophisticated mathematical methods such as differential inclusions [4]. As many of the real control functions are implemented on digital machines, they contain discrete state changes; the continuous models do not model the real system.

In this paper we present a symbolic algebraic approach to hybrid systems which can be seen as a mixture of the two above described methods.

We developed a method to define and compute closed form expressions called *solutions* of a hybrid system. A solution of a hybrid system is defined as a closed form algebraic expression which, when evaluated, (interpreted as a function,) determines the value of each of the state variables as the system evolves in time. This is accomplished by symbolically solving all differential equations and then by deriving a recurrence relation which, when

solved, gives the initial conditions for each mode the system is in as time passes. To be able to represent these piecewise and periodic functions as algebraic expressions we have to extend basic algebraic structures to contain elements representing "discrete switches" [8], [9], [6], [7] and periodic functions. The solutions of hybrid systems will often contain design parameters, parameters that can be chosen in order to optimize the behavior of the system by minimizing or maximizing some *cost-function* as in control theory. The solutions, being symbolic expressions, allow us to *verify* design constraints and to optimize design parameters. Further, in contrast to numerical solutions, the symbolic solutions are exact, they do not contain any rounding errors; rounding errors could trigger an unwanted discrete state change. The proposed approach differs form the logic based approach which normally use fix-point computation in order to derive formulas which hold in each state the system is in. Our approach allows to verify systems which do not convert, fix-point computation methods cannot be applied to such systems.
This paper gives an introduction, the basic definitions, algorithms and demonstrate the power of our methods with an example, a water level control system. It is not in the scope of this paper to give all technical details such as decidability and stability of hybrid systems.

2 Hybrid Differential Equations

Normally, the definition of a solution of a differential equation assumes that all coefficients of the differential equation are continuous. But as we model real systems, especially engineering systems containing a discrete control, the differential equation modeling the behavior of the system can change in a discrete manner. These discrete changes can in general be triggered not only by conditions on time [6],[7], [6] but also by conditions on the solution itself. The solutions of classical ordinary differential equations are *local*, meaning that the behavior of the system is defined by the shape of the vector field at the point of interest. Real systems can have a "finite memory", the vector field (given by the coefficients of the equations) can change based on a finite number of events of the past (e.g. hysteresis). We could even stop the system in any state (point of time) and then continue with some variables (initial states) changes in a discrete manner. Even the order of some differential equations could change (as in the example given later in the paper). Hybrid systems can perform actions; expressed in terms of differential equations: New initial conditions can be given triggered by some conditions on the state of the system.
The above shows that, in order to model hybrid systems we have to extend the notion of differential equations to **hybrid differential equations**. For example $y'' + signum(y)y' + y = 0$ describes a system (an oscillator) where the coefficient of y' depends on the position of the system, or $y' = H(y(x))$ where $H(x)$ is a hysteresis function.
Defining and computing the solution of hybrid differential equation is much more difficult than that of a continuous differential equations. The variety of possible behaviors is much larger. The classical uniqueness theorems for initial value problems, which requiter Lipschitz continuity of the equations. Hybrid systems are not Lipschitz continuous. We approach hybrid differential equations by transforming them into semantically equivalent (having the same solution) hybrid automata. A hybrid automata splits the system into modes, where in each mode the differential equation is continuous and all discrete state changes are performed as the mode is changed.

3 Hybrid Automata

We study systems containing several (continuous) variables changing, continuously or discrete, in time. A state of a system is given as each of the variables has a specific value. Clearly, if we consider time as an integral component of our systems, the system will never be two times in the same state. The values of the variables of the systems can either change continuously in time, or instantly (discrete), to new values. The continuous changes of the variables are described by differential equations, the discontinuous changes are described by assignments. We model such systems using hybrid automata, which are automata consisting of a finite set of modes and a finite set of transitions. Each mode contains a system of differential equations. The transitions between the modes are guarded with conditions and can contain assignments (actions).
Formal, a hybrid automata is given by:

- A finite set of **variables** $V = \{y_i\}$: All variables depend on time. We write $y_i(t)$ to denote the value of a variable y_i a time point t.
- A finite set of **modes** $S = \{s_i\}$ $i = 1 \cdots n$. The mode s_i contains the system of m autonomous differential equations $y_l' = f_{i,j}(y_l, t)$ $j = 1 \cdots m$. Without loss of generality we assume that these are coupled first-order systems. [1] Derivatives are always derivatives by the time t.
- A set of **transitions** T. The transition $t_{i,j}$ leads from mode s_i to mode s_j. A transition $t_{i,j}$ is a pair $(c_{i,j}, a_{i,j})$ of:
  - **Condition** $c_{i,j}$. $c_{i,j}$ a boolean expression which can contain predicates on the solution of the

[1] Although we sometimes use higher order equations for convenience, the formal definition is based on first-order systems.

differential y_i equation, or, in general, any other discrete or analog variables.

- **Actions** $a_{i,j}$**.** An action $a_{i,j}$ is an assignment to variables of V. Actions are used to "trigger-events" or to give new initial conditions to the differential equations of the next mode.

- The unique initial transition $t_{0,i} = (\mathbf{true}, a_{0,i})$ with an identical true condition and the action $a_{0,i}$ which initializes **all** variables of V.

Often a graphical representation of hybrid systems is given.

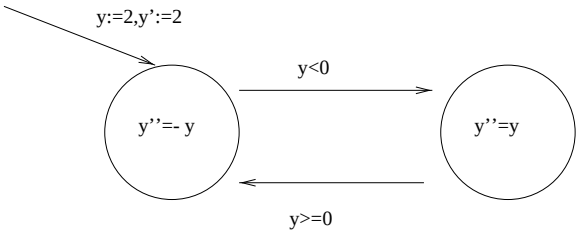

Figure 1: Sample system

The here defined hybrid automata are a variation of those found in literature [1] [2], ,[5], [12] the main differences can be observed in the definition of the semantics.

3.1 Semantics of a Hybrid Automata

We introduce some further conventions and notation:

- If a mode does not contain a differential equation in the variable y_i then we assume that this variable is constant, meaning that the mode implicitly contains the equation $y_i' = 0$.

Definition 3.1 (Well-defined) *A hybrid automata is well-defined if*

- **Deterministic** *In any mode s_i only one of the conditions $c_{i,j}$ can be true at any instance of time.*

- **No zero time** *If a condition $c_{i,j}$ it true at time point t_k and action $a_{i,j}$ is taken, then there is no transition condition $c_{j,l}$ true at time point t_k. This means that we cannot "pass through" a mode. (Else we could define an automaton which would never stay in any mode, and never proceed in time).*

- **Unique transition** *There exists at most one unique transition $t_{i,j}$ leading from mode s_i to mode s_j.*

Now we are ready to give the semantics of a hybrid automata. In order to define unique solutions of a hybrid automata we give a deterministic semantics. Our semantics can be called an earliest time semantics that is, the automaton changes mode at the earliest possible point of time. Other semantics found in literature e.g. [1] [2] allow mode changes at any point of time a transition condition is true, leading to a nondeterministic behavior. We see the possibility to generalize our approach in future work using intervals as solutions, meaning that each possible solution will enclosed by a with functions bounded interval.
With t_i we denote the time point where the automata switches mode the i-th time.
Semantics of a well-defined hybrid automata

- The automata starts with the initial transition, where all variables are assigned a specific value. The time starts at zero, $t_0 = 0$.

- When entering the mode s_i at time point t_k we start the "dynamic system" defined by the differential equations in this mode s_i. The initial conditions are given by the value of the variables at time point t_k. As soon as a condition $c_{i,j}$ becomes true,

$$t_{k+1} := \min_d c_{i,j}(t_k + d) = \mathbf{true}$$

the automata changes mode to s_j performing action $a_{i,j}$. Since the automata is "no-zero-time" the system will stay for some time in mode s_i. [2]

From now on we restrict our examination to hybrid automata where no variable can be changed by external "events". All changes of variables are performed by the hybrid automata itself. Such automata are called **closed hybrid automata**. This implies that we model the entire system of interest.

3.2 Hybrid Differential Equations and Hybrid Automata, Traces, Uniqueness

As stated, our aim is to develop a theory of solutions of hybrid automata which is similar to the solutions defined for differential equations.
Following the idea of a solution to an initial value problem for differential equation we say that **a hybrid automata with initial transition $t_{0,i}$ defines a vector valued function $\overline{f}$ of time t**:

$$\overline{f}(t) = (f_{y_1}(t), f_{y_2}(t), \cdots f_{y_n}(t)).$$

[2] We assume that this mode switch happens instantly. A hybrid automata where these transitions take time can always be obtained by adding extra modes modeling the time a transition would take to perform

Definition 3.2 (Transition-Trace, Trace) *A* **transition-trace** *T of a hybrid automata is the (finite or infinite) sequence of transitions $t_{i_k,i_{k+1}}$ which the hybrid automata performs as time passes. The* **trace** *of a system corresponding to a transition-trace is the sequence of modes s_{i_k} in which the hybrid automata is in as time passes.*

We define the **run** corresponding to a trace of a hybrid automata which gives the values of all variables as time passes.

Definition 3.3 (Run) *The run R corresponding to the trace $T = s_{i_0}, s_{i_1}, s_{i_2}, \cdots$ of a hybrid automata is a sequence of pairs (t_k, f_k) where t_k is the time point at which the automata switches from mode s_{i_k} to mode $s_{i_{k+1}}$ and f_k is a vector valued function, giving values for all variables, and satisfying the differential equations in mode $s_{i_{k+1}}$ with initial values $y(t_{k+1}) = f_k(t_k)$. $f_0(t_0)$ is the initial condition of mode s_1 given by the initial transition $t_{0,1}$.*

Now we are able to state a theorem which holds only for closed, well-defined hybrid automata.

Theorem 3.4 (Unique Trace, Run) *The trace and run of a closed, well defined closed hybrid automata is unique.*

Proof We point to the uniqueness results of initial value problems of differential equations. The system enters mode s_i form some mode s_k using a transition $t_{k,i}$ and performing the actions ak, i. The values of all variables give a unique set of initial conditions for the differential equations in mode s_i. These initial conditions define a unique behavior of the differential equations in this mode. The hybrid automata is well defined; the "no-zero-time" condition guarantees that there is no transition condition is true as we enter the mode. Hence, there will be a unique transition condition of a transition $t_{i,j}$ leading to mode s_j that is the first to be true. This leads again to unique initial conditions of the differential equations in the next mode. Hence the sequence of modes s_{i_j}, and the times t_j where the automaton switches mode are unique to a specific initial transition. ∎

Note that different initial transitions can change the behavior of a hybrid automata significantly. This is also true for unstable differential equations.

Behaviors of solutions of hybrid automata

- **Converging Solutions** The trace is finite. This means that the automata enters a mode s_i at time point t_k where $c_{i_j}(t) =$ **false** for all $t > t_k$.

$$s_{i_0}, s_{i_1}, \cdots, s_{i_n}$$

- **Periodic Solutions** The trace is infinite, but periodic,

$$s_{i_0}, s_{i_1}, \cdots, s_{i_k}, s_{i_{k+1}}, s_{i_{k+2}}, \cdots s_{i_{k+p}}, s_{i_{k+1}}, s_{i_{k+2}}, \cdots.$$

- **Chaotic Solutions** The trace is infinite and not periodic.

Note that a periodic trace does not imply that the values of the variables are periodic, only that, after a finite sequence of modes, we cycle through a sequence of modes. We will represent the first two types of behaviors as closed form expressions.

4 Transformations of Hybrid Automata

We present a transformation of hybrid automata which we will use later to simplify our algorithms.

Definition 4.1 (Equivalence) *Two hybrid automata H and $\tilde{H}$ are equivalent if the functions f defined by H and $\tilde{f}$ defined by $\tilde{H}$ are the same for each possible action in the initial transition $t_{0,i}$.*

Theorem 4.2 (Single condition, action entrance) *Given a hybrid automata H, we can construct a hybrid automata $\tilde{H}$ where all transitions leading to any mode s_i $t_{j,i}$ have the same condition and same action:*

$$\forall j, k \; t_{j,i}, t_{k,i} \in T \; c_{j,i} = c_{k,i} \wedge a_{j,i} = a_{k,i}$$

and H and $\tilde{H}$ are equivalent.

Proof Let s_i be a mode with two transitions $t_{j,i}$ $t_{k,i}$ with different conditions or different actions leading to it. We add a new mode $s_{\tilde{i}}$. The transition $t_{j,i}$ is changed to the transition $t_{j,\tilde{i}}$ ($t_{j,i}$ is deleted) and for each transition $t_{i,k}$ we add a new transition $t_{\tilde{i},k}$ with identical condition $c_{\tilde{i},k} = c_{i,k}$ and identical action $a_{\tilde{i},k} = a_{i,k}$. Clearly mode s_k will have additional new transitions leading to it, but their conditions and actions are identical. We continue until all modes have only transitions with identical conditions and actions. The process will terminate since we will not add any new "problem transitions" and the number of transitions is finite. ∎

Transforming a hybrid system will reduce the number of possible initial conditions in each mode.

5 Closed Form Solutions, Verification

Our goal is to compute an expression which will represent the behavior of a hybrid automata as a function of

time. In contrast to other approaches for the verification of hybrid automata we define and compute closed form solutions of hybrid automata, meaning we find closed form expression which computes the value of all variables of the hybrid system depending on time. As design parameters are symbolic constants we are able to detect (compute) constrains on the design parameters in order to satisfy constrains put on the hybrid system. Clearly, due to indecidability results of hybrid automata we are not able to compute closed form expression for each hybrid automata [12].

Definition 5.1 (Closed Form Solution) *Given a hybrid automata H with n variables. Let $\overline{f}$ be a n-dimensional vector valued function defined by this automata (3.3). An expression s with $s(t) = \overline{f}(t)$ is called a close form solution of this hybrid automata.*

5.1 Expressions

In order to represent closed form solutions of hybrid automata we have to enlarge the classical algebraic are expressions used to represent these solutions.
We start with the classical algebraic expressions formed using the variables y_i, formal parameters p_i, time t rational numbers, and the function symbols $+, -, *,$. Further, as needed, we add transcendent extensions such as e, ln, sin, cos. For expressions $t_1, t_2, \cdots t_n$ and t we write $t \circ \sigma = t \circ \{y_{i_1} \to t_{i_1}, y_{i_2} \to t_{i_2}, \cdots\}$ to represent the simultaneous substitution of the variables y_{i_k} in the expression t. Next we extend the expressions by three expression constructors namely:

5.1.1 Piecewise Continuous Functions

In [6] a decidable theory of piecewise continuous functions is presented. A piecewise continuous function is a function that can be represented using piecewise expressions, expressions of the from:

$$\mathit{piecewise}(c_1, f_2, c_2, f_2, c_3, f_3, \cdots)$$

where the c_i are boolean conditions and the f_i expressions. The expression f_i is selected if the condition c_i is true. [3].

5.1.2 Periodic and Generalized Periodic Functions

A function f is called periodic if for some p, the period,

$$x \in 0..p \; \forall k \in N \; f(x) = f(x + k * p)$$

WE represent periodic function using the following notation $\mathbf{per}_{[f,p]}$

[3] If two conditions are true at the same time, then the first in the sequence of conditions is chosen.

Definition 5.2 (Integer Divider and Reminder) *For a $x \in \mathbf{R}$ we define $\mathit{irem} : \mathbf{R} \to \mathbf{R}$ and $\mathit{idiv} : \mathbf{R} \to \mathbf{Z}$*

$$x \,\mathit{irem}\, p = k \text{ where } k \in \mathbf{Z} \text{ such that } \quad p\, k \leq x < p\,(k+1)$$

$$x \,\mathit{idiv}\, p = y \text{ where } y \text{ such that } \mu k \in \mathbf{Z}\; y + k\; p = x$$

(μk means the smallest k such that $\cdots$)

We generalize further and even allow the period of a function to change.

Definition 5.3 (Generalized Periodic) *A function $f : \mathbf{R} \to \mathbf{R}$ is* **generalized periodic** *with period p there exists a function $g : \mathbf{R} \times \mathbf{Z} \to \mathbf{R}$ with*

$$f(x) = g(x \,\mathit{irem}\, p, x \,\mathit{idiv}\, p)$$

we write then $f = \mathbf{gen\_periodic}_{\,[g,p]}$

Generalized periodic expressions allow us to represent functions which can not be represented using classical algebraic expressions. Such a function is for example $x \,\mathit{idiv}\, p$, the "stairway to heaven" function.

5.2 Computing Closed-Form Solutions

Now we are equipped to present our method used to compute closed form solutions of hybrid automata. The computation of the closed form solution is performed in four steps: (1) Solve the differential equations, (2) compute the possible traces of the system, and decide if the trace is finite or periodic, (3) compute the initial conditions for each mode in the trace, using a recurrence relation, as a function of time and the number of times we cycled through the periodic portion of the trace. (4) Find a closed form expression representing the function defined by the hybrid automata. We describe each step of the algorithm. Step one and three are fully automated, step two still needs user input in some cases. Given a hybrid automata H we first transform H to a single condition-action-entrance automata $\tilde{H}$ using 4.2.

1. **Symbolic Solving Step**
   - Compute the symbolic solution $f_{i,y_i}(t, y_0, \cdots, y_n)$ of the (system of) differential equations in each mode s_i where the starting time is parameterized t_{s_i} and initial conditions, not fixed by the conditions or actions leading to this mode, are parameterized. This results in the substitution $\mathit{sol}_i = \{y_j \to f_{i,y_j}\}$
   - For each mode s_i, for each transition $t_{i,j}$ determine the time point(s) $\mathit{trans}_{i,j}$ by solving the (system of) equations $c_{i,j} \circ \mathit{sol}_i$. These time points will depend on the parameters introduced as the initial conditions for this mode.

For some systems we detect restrictions of the parameters in this step since it must hold that $trans_{i,j} > t_{s_i}$.

2. **Trace Determining Step** First we determine all possible loops of the hybrid automata (a loop is a repeating sequence of modes). Clearly, there exist only finite number of loops; there is only a finite number of modes, hence each infinite trace has to enter one mode at least twice. We represent all possible traces as sequences of modes:

$$s_{i_0}, s_{i_1}, \cdots, s_{i_k}, s_{i_{k+1}}, s_{i_{k+2}}, \cdots s_{i_{k+p}}, s_{i_{k+1}}, s_{i_{k+2}}, \cdots$$

In the next step we will have to decide which of the possible traces are actually taken. This problem is in general undecidable, but many practical hybrid automata (e.g. control systems) are constructed with a periodic trace in mid.

3. **Recurrence and Construction Step**

- Given a finite trace we can easily construct a piecewise defined function which is the solution of the hybrid automata. For the trace

$$s_{i_0}, s_{i_1}, s_{i_2}, \cdots, s_{i_n}$$

we construct the function

$$\begin{aligned} &piecewise(t < t_1, sol_1, \\ &\qquad t < t_2, sol_2 \circ \{y_i(t_1) \to sol_1(t_1)\}, \cdots, \\ &\quad t > t_n, sol_n \circ \{y_i(t_{n-1}) \to sol_{n-1}(t_{n-1})\}) \end{aligned}$$

where $\{y(t_1) \to sol_2(t_k)\}$ gives the initial conditions as we enter a new mode.

- For a periodic trace

$$s_{i_0}, s_{i_1}, \cdots, s_{i_k}, s_{i_{k+1}}, s_{i_{k+2}}, \cdots s_{i_{k+p}}, s_{i_{k+1}}, s_{i_{k+2}}, \cdots$$

we derive a coupled recurrence relations for the loop

$$y_i = y_i \circ sol_{i_{k+p}} \circ \cdots \circ sol_{i_{k+1}} \circ sol_{i_k}$$

for each of the unknown initial conditions in mode s_{i_k}. The solution of these recurrences equations will determine the new initial values depending on the old values y_i and possible other values as we re-enter mode s_{i_k} after one period. We write $f_{i_k,y_i}(n)$ to denote the value of y_i as we enter the mode s_{i_k} the nth time.

- Solve the recurrence equations to determine the initial conditions for mode $s_{i_{k+1}}$ depending on n and $f_{y_i}(0)$.
- Compute the remaining initial conditions of the modes $s_{i_{k+j}}$ $j = 2..l$ using the solution of the recurrence relation.
- Compute the actual period of the automata, by computing the time needed for one loop of the system. The period my be generalized depending on n.
- Compute the value of $f_{y_i}(0)$ by following the sequence of modes $s_{i_0}, s_{i_1}, \cdots, s_{i_k}$ leading to the start of the period.
- Compose the general solution of the automata using piecewise and generalized periodic functions.

We implemented the steps into the computer algebra system Maple. Step one and three are automated, step two needs user input to select possible traces.

5.3 Example: Water Level Control

The water level control example can be found in many places in literature [1] [10]. Although, it is a simple system, it allows us to demonstrate our methods. By verifying the safety properties of the water level control system we were able to detect a condition which we did not find in literature.

The system consists of a reservoir which has an intake valve and and an out-flow. The intake valve can be opened or closed. Clearly valves do not open or close instantly, opening or closing takes time. The control has to be designed such that the reservoir is never empty at any point of time.

We can easily identify four modes of the system, (1) the valve is open, (2) the valve is closing, (3) the valve is closed, and (4) the valve is opening. In each of these modes the system can easily be described by a differential equation.

- (1) $y'(t) = a_1$ where a_1 is the rate the reservoir fills if the intake valve is open.
- (2) $y''(t) = a_2$ where a_2 is the closing rate of the valve.
- (3) $y'(t) = a_3$ where a_3 is the outflow-rate of the reservoir as the intake valve is closed.
- (4) $y''(t) = a_4$ where a_4 is the opening rate of the valve.

In contrast to other approaches we actually model the dynamic phase of opening and closing the valve using a second order differential equation and do not just say it takes n seconds to open the valve. The two water levels that trigger to close the valve or to open the valve are c_{12} and c_{34}. The water levels after opening (closing) the

valves are denoted c_{23} and c_{41}, but they will cancel out in the calculation and are just used to set up the symbolic solutions of the differential equations. The transitions of our system are the following: $t_{0,2} = (\mathbf{true}, y := c_{12})$, $t_{1,2} = (y = c_{12},)$, $t_{2,3} = (y' = a_3,)$, $t_{3,4} = (y = c_{34},)$, $t_{4,1} = (y' = a_1,)$, our automata does not contain any actions which would, in fact, make it simple to solve, except in the initial transition.

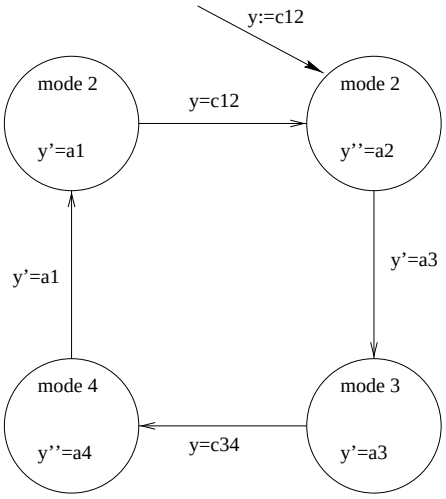

Figure 2: The water level

The Hybrid automata modeling the water level control system contains the parameters $a_1, a_2, a_3, a_4, c_{12}, c_{34}$
Assuming that $c_{12} < c_{34}$, $a_3 < 0 < a_1$ and $a_2 < 0 < a_4$ as first design constrains we observe that the trace is periodic, cycling $1- > 2- > 3- > 4- > 1- > \cdots$.
"Safety properties" of the system

- The reservoir is never empty, $\forall t\ y(t) > 0$.
- The reservoir will never overflow $\forall t\ y(t) < max$, where max is the maximal allowed water level in the reservoir.

We can immediately derive the following design constrains:
$0 \leq c_{12} \leq max$ and $0 \leq c_{34} \leq max$, the sensors which cause the valve to open and close have to lie within the range of the reservoir.
Now, given some a_1, a_2, a_3, a_4 (flow rates and valve closing speeds) we have to determine the conditions on c_{12}, c_{24} and max such that the safety requirements are met.

5.4 Solve Differential Equations

We start in mode one, and solve the differential equations with the initial conditions $y'(t) = a_1\ y(s_1) = c_{41}$ resulting in

$$sol1 = \mathrm{a_1}\, x - \mathrm{a_1}\, \mathrm{s_1} + \mathrm{c_{41}}$$

To switch from mode one to mode two, the water level has to pass the c_{12} mark. Solving we find that this happens at time point

$$\frac{\mathrm{a_1}\, \mathrm{s_1} - \mathrm{c_{41}} + \mathrm{c_{12}}}{\mathrm{a_1}}$$

Now we are in mode two. The water level is c_{12}. The initial flow rate of the water is a_1 since we are coming from mode one. We solve the differential equation $y''(t) = a_2$ under the initial conditions $y(s_2) = c_{12}, y'(s_2) = a_1$ and get:

$$1/2\, \mathrm{a_2}\, x^2 + (-\mathrm{a_2}\, \mathrm{s_2} + \mathrm{a_1})\, x + 1/2\, \mathrm{a_2}\, {\mathrm{s_2}}^2 - \mathrm{s_2}\, \mathrm{a_1} + \mathrm{c_{12}}$$

We have to determine the time point at which the valve is closed. This is the case if $y'(t)$ is a_3. computing the derivative and solving the resulting equation we receive:

$$-\frac{-\mathrm{a_2}\, \mathrm{s_2} + \mathrm{a_1} - \mathrm{a_3}}{\mathrm{a_2}}$$

Now in mode three we have to solve the differential equation $y'(t) = a_3$, we know the water level as we enter this mode by using the previously computed solution and time point.

$$1/2\, \mathrm{a_2}\, x^2 + (-\mathrm{a_2}\, \mathrm{s_2} + \mathrm{a_1})\, x + 1/2\, \mathrm{a_2}\, {\mathrm{s_2}}^2 - \mathrm{s_2}\, \mathrm{a_1} + \mathrm{c_{12}}$$

We will switch to mode 4 as the water level reaches c_{34}, we solve the equation to determine this time point.

$$1/2\, \frac{2\, \mathrm{a_3}\, \mathrm{s_3}\, \mathrm{a_2} + {\mathrm{a_1}}^2 - {\mathrm{a_3}}^2 - 2\, \mathrm{c_{12}}\, \mathrm{a_2} + 2\, \mathrm{c_{34}}\, \mathrm{a_2}}{\mathrm{a_2}\, \mathrm{a_3}}$$

Finally we are in mode four. Similar to mode two we solve the equation $y''(t) = a_4$ with the initial conditions $y(s_4) = c_{34}$ and $y'(s_4) = a_3$ and determine the point of time at which $y'(t) = a_1$, at which the valve is closed again.

$$y_4(t) = 1/2\, \mathrm{a_4}\, x^2 + (-\mathrm{a_4}\, \mathrm{s_4} + \mathrm{a_3})\, x + 1/2\, \mathrm{a_4}\, {\mathrm{s_4}}^2 - \mathrm{s_4}\, \mathrm{a_3} + \mathrm{c_{34}}$$

and the time point where the valve is closed is

$$\frac{\mathrm{a_4}\, \mathrm{s_4} - \mathrm{a_3} + \mathrm{a_1}}{\mathrm{a_4}}$$

Now we computed all solutions for one period of operation.

5.5 Construct Recurrence Relation

Using all the computed solutions we are able to find a recurrence relation which, computed the water level in mode 1 depending on the water level as we entered mode 1 the last time.

$$\begin{aligned} s_1(n+1) := (&\frac{1}{2} a_4 (2\, a_3\, (-\frac{1}{2} \\ &\frac{a_2\, (-2\, a_1\, s1(n)\, a_4 - {a_3}^2 + {a_1}^2 + 2\, c_{34}\, a_4 - 2\, c_{12}\, a_4)}{a_4\, a_1} \\ &-a_1 + a_3) + {a_1}^2 - {a_3}^2 - 2\, c_{12}\, a_2 + 2\, c_{34}\, a_2)/(a_2\, a_3) \\ &-a_3 + a_1)/a_4 \end{aligned}$$

Solving the recurrence relation and computing the water level as we enter state 2,3 and 4 we have all all the pieces and can construct the solution.

$$\mathbf{per}\begin{cases} a_1\,x+\frac{1}{2}\,\frac{-{a_3}^2+{a_1}^2+2\,c_{34}\,a_4}{a_4} & x<-\frac{1}{2}\,\frac{\%1}{a_4\,a_1} \\ \frac{1}{2}\,a_2\,x^2+(\frac{1}{2}\,\frac{a_2\,\%1}{a_4\,a_1}+a_1)\,x+\frac{1}{8}\,\frac{a_2\,{\%1}^2}{{a_4}^2\,{a_1}^2} & x<\frac{\%2}{a_2} \\ a_3\,x+\frac{1}{2}\,\frac{-2\,a_3\,\%2-{a_1}^2+{a_3}^2+2\,c_{12}\,a_2}{a_2} & x<\frac{1}{2}\,\frac{\%3}{a_2\,a_3} \\ \frac{1}{2}\,a_4\,x^2+(-\frac{1}{2}\,\frac{a_4\,\%3}{a_2\,a_3}+a_3)\,x+\frac{1}{8}\,\frac{a_4\,{\%3}^2}{{a_2}^2\,{a_3}^2} & \text{otherwise} \end{cases}$$

$$\%1 := -{a_3}^2+{a_1}^2+2\,c_{34}\,a_4-2\,c_{12}\,a_4$$

$$\%2 := -\frac{1}{2}\,\frac{a_2\,\%1}{a_4\,a_1}-a_1+a_3$$

$$\%3 := 2\,a_3\,\%2+{a_1}^2-{a_3}^2-2\,c_{12}\,a_2+2\,c_{34}\,a_2$$

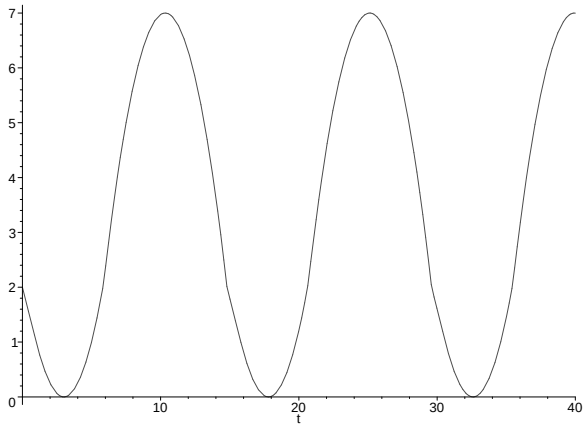

Figure 3: A sample level

5.6 Verification

Now we verify the safety conditions of the system with the goal to derive conditions on the design parameters of the system.

5.6.1 Overflow

We first verify that the reservoir will never overflow. For this we compute the maximum of the water level using the derivative, resulting in:

$$1/2\,\frac{2\,c_{34}\,a_4-{a_3}^2}{a_4}$$

Now the maximal water level has to be smaller than the maximum the reservoir can hold, we solve this resulting condition for c_{34}

$$c_{34}=1/2\,\frac{{a_3}^2+2\,\max a_4}{a_4}$$

It turns out that we cannot freely choose the size of the reservoir. Clearly the longer the valve takes to close the larger the reservoir has to be to prevent overflow.

5.6.2 Empty Reservoir

To verify that the reservoir will not be empty we compute the minimal water level, again using the derivative:

$$1/2\,\frac{2\,c_{12}\,a_2-{a_1}^2}{a_2}$$

The resulting condition is that

$$\left\{1/2\,\frac{{a_1}^2}{a_2}<c_{12}\right\}$$

The minimum possible value for c_{12} is restricted by the closing speed of the valve but further we know that $c_{12} < c_{34} < max$.

We solved many other examples found in literature e.g. reactor temperature control, train crossing and currently investigate an application to robotics and control of a chemical plant.

6 Conclusion

We developed a theory of hybrid systems by considering them as a generalization of differential equations. Similar to initial value problems of differential equations we found a uniqueness theorem. Computing the symbolical solution of a hybrid system allows to introduce formal design parameters, which can be used to verify or optimize the design of the system. The presented symbolic approach enables us to directly derive constrains on design parameters. Using recurrence relations to compute the initial conditions for the modes of a hybrid automata allows, in contrast to fix-point computations, to derive equations which hold for the execution of the systems even if the system does not converge. The presented methods were tested using the classic examples found in literature. We found design constrains on control systems which other analytical methods failed to detect.

References

1. R. Alur, C.Ciyrciybetus, N. Halbwachs T.A. Henzinger P. H. Ho (1995) "The Algorithmic Analysis if Hybrid Systems", *Theoretical Computer Science, 138 1995, pp. 3-34.*

2. R. Alur, T. Henzinger, P.H. Ho (1993) " Automatic Symbolic Verification of Embedded Systems", Proceedings of the 14th Annual IEEE Real-time Systems Symposium (RTSS 1993) pp. 2-11.

3. T. A. Henzinger,"The Theory of Hybrid Automata", *11th Annual IEEE Symposium on Logic in Commuter Science (LICS 96) pp. 278-292* .

4. P. Filipoph, "Differential equations with discontinuous Right-hand Side

5. M. S. Branicky (1998), "A Unified Framework for Hybrid Control: Model and Optimal Control Theory", *IEEE Transactions on Automatic Control, Vol. 43, No. 1, 1998.*

6. M. v. Mohrenschildt, (1998), "A Normal Form for Rings of Piecewise Functions", *Journal of Symbolic Computation, 13 pp.*

7. M. v. Mohrenschildt (1997), "Using Piecewise to Solve Classes of Control Theory Problems", *MapleTech97, Vol .4, No. 3, pp. 33-37.*

8. E.Engeler, M. v. Mohrenschildt, (1994), "Solving Discontinuous Differential Equations", *The Combinatory Program*, Birkhauser, pp. 98-116

9. M v. Mohrenschildt, (Sep 1998), "Solving Discontinuous Ordinary Differential Equations with a Computer Algebra System" submitted to *J. of Symbolic Computation*

10. Hybrid Systems, *Lecture Notes in Computer Science (736)* Springer.

11. F.H. Clarke and others "Non-smooth Analysis and Control Theory"

12. Y. Kesten, A. Pnueli "Integration Graphs: A Class of Decidable Hybrid Systems." *Lect. Notes. Comp. Sci. 736*