Design of Controllers for Linear Hybrid Systems

Rupak Majumdar[1] and R.K. Shyamasundar[2]

[1] Department of Computer Science and Engineering, Indian Institute of Technology, Kanpur 208 016, India

[2] Corresponding Author: Computer Science Group, Tata Institute of Fundamental Research, Homi Bhabha Road, Bombay 400 00, India, e-mail: shyam@tcs.tifr.res.in

Abstract. In this paper, we describe an approach for deriving control laws for linear hybrid systems given the phase transition system and the global invariance. After illustrating the method, we show that the approach can be applied for purposes of verification, viability and discuss extensions to nonlinear hybrid systems.

1 Introduction

Hybrid systems are continuous time systems with continuous variables with phased operations. Thus, hybrid system can be thought of as systems consisting of discrete and continuous state systems; the discrete behaviour corresponds to transitions between phases and continuous behaviour corresponds to the evolutions within each phase. For a transition to take place from one phase to another phase a given condition should be satisfied over the hybrid state space. A variety of process control systems exhibit such a behaviour. Design, analysis and verification of hybrid systems has been an important area of active research [5, 2]. Automata-theoretic and logical approaches have been primarily used for verification and synthesis. In this paper, we describe an approach for deriving control laws for hybrid systems. We first describe a method for linear hybrid systems (ℓ`hs`) and illustrate the method through an example. We then show the applicability of the method for verification and viability and finally, discuss extensions to nonlinear hybrid systems.

2 Background

Several models for hybrid systems have been given in [2, 5]. The definition of a hybrid transition systems (hts) based on [1, 11] is given below.

Definition 1. *Hybrid transition system* (hts) $H = (Q, R^n, \Sigma, \mu_1, \mu_2, E)$ where,

- Q is a finite set of discrete control locations (denoted by subscripts of ℓ or l); R^n is the continuous state of continuous data variables, say X ($|X| = n$); Σ is a set of events; a subset of Σ can be associated with each transition.
- μ_1 is a function associating the continuous data state transformation in terms of time. This could be a set of differential inclusions/ set of differential

equations expressing the dynamics in each phase. By $\mu_{1q}, q \in Q$, we mean the differential equations/inclusion (satisfying the usual *healthiness criteria*) associated with phase q.

– A labeling function $\mu_2 : Q \to 2^{R^n}$ assigns to each location a location-invariant $\mu_2(\ell)$. $\mu_2(\ell)$ is satisfied as long as the system remains in ℓ.
– A set of transitions (or the edges), E, is given by the relation $E \subset Q \times \Sigma \times (R^n \to R^n) \times Q$. Each edge $e = (l_e, \sigma, \mu_3, l'_e)$ has a reset map μ_3 which resets the variables with the transition. The event σ is said to accepted with the transition and is referred to as the switching state. μ_{3e} to denotes the resetting associated with edge e.

The specifications of a hybrid system is formulated as a hts with the following two main invariants:

- *System or global invariants:* These assertions must hold irrespective of the location or state of the system. They apply automatically, if not over-written by local invariants.
- *Location invariants:* is the assertion that is satisfied by the system as long as it remains in ℓ.

Definition 2. A hts, $\mathcal{H}$ is said to be *linear* if it satisfies the following restrictions:

- The activities $\mu_1(\ell)$ in a location ℓ are restricted to linear functions defined by the set of differential inclusions (cf. [4]) of the form $\dot{x} = [k_l, k_u]$, where k_l, k_u are constants. This means that the derivative of x in location ℓ can take any value between k_l and k_u. The set of differential inclusions in ℓ form a convex polyhedral set over the domain of derivatives.
- the location invariant $\mu_2(\ell)$ is defined by a set of linear inequalities over X and has the form $A.x + b \leq 0$, where A, b are constant matrices and $x \in X$.
- The phase transition is accompanied by an update $\mu_3(e)$ of the form $x := R.x + c$, where R, c are constant matrices.

3 Deriving a Control Law

Our main problem is stated below:

> Given the global invariants, a linear hts without the local invariants and the switching state (i.e.,σ in edge e), we will complete the hts with local invariants and the continuous switching states in each of the edges e.

That is, starting from an incomplete hts, we shall arrive at a complete hts whenever feasible and then decide on the control strategy. The problem is also discussed in [10].

3.1 Reachability Between Phases

Basically, from the given incomplete hts, we find out the feasibilities of transitions between phases and try to arrive at a strategy to device the control law that would satisfy the specification. First, we define reachability in terms of the notion of *jump set* and *extended jump set* as in [11].

Definition 3. The reachability problem for a hybrid system H is defined as: *Given* $\mathrm{x}, \mathrm{x}' \in R^n$*, is there a trajectory* ζ *and a time* $t \geq 0$ *such that* $\zeta(0) = \mathrm{x}$ *and* $\zeta(t) = \mathrm{x}'$ *so that at all points* $\zeta(t'), t' \leq t$*, the invariants are satisfied.*

We can extend the definition to regions as follows: Given two polyhedral sets $P, P' \in 2^{R^n}$, are there two points $\mathrm{x} \in P$ and $\mathrm{x}' \in P'$ such that the reachability between x, and x′ is satisfied.

To complete the specifications, we basically ask the following question: Does there exist a control law that makes it possible to reach location ℓ_j from ℓ_i satisfying the global and the local invariant specifications? If no such path exists, then it is clear that the specification is not realizable. If envisaged transitions are indeed reachable, among all locations, then it shows that there are control laws that satisfy the specifications. However, it is not necessary that there should be an unique control law. After ensuring the existence of control laws, we show how we can select a control law according to some *optimality* criteria.

Definition 4. The *jump set* λ for a transition $e = (l, \sigma, \mu_3, l')$ is defined as the set of points in $\mu_2(l)$, which are mapped by μ_{3e} to a subset of $\mu_2(l')$.

$$\lambda_e = \{x \in \mu_2(l) : \mu_{3e}(x) = R_e x + c_e \in \mu_2(l')\}$$

The jump conditions makes sure that the system will always enter into the successor satisfying its invariants. Only points in λ_e can take the transition; others are forbidden. It must be noted that it is not necessary for the current transformation defined by e to take to all the possible states represented by $m_2(\ell')$.

Definition 5. Let D be a linear system and S be a convex polyhedron denoting the valuations of x. We denote by $S \swarrow D$ the set of valuations[3] that can reach S by letting the variables continuously evolve according to a constant derivative in D such that the final state is in S.

$$S \swarrow D = \{v + t \times (-d) | v \in S, d \in D, t \in R^{\geq 0}\}$$

Forward time elapse operator can be similarly defined by $S \nearrow D$:

$$S \nearrow D = \{v + td | v \in S, d \in D, t \in R^{\geq 0}\}$$

Definition 6. An *extended jump set* λ_e^{ext} for a transition e is the set of points in $\mu_2(l)$ that eventually reach λ_e under the dynamics of location l.

---
[3] This is similar to the *backward time elapse* operator given in [6].

$$\lambda_e^{ext} = \lambda_e \swarrow \mu_1(l)) \cap \mu_2(l)$$

The extended jump set guarantees that a transition e will eventually be possible.

First, we verify whether there could be control values for direct transitions between two locations defined in the hts. In the case of a ℓ**hs**, the sets λ_e and λ_e^{ext} can be computed by interpreting the constraints as convex sets in n-dimensions. For a transition from l_i to l_{i+1}, we have the following constraints:

$$\begin{aligned} \mu_2(l_i):&\ A_i x + b_i \leq 0 \\ \mu_2(l_{i+1}):&\ A_{i+1} x + b_{i+1} \leq 0 \\ \mu_{3e}:&\ x := R_e x + c_e \end{aligned}$$

Substituting the third equation in the second, we get

$$\mu_2(l_{i+1})' :\ A_{i+1}.R_e x + A_{i+1} c_e + b_{i+1} \leq 0$$

Thus, the jump-condition λ_e is the intersection of the two convex polyhedra– $\mu_2(l_i)$ and $\mu_2(l_{i+1})'$. Hence,

$$\begin{pmatrix} A_i \\ A_{i+1}.R_e \end{pmatrix} x + \begin{pmatrix} b_i \\ A_{i+1} c_e + b_{i+1} \end{pmatrix} \leq 0$$

To compute λ_e^{ext}, we apply the time elapse operator on the set λ_e and then, find the intersection of this set with $\mu_2(l_i)$. From this, we establish:

Theorem 7. *λ_e^{ext} is a bounded polyhedral set if $\mu_2(l)$ is bounded.*
Proof. The time elapse operator transforms convex polyhedra to convex polyhedra. Given the system of generators in the frame representation of two polyhedra as $P = (V, R), D = (V', R')$, the time-elapsed polyhedron $P \swarrow D$ has the system of generators $(V, R \cup (-V') \cup R')$, where $v \in -V'$ iff $-v \in V'$. Further, the intersection of two polyhedral sets is also polyhedral. Combining the two, and noting that the invariant is a bounded polyhedral set, we get λ_e^{ext} to be bounded polyhedral.

Having found the direct control between locations, let us find a control path between any two locations (between which a topological path exists). For this purpose, let us assume that we have to check the reachability of a certain state s_f from an initial state s_i. If such a path exists, say $\langle s_i, l_1, l_2, \ldots, s_f \rangle$, then a transition from location l_i in the path to l_{i+1} will be possible only when the system is in λ_e^{ext} in l_i. Carrying the argument further, we must ensure in a path $l_{i-1} \rightarrow l_i \rightarrow l_{i+1}$, the transition $l_{i-1} \rightarrow l_i$ transfers the system into the allowed region λ_e^{ext} for the transition $l_i \rightarrow l_{i+1}$ to occur.

Algorithm
The algorithm to generate control laws (if any) basically would systematically test the paths from s_i to s_f, starting at s_f and going backwards. In this way, if λ_e for the transition $s_i \rightarrow l_1$ is not void, then this path satisfies the requirements. Otherwise, we would have to check the next path.
Steps:

1. Test for a control transition from s_i to s_f using simple paths as above.
2. If there is no simple path leading from s_i to s_f, we have to try cycles. Loops may be effective in increasing the extended jump set of a location. For example, in the path $\langle s_i \ldots l_i l_{i+1} s_f \rangle$ let there be a cycle $\langle l_i m_1 m_2 \ldots m_n l_i \rangle$. Suppose in iterating backwards, we have computed $\lambda^{ext}_{(l_i,l_{i+1})}$. Now, we iterate around the cycle $l_i \leftarrow m_n \leftarrow \ldots l_i$. This means that if the trajectory should fall in $\lambda^{ext}_{(l_i,m_1)}$ then the transition $l_i \rightarrow l_{i+1}$ is also guaranteed (by going around the cycle once). Thus, the extended jump set for l_i must be updated to
$$\lambda^{ext}_{(l_i,l_{i+1})} = \lambda^{ext}_{(l_i,l_{i+1})} \cup \lambda^{ext}_{(l_i,m_1)}$$
Updating $\lambda^{ext}_{(l_i,l_{i+1})}$ means that even the other jump sets in the cycle must be updated. So we can go around the cycle once more and update location jump-sets. Finally, a fixed point will be reached when
$$\lambda^{ext*}_{(l_i,l_{i+1})} = \lambda^{ext*}_{(l_i,l_{i+1})} \cup \lambda^{ext}_{(l_i,m_1)}$$
Hence, in checking a path with cycles, we iterate backward up to the beginning of the cycle, then go around the cycle once, updating the jump-sets. If the updated jump-set allows a path from s_i to s_f, we take it. Otherwise, we take another round of the cycle. We carry out this procedure until a path is found or we have reached the fixed point of our iterations. In the second case, we move on to another cycle in the path.

Theorem 8. *If $\mu_2(l_i)$ is bounded then under the assumptions above, the fixed point λ^*_{ext} exists.*

Proof. Let L be a map corresponding to taking the set λ around the cycle once, i.e. it maps a point $x \in \lambda$ to the point it would reach after traversing the cycle once. We define the sequence of sets $\lambda_1 = \lambda^{ext}, \cdots, \lambda_{k+1} = L(\lambda_k)$. Letting $\lambda'_k = \bigcup_{j=1\ldots k} \lambda_j$, we define

$$\lambda'_{k+1} = L(\lambda'_k) \cup \lambda'_k \quad \text{and we get}$$
$$\lambda'^* = \bigcup_{j=1\ldots\infty} \lambda_j$$

Since $\lambda_k \subseteq \mu_2$, λ_i's are bounded, and hence $\langle \lambda'_k \rangle$ is bounded. Any point $p \in \lambda'^*$ satisfies $p \in \lambda_k$ for some k and therefore by applying L^{-1} $(k-1)$ times we get a point $p' \in \lambda_1$. Since the sequence is bounded and monotone, the fixed point λ'^* exists and

$$\lambda'^* = L(\lambda'^*) \cup \lambda'^*.$$

Conditions for Computability: The shortcoming of the above method is that the union of two convex sets is in general non-convex. This means that the sets cannot be represented simply by means of inequalities/convex predicates. The terms in the disjunctive normal form rapidly grow and cannot be handled efficiently. Hence, in such cases, we resorts to approximation techniques. For lack of space, we will postpone further discussion on these aspects and arrive at conditions under which $\lambda^{ext}_{(l_i,l_{i+1})}$ is a subset of $\lambda^{ext}_{(l_i,m_1)}$ and thus, maintaining the convexity.

Theorem 9. *If in the path $\langle s_i, \ldots, l_i, l_{i+1}, \ldots, s_f\rangle$ a cycle $\pi = \langle l_i, m_1, \ldots, m_n, l_i\rangle$ is introduced at position l_i then under the following conditions, the updated set λ^{ext}_{new} remains convex:*

$$\forall m_j \in \pi,\ \mu_2(l_i) \subseteq \mu_2(m_j)$$
$$\forall j,k \in \pi,\ \mu_3(m_j, m_k) : x := x.$$

Proof. *For purposes of convenience, we use $\lambda(\ell, \ell')$ for λ_e corresponding to the edge $e = (\ell, \sigma, \mu_3, \ell')$. We will prove that under the given conditions, $\lambda^{ext}(\ell_i, \ell_{i+1})$ is a subset of $\lambda^{ext}(\ell_i, m_1)$.*

For transition m_n to ℓ_i, we have

$$\lambda(m_n, \ell_i) = \lambda^{ext}(\ell_i, \ell_{i+1}) \cap \mu_2(m_n) = \lambda^{ext}(\ell_i, \ell_{i+1}) \subseteq \lambda^{ext}(m_n, \ell_i)$$

Further, in an iteration from node m_n to any node m_k in the cycle,

$$\lambda(m_k, m_n) = \lambda^{ext}(m_n, \ell_i) \cap \mu_2(m_k)$$

Since $\mu_2(\ell_i) \subseteq \mu_2(m_k)$, we have $\lambda^{ext}(m_n, \ell_i) \cap \mu_2(\ell_i) \subseteq \lambda(m_k, m_n)$, Further, $\lambda^{ext}(m_k, m_n) \cap \mu_2(\ell_i) \subseteq \lambda^{ext}(m_k, m_n)$. Hence, we get

$$\lambda^{ext}(\ell_i, \ell_{i+1}) \subseteq \lambda^{ext}(m_k, m_n)) \cap \mu_2(\ell_i) \subseteq \lambda^{ext}(m_k, m_n)$$

Similarly, for the last part of the cycle, we close back on ℓ_i. For a transition from ℓ_i to m_k, we get on similar lines

$$\lambda(\ell_i, m_k) = \lambda^{ext}(m_k, m_n) \cap \mu_2(\ell_i)$$

From the above two equations

$$\lambda^{ext}(\ell_i, \ell_{i+1}) \subseteq \lambda^{ext}(m_k, m_n) \cap \mu_2(\ell_i) = \lambda^{ext}(\ell_i, m_k).$$

The last equality follows since $\lambda^{ext}(\ell_i, m_k) \subseteq \mu_2(\ell_i)$.

The control synthesis problem can be formulated as a fixed point computation on states, analogous to the verification counterpart. For this, we define the *predecessor operator* π.

It may be noted that the state of a system can be written as

$$P = \{\ell_0\} \times P_0 \cup \ldots \cup \{\ell_m\} \times P_m$$

where $\{\ell_0, \ldots, \ell_m\} = Q$, and P_i's are subsets of R^n. Thus, we can represent system state as a unique tuple of regions on $(P_0, \ldots, P_m)$. We extend the notion of polyhedral sets to polyhedral tuples where all its components are polyhedral sets.

The predecessor function π is a map from set tuples to set tuples and denotes all states that can reach P through a time or a transition step.

If $\pi(P_0, \ldots, P_m) = (P'_0, \ldots, P'_m)$, then P'_i can be written as:

$P_i' = \bigcup_{e=(\ell_i,\sigma,\mu_3,\ell_j)\in E}\{x : x \in \lambda_e^{ext} \wedge \exists x' \in (x \nearrow \mu_1(\ell_i)).\mu_{3e}(x') \in P_j)\}$

This means that x is in P_i' if for some ℓ_j, we can stay in ℓ_i for some time and then take a transition that puts us in $\{\ell_j\} \times P_j$.

Theorem 10. *If $P = (P_0, \ldots, P_m)$ is polyhedral, so is $\pi(P)$.*

We can now formulate the synthesis problem similar to that of the verification problem (discrete case problem is described in [3]). The actual formulation is omitted for lack of space. It should be noted that the algorithm is semi-decidable, because the computation may not converge in a finite number of steps.

3.2 Control Law Generation

We adopt an "as late as possible" (ALAP) strategy. This means that the transition from one location to another is taken only when the trajectory is going to leave the jump-set for a transition. The point where it does so can be computed using the system dynamics. This ensures that the maximum variation of the variables is covered while maintaining all invariants.

We can also introduce, at this point, further constraint conditions which may reflect optimality conditions on the system perhaps including cost criterion.

In a process control application, when the plant has to be circulated between several operating points, we can proceed as follows.

1. First, compute the reachable paths from one control point to another using the system invariants and the above algorithm for all pairs of control locations the plant needs to be cycled around.
2. Then, using the ALAP approach, generate the actual control signals (σ's) to effect the transitions.

3.3 Illustrative Example

Consider a system which has to be driven cyclically between three states. The system can reside in three different control modes – where the dynamics at each mode is given as constant vectors. The system can make transitions from one control location to another. The operating points are (0.05, 0.05), (2.05, 2.05) and (2.05, 3.05). The process is robust, and we can tolerate a deviation of .05 units from the optimal value. We have to design a controller to cycle the system through the three operating points. The values are normalized, and the global constraint is that no variable should exceed the value 4.

The hybrid automaton for the system is given in Figure 1. The global invariance is given as $0 \leq x \leq 4$. The vertices show the different locations, with s_1,s_2 and s_3 being the operating regions. The vector $\dot{x}$ gives the activity, $\mu_1(l)$ of the location. We assume in this example that there are no further local invariants, and that the reset map for each edge is $x := x$ (hence omitted from the figure).

The three operating points, s_1, s_2, s_3, are treated as special locations for notational uniformity. We assume that these regions are reachable bi-directionally from all other locations, and μ_1 for these locations is 0 for all continuous variables. To differentiate these special locations, they are shown with double circles. Note that in our example, the behavior of the continuous variables are governed by differential equations, but these are a special case of differential inclusions. For example, in location l_1, $\dot{x_1} = [1,1]$, but we have denoted this simply by $\dot{x_1} = 1$. The vector $\dot{x}$ in each location has the two components $\dot{x_1}$ and $\dot{x_2}$ corresponding to the two continuous variables.

The solution to the Reachability problem has the possible solutions: $\langle s_1, l_1, l_2, l_3, l_1, s_2\rangle$, $\langle s_2, l_1, l_2, l_3, s_3\rangle$. and $\langle s_3, l_2, l_3, l_1, l_2, s_1\rangle$.

Figure 3 shows how the reach set is constructed from the final location backwards. Since the transition $s_i \to l_1$ at the end of the path is admissible, we conclude that this is a valid path.

First we try the path s_1, l_1, s_2 (fig 2.). Since this does not lead us to our final state, we try the cycle $s_1, l_1, (l_2, l_3, l_1), s_2$. Fig. 3. gives the zone computations. This path leads from s_1 to s_2. Finally, we show that by going around the cycle once, we have reached the fixed point of the iterations, because another round of the cycle produces the same zone as in the first turn. Calculations for $s_2 \to s_3$ and $s_3 \to s_1$ are similar.

Control Law : Water level controller Consider the water level controller described in [1]. The global invariant is $1 \leq y \leq 12$. The two states are open and shut where open: $\dot{y} = -2$ and shut: $\dot{y} = 1$

We model the 2-unit delay in the edge-update condition by the settings in the transitions as follows: open $\to$ shut: $y := y - 4$ and shut $\to$ open: $y := y + 2$. This is illustrated in Figures 4-5. Assuming the initial state to be characterized by $y = 5$, the jump sets for the transitions can be computed as follows:

- open $\to$ shut: $5 \leq y \leq 12$
- shut $\to$ open: $1 \leq y \leq 10$.

Adapting an ALAP transition policy, we arrive at the following switching transitions:

- $y = 5$ – open to shut
- $y = 10$ – shut to open

These are the same as in the standard example.

4 Viability, Verification and Generalizations

Our approach can in fact be used for verifying various properties. The standard control examples (e.g. water level controller, temperature controller) deal with safety properties of the system. That is, instead of looking for any special locations to cycle the system through, we are more generally interested in the transition characteristics of the system that satisfies all local and global invariants. Such safety analysis becomes quite easy in our model; the transition conditions to satisfy the safety properties are precisely the jump sets we need to compute. These will act as transition guards on the edges. Then the ALAP control policy would set up transition guards identical to the standard examples. Further several liveness properties can also be verified on the lines of the approach in section 3.

To guarantee that a given jump would always be possible, we should check the viability of the system.

Viability denotes the system's ability to take an infinite number of transitions [4]. This places a restriction on the evolution since at each step, some transition must remain unblocked. Using the concept of a viability kernel, we can decide the region of the state space in which the automaton must lie in order to satisfy the viability requirement.

Definition 11. We define the switch-and-coast operator $n_e : 2^{R^n} \rightarrow 2^{R^n}$ for the edge $e = (l, \sigma, \mu_3, l')$ as follows: $n_e(Y) = \lambda_e \cap \mu_{3e}^{-1}(Y \swarrow \mu_1(l'))$.

Thus, $n_e(Y)$ is the set of states from which a transition through e can be immediately taken and then Y can be reached after some time in l'. Then the following two properties are satisfied by n_e.

Theorem 12. *1. n_e is monotone, i.e. $Y_1 \subset Y_2 \Rightarrow n_e(Y_1) \subset n_e(Y_2)$.*
2. n_e distributes over $\cup$, i.e. $n_e(Y_1 \cup Y_2) = n_e(Y_1) \cup n_e(Y_2)$.

Definition 13. We define $out(l)$ as the set of edges that exit ℓ. Then, let the $|E|$ edges of the given ℓhsbe ordered as $\{e_1, e_2, \ldots, e_{|E|}\}$. Let $\Lambda = (\lambda_{e_1}, \lambda_{e_2}, \ldots, \lambda_{e_{|E|}})^T$.

Definition 14. Define $N : [2^{R^n}]^{|E|} \rightarrow [2^{R^n}]^{|E|}$ as $[N(Z)]_i = n_{e_i}(\cup_{j | e_j \in out(l'_{e_i})} Z_j)$

$N(Z)_i$ is the set of continuous states from which a transition through e_i can be taken immediately, given the current location is l_{e_i}, and then some Z_j can be reached after some time in l'_{e_i}, where e_j is an edge that exits l'_{e_i}. Then, if k transitions are to be taken beginning with e_i, with the possibility of extending to a $k+1$th transition, the continuous state must be in $[N^k(\Lambda)]_i$. Note that N is monotone over k, i.e. $N^{k+1}(\Lambda) \subset N^k(\Lambda)$ for all k.

We can now define the viability kernel as the set $\Lambda^* = \cap_k N^k(\Lambda)$. From the monotonicity of N, we get a computability condition on the viability kernel through the fixed point of the iteration $X_{k+1} = N(X_k), X_0 = \Lambda$. For, suppose there exists k such that $N^{k+1}(\Lambda) = N^k(\Lambda)$. Then

$$N^{k+2}(\Lambda) = N(N^{k+1}(\Lambda)) = N^k(\Lambda).$$

From the monotonicity of N and the definition of Λ^*, we get $\Lambda^* = N^k(\Lambda)$. The conditions under which such a k will exist is given in [4].

Illustrative Example: Viability Let us call the edge open→shut, e_1, and the edge shut→open, e_2, according to the convention above. Then, we can see that $N(\Lambda)_1 = n_{e_1}(\lambda_2) = [5, 12]$, and $N(\Lambda)_2 = n_{e_2}(\lambda_1) = [1, 10]$. This implies $\Lambda = N(\Lambda)$, and therefore, $\Lambda^* = \Lambda$. This shows that the transitions e_1 and e_2 both leave the system in its viability kernel. In other words, the operation of the water level controller is viable.

4.1 Nonlinear Systems

Here we show that the class of solvable nonlinear automata can be reduced to a linear automaton through a bisimulation and so,the methods developed earlier will work for these systems after some pre-processing.

Let A be a hts. The non-clock data variables $x_1, x_2, \ldots, x_n$ are said to be *determined* in the location v of A if (1) the governing equation for $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ at v is given by the equation $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) = (f_1, f_2, \ldots, f_n)$ where each $f_i = f_i(x_1, x_2, \ldots, x_n)$. The initial value problem $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \mathbf{x}(0) = \mathbf{x}_0$ has an algebraic solution $\mathbf{x}_{x_0}(t)$, and for each finite constant c appearing in a term $x_i\ \rho c, \rho \in \{\leq, =, \geq\}$, $x_i(t) - c$ has a finite number of roots.

Note that the value of $\mathbf{x}$ at any time t in location l is given by $\mathbf{x}_{x_0}(t)$.

The location l called *definite* for $\mathbf{x}$ if the initial condition of l implies $\mathbf{x} = \mathbf{c}$, for some initial value vector $\mathbf{c} \in R^n$. The edge e is definite if whenever $\mu_1(l) \neq \mu_1(l')$, we have $\mu_3(e) : \mathbf{x}' = \mathbf{c}$, for some arrival value vector $\mathbf{c}$.

Translation Procedure: We transform each non-linear variable to a clock. To each location l, we associate a set $init(l)$ of arrival values of the variables. Now each location is split into $|init(l)|$ locations each of which contain a unique value of the variables $\mathbf{x}$ (which is taken as $\mathbf{x}_0$, the initial value of $\mathbf{x}$. Then we replace the variables x_i by a series of clock variables t_i in all the initial, accepting and invariant conditions in x_i as follows:

1. The initial value for t_i at any location is 0 with the governing equation t_i is $\dot{t_i} = 1$.
2. The value of x_i at any time is given by $x_i(t_i)$ where $x_i(t)$ is the solution of $\dot{x} = f(x)$ with the initial condition x_{0i}.
3. *Invariance.*If the term $x_i \leq x_f$ appears in the location invariant ,we change it to a condition on t_i as follows: If $x_{0i} > x_{f\,i}$ then this location cannot be visited. Else, solve the equation (1) for $x_i(t)$ and find the smallest positive root of the equation $x_i(t_i) - x_{f\,i} = 0$ for t_i. If no such root exists, then the control can stay here forever. So, the invariance condition is 'true'. Else, if r_i is the smallest positive root of the equation, we replace the condition on x_i by the condition $t_i \leq r_i$.

Using the theorem from [8], it can be shown that the initialized multirate automata and the initialized stopwatch automata are timed bisimilar – from which the decidability follows. Since, the initialized rectangular automata form the largest class of such decidable systems (cf. [7]) any effort to generalize will lead to undecidability.

5 Discussion

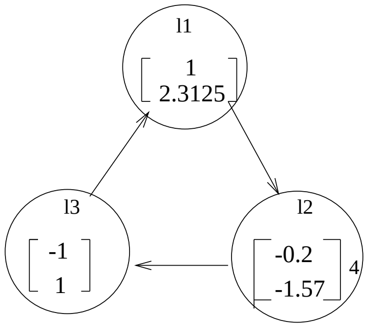
In the previous sections, we have described an approach for deriving control laws for a given ℓhsusing classical approaches. Most of the standard examples can be analyzed using this approach. For the computational aspects discussed in section 3.1, it appears to be profitable to look for an interpolation operator which would give an estimate of the extended set. One can arrive at operators preserving convexity at the cost of losing some valid region of the extended jump set. It must be noted that unlike the case of verification, we cannot overestimate our region. We are working towards the implementation of such an operator based on the canonical representation of linear constraints [9] and abstract interpretation.

Acknowledgments

The authors thank the referees for their valuable comments and pointing out the work [10]. The first author thanks Jawaharalal Nehru Centre for Advanced Research, Bangalore for the support. The research of the second author was partially supported by Indo-French Centre, IFCPAR, New Delhi.

References

1. R.Alur, C.Courcoubetis, N. Halbwachs, T.A.Henzinger and P.-H.Ho, X. Nocollin, A. Olivero, J. Sifakis, and S. Yovine, *The algorithmic analysis of hybrid systems*, TCS, 138, pp.3-34, 1995
2. P. Antsaklis, W. Kohn, A. Nerode and S. Shastry (Eds.), *Hybrid Systems*, LNCS, 999, Springer-Verlag, 1995.
3. E.Asarin, O.Maler, A.Pnueli, *Symbolic Controller Synthesis for Discrete and Timed Systems*, in [2].
4. A.Deshpande and P.Varaiya, *Viability in Hybrid Systems*, in [2], pp. 128-147.
5. R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel (Eds.), *Hybrid Systems*, LNCS, 736, 1993.
6. N.Halbwachs, P.Raymond, Y.-E.Proy. *Verifying linear hybrid systems by means of convex approximations*, Proc. SAS 94, LNCS 864, Sept. 1994.
7. T.A.Henzinger, W.Kopke, A.Puri, P.Varaiya, *What's decidable about Hybrid automata*, Proc. 27th ACM STOC 1995.
8. P.-H.Ho, *The Algorithmic Analysis of Hybrid Systems*, Ph.D Thesis, Cornell University, 1995.
9. J.L.Lassez and K.McAloon, *Independence of Negative Constraints*, TAPSOFT 1989.
10. M.Tittus and B.Egardt, *Controllability and Control-law synthesis for linear hybrid systems*, Lecture Notes in Control and Information Sciences, Vol 199, Springer-Verlag, 1994.
11. M.Tittus and B.Egardt, *Hybrid Objects*, in [AKNS95], pp. 493-508.



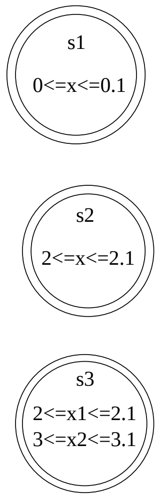


Fig 1. Hybrid Automaton for example

Invariance: 0<= x <= 4

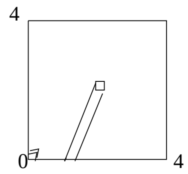
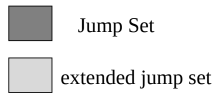


Fig 2. s1->l1->s2

(No feasible path from s1 to s2)



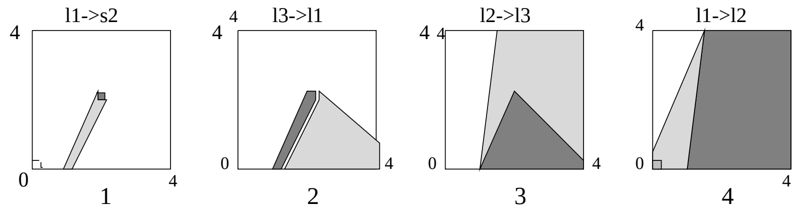


Fig 3: Computation of Jump Sets ($s_1 \rightarrow l_1 \rightarrow l_2 \rightarrow l_2 \rightarrow l_1 \rightarrow s_2$)

(s1->l1 transition is now possible)

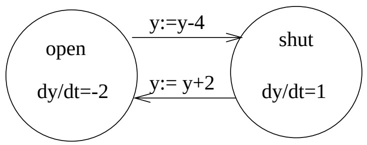
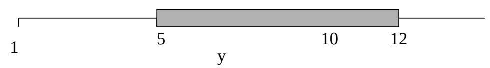


Fig 4. Water Level Controller



Jump Set for the transition open-> shut

Jump Set for the transition shut->open

Fig 5. Jump Sets for safety condition