

Computing reachable sets for uncertain nonlinear hybrid systems using interval constraint propagation techniques[☆]

Nacim Ramdani^a, Nediialko S. Nediialkov^b

^a*CERTES, Université Paris-Est Créteil, FR-9400, France (e-mail:nacim.ramdani@u-pec.fr)*

^b*Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada, L8S 4K1 (e-mail: nediialk@mcmaster.ca)*

Abstract

We investigate solution techniques for numerical constraint satisfaction problems and validated numerical set integration methods for computing reachable sets of nonlinear hybrid dynamical systems in presence of uncertainty. To use interval simulation tools with higher dimensional hybrid systems, while assuming large domains for either initial continuous state or model parameter vectors, we need to solve the problem of flow/sets intersection in an effective and reliable way. The main idea developed in this paper is first to derive an analytical expression for the boundaries of continuous flows, using interval Taylor methods and techniques for controlling the wrapping effect. Then, the event detection and localization problems underlying flow/sets intersection are expressed as numerical constraint satisfaction problems, which are solved using global search methods based on branch-and-prune algorithms, interval analysis and consistency techniques. The method is illustrated with hybrid systems with uncertain nonlinear continuous dynamics and nonlinear invariants and guards.

Keywords: Continuous-time systems. Hybrid systems. Interval analysis. Nonlinear systems. Reachability. Uncertain systems.

1. Introduction

Reachability analysis is an important step when addressing verification or synthesis tasks with hybrid systems. Several methods have been developed recently for computing over-approximations of reachable sets using various set representations [2, 3, 4, 5, 6, 7, 8, 9] or hybrid abstractions [10, 11, 12, 13, 14, 15, 16, 17]. When the continuous dynamics is nonlinear, approximate numerical simulation tools [18] are also used. In fact, when used with nonlinear dynamics, conventional numerical integration techniques derive approximations with unknown global error. Few authors have investigated *validated* methods to derive enclosures that contain true solutions of nonlinear continuous dynamics. Validated set integration methods based on interval Taylor methods (ITM) (see [19] for a review) have already been used for hybrid systems simulation and analysis. In [20], ITM were used for rigorous

[☆]This is a revised and extended version of an earlier paper [1]

simulation of hybrid systems, and an effective technique was developed to enclose as tightly as possible mode switching points. ITM were used within the HYPERTECH tool [21] for the analysis of hybrid systems with nonlinear continuous dynamics. In [22], comparison theorems for differential inequalities were used with ITM to develop a more efficient nonlinear set integration method, when large domains are taken for either initial continuous state or parameter vectors. In [21, 22], event detection is done simply by using the *a priori* solution enclosure set, that is, the rough set given by the Picard-Lindelöf operator [19], which contains whole flow pipe between two integration time steps, whereas event localization within these time intervals was not addressed. To extend these interval simulation methods to hybrid systems of higher dimension, and in presence of large uncertainties, we must solve the problem of flow/sets intersection in an effective and reliable way. Event detection with nonlinear differential equations has benefited from a large amount of work, and there are efficient algorithms, which can solve event detection and localization [23, 24]. Here, we consider flow pipes of non trivial size intersecting guard sets, thus leading to a continuum of time instants, where events may occur. Our objective is to enclose the time intervals, where these events occur, and compute a reliable enclosure of jump transitions including reset functions.

The main idea developed in this paper is first to use ITM, along with methods for controlling the wrapping effect, to derive an analytical expression for the boundaries of continuous flow pipes. Then, using these expressions, the event detection and localization problems underlying flow/sets intersection can naturally be expressed as numerical constraint satisfaction problems [25, 26, 27]. Finally, the latter are solved using global search methods based on branch-and-prune algorithms, interval analysis and consistency techniques.

The idea of using constraint propagation techniques for the safety verification of hybrid systems is not new. Such techniques were used within an abstraction refinement framework [28], that is, without computing explicitly reachable sets. To the contrary, the present work addresses the explicit computation of reachable sets without abstraction. Furthermore, interval constraint propagation techniques allied with ITM were integrated recently into a SAT-modulo-theory approach to facilitate automated reasoning about large boolean combinations of non-linear arithmetic constraints involving ODEs [29].

Finally, the methods advocated in this paper bring a technical improvement to the hybrid reachability scheme introduced in [21] and to the event localization method suggested in [20, 24], when initial state and parameter vectors are taken in domains of large size.

The structure of this paper is as follows. Sect. 2 introduces notation and the problem we study. Sect. 3 overviews the interval tools we use for solving flow intersection with invariants and guards, and evaluating reset functions. Sect. 4 shows how to compute analytical expressions for flow boundaries using ITM. Sect. 5 contains the main contribution of this paper, namely, solving flow/invariants and flow/guards intersection. The method is illustrated in Sect. 6, with two nonlinear hybrid systems in presence of parametric uncertainty.

2. Hybrid automata

We consider a hybrid automaton [2] given by $H = (\mathcal{Q}, \mathcal{D}, \mathcal{P}, \Sigma, \mathcal{A}, \text{Inv}, \mathcal{F})$. Here \mathcal{Q} is a set of locations. Given a location $q \in \mathcal{Q}$, the continuous dynamics, and hence flow transitions,

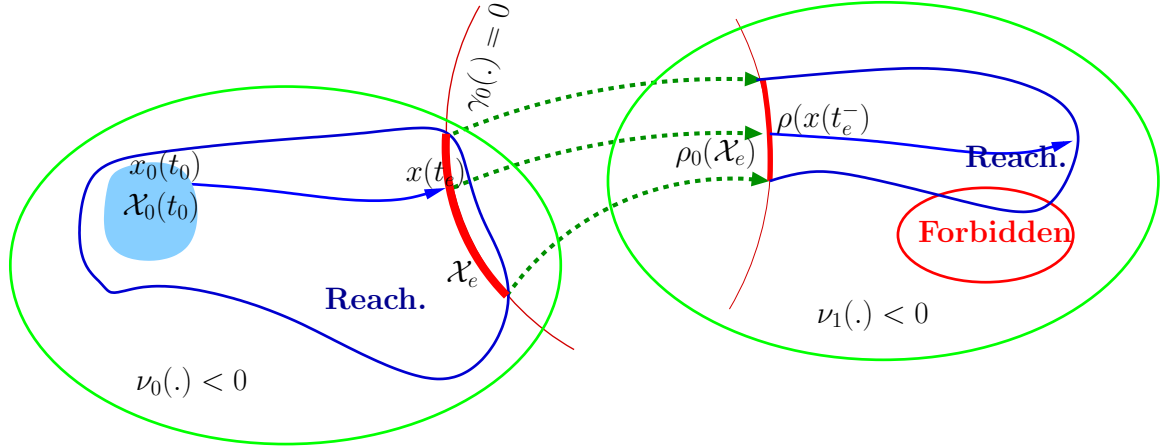


Figure 1: Set reachable in finite time by system (1-2). When starting from an initial state vector $x(t_0)$ taken in $\mathcal{X}(t_0)$, a discrete transition occurs, if continuous flow intersects the guard set at time t_e . Then continuous state vector is reset as $x_1(t_e^+) = \rho_0(x_0(t_e^-))$. \mathcal{X}_e is the set of all possible $x(t_e)$ when $x(t_0)$ varies in $\mathcal{X}(t_0)$. Reachable set may intersect a forbidden area.

are described by non-autonomous differential equations $f_q \in \mathcal{F}$ of the form

$$\text{flow}(q) : \quad \dot{x}(t) = f_q(x, p, t), \quad (1)$$

where $f_q : \mathcal{D} \times \mathcal{P} \times \mathbb{R}^+ \mapsto \mathcal{D}$ is nonlinear and assumed sufficiently smooth over $\mathcal{D} \subseteq \mathbb{R}^n$, with dimension n that may depend on q , and $p \in \mathcal{P}$, where \mathcal{P} is an uncertainty domain for the parameter vector p . Inv is an invariant, which assigns a domain to the continuous state space of each location. It is defined by the following system of inequalities:

$$\text{Inv}(q) : \quad \nu_q(x(t), p, t) < 0,$$

where inequalities are taken componentwise, $\nu_q : \mathcal{D} \times \mathcal{P} \times \mathbb{R}^+ \mapsto \mathbb{R}^m$ is also nonlinear, and the number m of inequalities may also depend on q . The set \mathcal{A} is the set of discrete transitions $\{e = (q \rightarrow q')\}$ given by the 5-uple $(q, \text{guard}, \sigma, \rho, q')$, where q and q' represent upstream and downstream locations respectively; guard is a condition given as the system of equations

$$\text{guard}(e) : \quad \gamma_e(x(t), p, t) = 0; \quad (2)$$

σ is an event, and ρ is a reset function. In this paper we consider uncertain, but simple affine reset functions; the case of nonlinear reset functions can be addressed using existing set computation techniques (see also Remark 4).

A transition $q \rightarrow q'$ occurs when continuous state flow reaches the guards set, i.e. when the continuous state satisfies condition (2), as depicted in Fig. 1.

Remark 1. We can use equality in (2), because we use global search methods capable of detecting and localizing events that may occur on sets of zero measure. Of course, we can also use inequality guard conditions if necessary.

For simplicity, we introduce a new state variable $z = (x, p, t)$ with $\dot{z} = (\dot{x}, 0, 1)$ and let $\mathcal{Z} = \mathcal{D} \times \mathcal{P} \times \mathbb{R}^+$. System (1–2) becomes

$$\text{flow}(q) : \dot{z}(t) = f_q(z), \quad (3)$$

$$\text{Inv}(q) : \nu_q(z(t)) < 0 \quad \text{and} \quad (4)$$

$$\text{guard}(e) : \gamma_e(z(t)) = 0. \quad (5)$$

Now all uncertain quantities are embedded in the initial state vector. In fact, systems of differential equations with nonlinear guards can be transformed to equivalent systems with linear guards by appending a new state variable $y = \gamma_e(z)$ [24]. Our method does not require this transformation. As a consequence, we do not need to increase the size of the continuous state vector in (3).

3. Solving constraint satisfaction problems with interval methods

In this section, we overview key concepts regarding methods based on interval analysis that we use for finding intersections with invariants and guards, and evaluating jump functions.

Consider the system of m (in)equalities over n variables $z \in \mathbb{R}^n$

$$\mathcal{C} : \bigwedge_{1 \leq i \leq m} (h_i(z) \prec 0), \quad \prec \in \{=, <\} \quad (6)$$

and denote the domain of z by \mathcal{Z} . Here inequalities are considered when one computes mode invariants or prune solution sets from those parts that are not contained in invariants. Equalities are considered when one addresses flow/guard intersection and the evaluation of jump successors.

System (6) is a *numerical constraint satisfaction problem* $\text{CSP} : (\mathcal{Z}, \mathcal{C})$. Denoting by \mathcal{S} its set of solutions, we have

$$\mathcal{S} = \{z \in \mathcal{Z} \mid \bigwedge_{1 \leq i \leq m} (h_i(z) \prec 0)\}. \quad (7)$$

An enclosure of \mathcal{S} can be computed in a reliable and guaranteed way via branch-and-prune approaches using interval analysis [27].

3.1. Interval analysis

A real interval $\mathbf{a} = [\underline{a}, \bar{a}]$ is a closed subset of \mathbb{R} . We have $\text{Inf}(\mathbf{a}) = \underline{a}$ and $\text{Sup}(\mathbf{a}) = \bar{a}$. The set of all real intervals is denoted by \mathbb{IR} . Real arithmetic operations are extended to intervals. Consider $\circ \in \{+, -, *, \div\}$ and \mathbf{a} and \mathbf{b} intervals. Then

$$\mathbf{a} \circ \mathbf{b} = [\inf_{u \in \mathbf{a}, v \in \mathbf{b}} u \circ v, \quad \sup_{u \in \mathbf{a}, v \in \mathbf{b}} u \circ v].$$

An interval vector is the Cartesian product of n intervals. The set of n -dimensional real interval vectors is denoted by \mathbb{IR}^n . For an n -dimensional real vector $u \in \mathbb{R}^n$ and an n -dimensional real interval vectors $\mathbf{a}, \mathbf{b} \in \mathbb{IR}^n$, we have $u \in \mathbf{a} \Leftrightarrow \forall i \ u_i \in \mathbf{a}_i$, and $\mathbf{a} \subseteq \mathbf{b} \Leftrightarrow$

$\forall i \mathbf{a}_i \subseteq \mathbf{b}_i$. Consider $g : \mathcal{D}_g \subseteq \mathbb{R}^n \mapsto \mathbb{R}^m$. The range of g over an interval vector $\mathbf{a} \subseteq \mathcal{D}_g$ is given by $g(\mathbf{a}) = \{g(u) \mid u \in \mathbf{a}\}$. An inclusion function for g can be obtained by replacing each occurrence of a real variable by the corresponding interval and each standard function by its interval counterpart. The quality of this inclusion function depends on the formal expression for g .

Given a bounded set \mathcal{E} of complex shape, one usually defines an axis-aligned box or a paving, i.e. a union of non-overlapping boxes $\overline{\mathcal{E}}$, which contains the set \mathcal{E} : this is known as an *outer* approximation of it. Likewise, one also defines an *inner* approximation $\underline{\mathcal{E}}$ which is contained in the set \mathcal{E} . Hence, we have the following property $\underline{\mathcal{E}} \subseteq \mathcal{E} \subseteq \overline{\mathcal{E}}$.

3.2. Branch-and-prune algorithms

Consider system (6) and the case when \prec is $=$. Constraint-satisfaction algorithms work as follows.

Algorithm Interval-Solve(input : ‘ $\wedge_{1 \leq i \leq m} h_i(z) = 0$ ’, \mathbf{z} ; output : list of boxes $\overline{\mathcal{S}}$)

1. define a running list of boxes \mathcal{L} and initialize it with $\mathbf{z} = \text{Hull}(\mathcal{Z})$.
2. while list \mathcal{L} is not empty do
3. pick first box \mathbf{z} from the list
4. evaluate $h_i(\mathbf{z})$
5. if $(\exists i : 0 \notin h_i(\mathbf{z}))$ discard box \mathbf{z}
6. else if $((\|\mathbf{z}\| < \epsilon_1) \text{ or } (\max_i \|h_i(\mathbf{z})\| < \epsilon_2))$
 store box \mathbf{z} in list $\overline{\mathcal{S}}$
7. else partition \mathbf{z} and store new boxes in \mathcal{L}

When \prec is $<$, it suffices to replace the test at step 5 by $\exists i : \text{Inf}(h_i(\mathbf{z})) \geq 0$.

Clearly, this simple algorithm is of exponential complexity. There are several technical and heuristic improvements, which make it possible to control the overall computation time and memory usage [27, 30]. These improvements address inclusion functions at step 4 using monotonicity as follows

$$\text{if } \text{Inf} \left(\frac{\partial h_i}{\partial z_k}(\mathbf{z}) \right) > 0 \Rightarrow \text{Sup}(h_i(\mathbf{z})) = \text{Sup}(h_i(\mathbf{z}_1, \mathbf{z}_2, \dots, \text{Sup}(\mathbf{z}_k), \dots, \mathbf{z}_n)).$$

They also address partitioning strategies, and the possible use of interval narrowing procedures on box \mathbf{z} .

3.3. Interval narrowing procedures

The idea underlying these procedures is to use a reduction function that reduces the size of box \mathbf{z} during the branching scheme of algorithm *Interval-Solve* without using bisection. This reduction can be achieved by an interval narrowing operator for (6) on \mathbf{z} , which we write as

$$\mathbf{z}' = \text{Interval-narrowing}(\mathcal{C}, \mathbf{z}).$$

This operator removes from \mathbf{z} all parts that do not contain a solution to (6) and satisfies the following properties: (a) $\mathbf{z}' \subseteq \mathbf{z}$ and (b) $\mathbf{z}' \cap \mathcal{S} = \mathbf{z} \cap \mathcal{S}$, where \mathcal{S} is the solution set (7).

Most narrowing operators use consistency filtering techniques (e.g. [31], see also the review [32]) and/or constraint propagation [27]. Interval propagation techniques are based on the interval extension of the local Waltz filtering [25, 33, 26]. Consistency filtering techniques rely on local consistency properties.

Remark 2. A *global* consistency property requires that one can find an assignment, that is, a value for all the variables such that all constraints are satisfied at the same time. A *local* consistency property requires that one can find a value for all the variables such that each constraint, when taken individually, is satisfied. Clearly, local consistency is a weaker requirement than global consistency.

The interesting part is that whenever a consistency property is violated, there is an associated rule for pruning some interval. Most solvers use a local consistency named *arc-consistency*. Assume c is a k -ary constraint over variables $z = (z_1, \dots, z_n)$; c is arc-consistent if, for any $z_l \in \mathbf{z}_l$, there exists at least one value in each domain \mathbf{z}_m , $m \neq l$, such that c holds. 2B-consistency and 3B-consistency are relaxations of arc-consistency, which are more easily verified.

Here we use 3B-consistency [31]. In this method, we consider each variable z_l in turn and its range $[\underline{z}_l, \bar{z}_l]$. Let \hat{z}_l be a point within this range. We first calculate the interval evaluation of the functions in the system (6) with the full ranges for all the variable except for the variable z_l , where the range will be $[\underline{z}_l, \hat{z}_l]$. If one of the constraints is not satisfied, then we may reduce the range of the variable z_l to $[\hat{z}_l, \bar{z}_l]$. Then we repeat this procedure with $[\hat{z}_l, \bar{z}_l]$ until all constraints are satisfied. Then we proceed with this ‘*shaving*’ procedure for the upper bound of \mathbf{z}_l . Here, we use as first \hat{z}_l the value $\underline{z}_l + \delta z_l$, where δz_l is given by the user, and at each step δz_l is augmented (by a factor 2).

Numerical implementation. The above system solving methods are implemented in the ALIAS C++ library [34]. We use the `Profil/Bias` C++ class library [35] for interval computation and the `FABDAB++` package [36] for automatic differentiation.

4. Enclosing flows with interval Taylor methods

4.1. Guaranteed set integration

Consider the *uncertain* dynamical system described by (3). Recall that $z = (x, p, t)$. Denote by \mathcal{Z}_0 the initial domain for state vector $z(t_0)$ at time $t_0 \geq 0$. Let us denote by $\mathcal{Z}(t; t_0, \mathcal{Z}_0)$ the set of solutions of (3) at time t originating from each initial condition in \mathcal{Z}_0 at t_0 . Define a time grid $t_0 < t_1 < t_2 < \dots < t_{n_T}$, which is taken equally spaced in this paper, and assume $\mathcal{Z}_0 = \mathbf{z}_0 = [\underline{z}_0, \bar{z}_0]$.

Guaranteed set integration via interval Taylor methods compute interval vectors \mathbf{z}_j , $j = 1, \dots, n_T$, that are *guaranteed* to contain the set of solutions $\mathcal{Z}(t_j; t_0, \mathcal{Z}_0)$ of (3) at time t_j . They first verify existence and uniqueness of the solution using the Banach fixed point theorem and the Picard-Lindelöf operator [37, 38, 39] and compute an a priori enclosure $\tilde{\mathbf{z}}_j$ such that $\tilde{\mathbf{z}}_j \supseteq \mathcal{Z}(t)$ for all t in $[t_j, t_{j+1}]$ [19, 40]. A tighter enclosure for the set of solutions

of (3) at t_{j+1} is then computed using a Taylor series expansion of order k of the solution at t_j , where $\tilde{\mathbf{z}}_j$ is used to enclose the remainder term. Now, it is easy to see that this expansion is valid for any t in $[t_j, t_{j+1}]$. Consequently, the set of solutions of (3) at time t in $[t_j, t_{j+1}]$ is simply given by

$$\mathcal{Z}(t; t_j, \mathbf{z}_j) \subseteq \mathbf{z}(t; t_j, \mathbf{z}_j) = \mathbf{z}_j + \sum_{i=1}^{k-1} (t - t_j)^i \mathbf{f}_q^{[i]}(\mathbf{z}_j) + (t - t_j)^k \mathbf{f}_q^{[k]}(\tilde{\mathbf{z}}_j). \quad (8)$$

Eq. (8) is an analytical expression for the boundaries of the continuous flow characterized by (3). The coefficients $\mathbf{f}_q^{[i]}(\mathbf{z}_j)$ are the Taylor coefficients of the solution, which are computed numerically via automatic differentiation. It is well known that the scheme (8) is width increasing, and thus not suitable for numerical implementation. Effective methods use the mean-value form, matrix preconditioning and linear transformations.

In this paper we control wrapping using the *mean-value* approach [19]. At each time step t_j , the solution enclosure is computed in the form

$$\mathcal{Z}(t_j; t_0, \mathcal{Z}_0) \in \{v_j + A_j r_j \mid v_j \in \mathbf{v}_j, r_j \in \mathbf{r}_j\}. \quad (9)$$

The performance of the method relies significantly on the choice of matrices A_j . An effective method introduced by Lohner uses QR-factorization [19].

4.2. An analytical expression for the continuous flow pipe

Let us extend the mean-value approach to characterize the solution enclosure at time $t \geq t_0$, which is not necessarily on the time-grid $\{t_0, t_1, t_2, \dots, t_{n_T}\}$. This solution enclosure can in fact be computed in the form

$$\mathcal{Z}(t; t_0, \mathcal{Z}_0) \in \{v(t) + A(t)r(t) \mid v(t) \in \mathbf{v}(t), r(t) \in \mathbf{r}(t)\},$$

and we can define a compound form $\boldsymbol{\chi}(t)$ for the solution enclosure at time t as follows

$$\boldsymbol{\chi}(t) \equiv \{\mathbf{z}(t), \hat{\mathbf{z}}(t), \mathbf{v}(t), \mathbf{r}(t), A(t)\}. \quad (10)$$

where $\hat{\mathbf{z}}(t) := \text{Mid}(\mathbf{z}(t))$. We can now write an analytical expression for the solution set of (3) at time $t \in [t_j, t_{j+1}]$, as follows:

Algorithm φ (input : $\boldsymbol{\chi}_j, t_j, t, \tilde{\mathbf{z}}_j$, output : $\boldsymbol{\chi}(t)$)

1. $\mathbf{v}(t) := \hat{\mathbf{z}}_j + \sum_{i=1}^{k-1} (t - t_j)^i \mathbf{f}_q^{[i]}(\hat{\mathbf{z}}_j) + (t - t_j)^k \mathbf{f}_q^{[k]}(\tilde{\mathbf{z}}_j)$
2. $\mathbf{S}(t) := I + \sum_{i=1}^{k-1} (t - t_j)^i \frac{\partial \mathbf{f}_q^{[i]}}{\partial \mathbf{z}}(\mathbf{z}_j)$
3. $\mathbf{q}(t) := (\mathbf{S}(t)A_j)\mathbf{r}_j + \mathbf{S}(t)(\mathbf{v}_j - \hat{\mathbf{z}}_j)$
4. $\mathbf{z}(t) := \mathbf{v}(t) + \mathbf{q}(t)$
5. obtain $A(t)$ via QR-factorization of $\text{Mid}(\mathbf{S}(t)A_j)$
6. $\mathbf{r}(t) := A(t)^{-1}(\mathbf{S}(t)A_j)\mathbf{r}_j + (A(t)^{-1}\mathbf{S})(\mathbf{v}_j - \hat{\mathbf{z}}_j)$
7. $\hat{\mathbf{z}}(t) := \text{Mid}(\mathbf{v}(t))$
8. $\boldsymbol{\chi}(t) := \{\mathbf{z}(t), \hat{\mathbf{z}}(t), \mathbf{v}(t), \mathbf{r}(t), A(t)\}$

(Here Mid is componentwise midpoint and I is the identity matrix). Finally, solution enclosure at time t_{j+1} is given by $\boldsymbol{\chi}_{j+1} = \varphi(\boldsymbol{\chi}_j, t_j, t_{j+1}, \tilde{\mathbf{z}}_j)$.

4.3. An inclusion function for the continuous flow pipe

It is now interesting to see how we can compute say the solution set

$$\mathbf{z}([t, t']; t_j, \mathbf{z}_j) = \{\mathbf{z}(\tau), \tau \in [t, t'] \subseteq [t_j, t_{j+1}]\},$$

which is the set of solutions of (3) at time τ , originating from each initial condition in \mathbf{z}_j at t_j , when τ is taken in a time interval $[t, t'] \subseteq [t_j, t_{j+1}]$. Formally speaking this is merely an inclusion function for $\varphi(\boldsymbol{\chi}_j, t_j, t, \tilde{\mathbf{z}}_j)$. We have

$$\boldsymbol{\chi}([t, t']; t_j, \boldsymbol{\chi}_j) = \varphi(\boldsymbol{\chi}_j, t_j, [t, t'], \tilde{\mathbf{z}}_j).$$

Lastly, an over-approximation of the set reachable over the time interval $[t_1, t_2]$ is merely given by $\boldsymbol{\chi}([t_1, t_2]; t_1, \boldsymbol{\chi}(t_1))$.

5. Computing hybrid transitions

This section gathers the main contribution of this paper. We use the tools introduced in Sect. 3 and 4 to solve flow/invariants and flow/guards intersections, jump transitions and reset functions.

5.1. Computing flow/invariants intersection

Here, we compute the intersection of the continuous flow with the invariant set at time t_j only, and we do not compute this intersection for all t in (t_j, t_{j+1}) .

At time t_j , denote $\mathbf{z}'_j = \text{inv}(q) \cap \mathbf{z}_j$. We have

$$z_j \in \mathbf{z}'_j \Leftrightarrow \text{inv}(q) \cap \mathbf{z}_j \Leftrightarrow (z_j \in \mathbf{z}_j) \wedge (\nu_q(z_j) < 0).$$

Now, since we are using (9) and the compound form (10), we have

$$z_j \in \text{inv}(q) \cap \mathbf{z}_j \Rightarrow \exists v_j \times r_j \in \mathbf{v}_j \times \mathbf{r}_j \quad \text{such that} \quad \nu_q(v_j + A_j r_j) < 0.$$

To obtain \mathbf{z}'_j , we have to solve the CSP

$$(\mathbf{v}_j \times \mathbf{r}_j, \nu_q(v_j + A_j r_j) < 0). \tag{11}$$

To do so using the methods introduced in section 3 in an effective way, we just need to compute the Jacobian of ν_q with respect to v_j and r_j . We have

$$\frac{\partial \nu_q}{\partial v_j} = \frac{\partial \nu_q}{\partial z} \quad \text{and} \quad \frac{\partial \nu_q}{\partial r_j} = \frac{\partial \nu_q}{\partial z} \cdot A_j$$

where $\partial \nu_q / \partial z$ is obtained using automatic differentiation. If $\mathbf{v}'_j \times \mathbf{r}'_j$ is the set of solutions to CSP (11), the solution set $\mathbf{z}'_j = \text{inv}(q) \cap \mathbf{z}_j$ is given by

$$\mathbf{z}'_j = \{v_j + A_j r_j, \mid v_j \in \mathbf{v}'_j, r_j \in \mathbf{r}'_j\}.$$

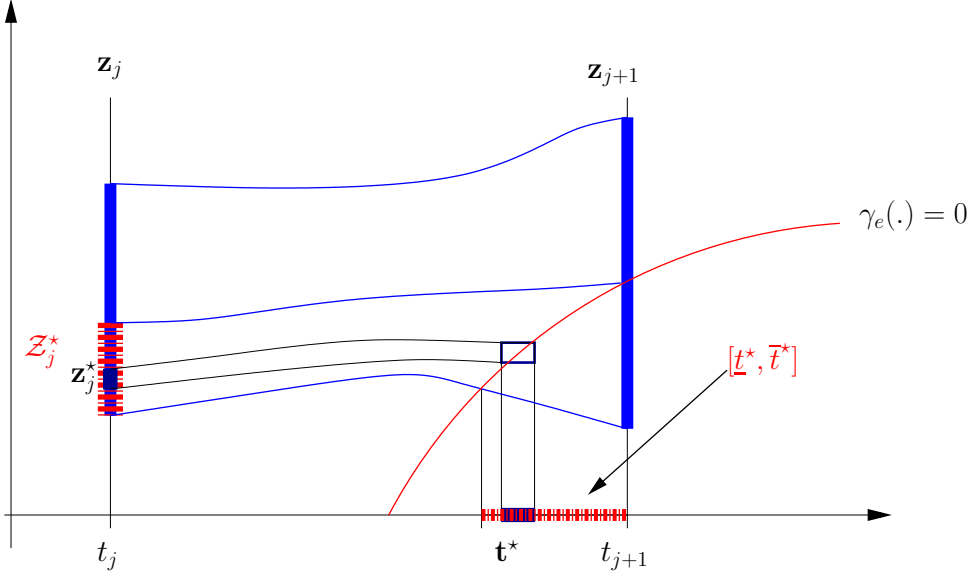


Figure 2: Solving flow/guard intersection. Our aim is to find the set of all initial conditions z_j^* that lead to a flow $z(t^*; t_j, z_j^*)$ which intersects the guard set at time t^* . Given a subset \mathbf{z}_j^* , this leads to a continuum of time instants \mathbf{t}^* . The combination of all possible $z_j^* \times t^*$ is the set $[\underline{t}^*, \overline{t}^*] \times \mathcal{Z}_j^*$.

5.2. Computing flow/guards intersection

The jump transition $e = q \rightarrow q'$ occurs at time t_e , when continuous flow reaches the guards set, that is, when the continuous state $z(t)$ satisfies condition (5) at t_e . Since we are dealing with flow pipes of non-zero size, there will be a continuum of time instants, where such intersection may occur. Hence, our aim is to find all the couples $t_e \times z(t_e)$ that satisfy (5). Now, since an analytical expression for an over-approximation of the boundaries of the flow pipes is given by algorithm φ over the time interval $[t_j, t_{j+1}]$, we just need to find all the initial state vectors z_j^* that lead to a $z(t_e; t_j, z_j^*)$ that satisfies (5) at $t_e = t^*$, that is, the set $[\underline{t}^*, \overline{t}^*] \times \mathcal{Z}_j^*$ depicted on Fig. 2. Therefore, the main idea is to filter the domain for initial state vector, namely \mathbf{z}_j , by pruning all the parts that lead to a flow that does not intersect the guards sets for all $t \in [t_j, t_{j+1}]$. We will show now which CSP to solve with respect to t_e and \mathbf{z}_j .

Using the analytical expression for the boundaries of the flow pipes as given by algorithm φ and considering solution sets \mathbf{z}_j and (5), the jump transition $e = q \rightarrow q'$ occurs at time t_e if

$$\exists t_e \times z(t_e) \in [t_j, t_{j+1}] \times \mathbf{z}(t_e, t_j, z_j) \quad \text{such that} \quad \gamma_e(\mathbf{z}(t_e)) = 0,$$

which leads to

$$\exists t_e \times z_j \in [t_j, t_{j+1}] \times \mathbf{z}_j \quad \text{such that} \quad \gamma_e(\varphi(z_j, t_j, t_e, \tilde{\mathbf{z}}_j)) = 0. \quad (12)$$

Now, recall that algorithm φ uses the compound form (10), where solution sets are characterized using form (9). Since the size of box \mathbf{v}_j depends mainly on $\mathbf{f}_q^{[k]}(\tilde{\mathbf{z}}_j)$ (see algorithm φ)

and is generally small, we let aside \mathbf{v}_j by keeping it fixed. Hence, solving (12) with respect to $t_e \times r_j$ instead of $t_e \times z_j$, (12) becomes

$$\exists t_e \times r_j \in [t_j, t_{j+1}] \times \mathbf{r}_j \quad \text{such that} \quad \gamma_e(\varphi(\mathbf{v}_j + A_j r_j, t_j, t_e, \tilde{\mathbf{z}}_j) = 0). \quad (13)$$

Solving (13) boils down to finding the solution set \mathcal{S}_{γ_e} of the CSP

$$([t_j, t_j + 1] \times \mathbf{r}_j, ' \gamma_e(\varphi(\chi_j(\cdot), \cdot, t, \cdot)) = 0 '). \quad (14)$$

Remark 3. Note that since we let aside \mathbf{v}_j , hence $\tilde{\mathbf{z}}_j$, $\varphi(z_j, t_j, t_e, \tilde{\mathbf{z}}_j)$ will have non-zero width. This will introduce overapproximation when characterizing \mathcal{S}_{γ_e} . Nevertheless, the latter should remain small as it depends mainly on the size of $\mathbf{f}_q^{[k]}(\tilde{\mathbf{z}}_j)$, that is, the enclosure of the truncation error, which is usually small.

To do so using tools introduced in section 3 in an effective way, we need to derive the Jacobian of γ_e with respect to t and r_j . It is easy to see that

$$\frac{\partial \gamma_e}{\partial t} = \frac{\partial \gamma_e}{\partial z} \cdot \frac{\partial z}{\partial t} = \frac{\partial \gamma_e}{\partial z} \cdot f_q$$

Now according to (8), we have

$$\frac{\partial z}{\partial z_j} = I + \sum_{i=1}^{k-1} (t - t_j)^i \frac{\partial \mathbf{f}_q^{[i]}}{\partial z}(z_j) = S,$$

which is computed at step 2 of algorithm φ , and according to (9) we have

$$\frac{\partial \gamma_e}{\partial r_j} = \frac{\partial \gamma_e}{\partial z} \cdot \frac{\partial z}{\partial z_j} \cdot \frac{\partial z_j}{\partial r_j} = \frac{\partial \gamma_e}{\partial z} \cdot S \cdot A_j.$$

Here we use automatic differentiation to obtain $\partial \gamma_e / \partial z$.

Denote by $[\underline{t}^*, \bar{t}^*] \times \mathbf{r}_j^*$ the enclosure of the set \mathcal{S}_{γ_e} , solution set for CSP (14). We can summarize our main result in the following proposition.

Proposition 1 (Flow/guards set intersection).

- If set $[\underline{t}^*, \bar{t}^*] \times \mathbf{r}_j^*$, solution set for CSP (14), is empty, then the flow does not intersect the guards set for all $t \in [t_j, t_{j+1}]$.
- If there exists $t_e \in [t_j, t_{j+1}]$ at which the flow intersects the guards set, then $t_e \in [\underline{t}^*, \bar{t}^*]$, and there exists r_j taken in \mathbf{r}_j^* such that if the flow starts with r_j at t_j , it will intersect the guards sets at t_e .

5.3. Computing discrete transition

If the continuous flow intersects the guards set at time t_e , a discrete transition $e = q \rightarrow q'$ occurs and $\chi(t_e^+) = \rho_e(\chi(t_e^-))$. Now, since we have a conservative approximation, namely $[\underline{t}^*, \bar{t}^*]$ of the continuum of time instants, where switching occurs, we need a relaxation for computing reset functions. We will use the following relaxation.

Proposition 2 (Relaxed reset function).

If set $[\underline{t}^, \bar{t}^*] \times \mathbf{r}_j^*$ is not empty, we say that the event $e = q \rightarrow q'$ occurs at $t_e = \underline{t}^*$ and assume that $\chi(t_e^-) = \chi([\underline{t}^*, \bar{t}^*])$.*

Nonetheless, we keep track of the actual switching points $t_e \in [\underline{t}^*, \bar{t}^*]$, because the time interval $[\underline{t}^*, \bar{t}^*]$ is stored in $\mathbf{z}(t_e^+)$. Indeed, recall that $\mathbf{z} = (x, p, t)$. In order to perform an accurate estimation of the switching time, hence performing an accurate event localization, one may need to control the overestimation that may be introduced by this relaxation. This can be done by analyzing the size of either time interval $[\underline{t}^*, \bar{t}^*]$ or box \mathbf{z}^* , as obtained via $\chi^* = \varphi_q(\chi_j^*, t_j, [\underline{t}^*, \bar{t}^*], \tilde{\mathbf{z}})$. If one of them is too large, we may use bisection. Since we are performing flow transition using the compound form (10), state partitioning must be done with care. It is done as follows

Algorithm Partition(input : χ ; output : ${}^L\chi, {}^R\chi$)

1. split $\mathbf{z} \rightarrow {}^L\mathbf{z}, {}^R\mathbf{z}$,
2. solve ${}^R\mathbf{r} = \{r \in \mathbf{r} \mid \exists z \times v \in {}^R\mathbf{z} \times \mathbf{v}, z = v + Ar\}$
3. solve ${}^L\mathbf{r} = \{r \in \mathbf{r} \mid \exists z \times v \in {}^L\mathbf{z} \times \mathbf{v}, z = v + Ar\}$
4. ${}^R\chi = \{{}^R\mathbf{z}, \hat{z}, \mathbf{v}, {}^R\mathbf{r}, A\}$
5. ${}^L\chi = \{{}^L\mathbf{z}, \hat{z}, \mathbf{v}, {}^L\mathbf{r}, A\}$

For solving the linear systems at step 2 and 3, we use the interval version of Gauss elimination [27].

Remark 4. In this paper we consider uncertain, but simple reset functions $\rho_e(z(t_e^+)) = \beta + \text{diag}(\alpha)z(t_e^-)$, where $\text{diag}(\alpha)$ is a diagonal matrix, $\alpha \in \boldsymbol{\alpha}$, and $\beta \in \boldsymbol{\beta}$. The case of nonlinear and complex reset functions can be addressed easily using the several techniques developed in the interval analysis literature for computing accurately the image of a set by a nonlinear function (see e.g. algorithm `ImageSP` in [27]).

Finally, in order to obtain the compound form $\chi(t_e^+)$ we need to find sets \mathbf{v}' , and \mathbf{r}' and matrix A' such that $\mathbf{z}(t_e^+) = \{v' + A'r' \mid v' \in \mathbf{v}', r' \in \mathbf{r}'\}$. We have

$$\rho_e(z(t_e^+)) = \beta + \text{diag}(\alpha)z(t_e^-) \Rightarrow (A' = A) \wedge (\mathbf{v}' = \beta + \text{diag}(\alpha)\mathbf{v}) \wedge (\mathbf{r}' = \text{diag}(\alpha)\mathbf{r}).$$

5.4. Hybrid transition

We can now summarize the previous methods in algorithm **Hybrid-Transition** given in Table 1. This algorithm addresses both continuous and discrete transitions. Its main structure is similar to the one suggested in [21]. In this version, integration time step is fixed. Algorithm handles lists of triples $(q, t, \chi(t))$ composed of the discrete location, a clock time and the compound form (10) for continuous state solution set at time t . The lists are as follows:

1. Reachable state list \mathcal{L}_j^R is the set reached at time t_j . It contains enclosures of the continuous flows for all modes, as obtained via algorithm φ , when time sweeps from t_0 to t_j .
2. Frontier list \mathcal{L}_j^F is the frontier reached at time t_j . It contains the enclosures of the continuous flows as obtained via algorithm φ at time t_j , which intersect the invariant sets. Note that the compound form χ is used for characterizing solution sets.
3. Running jump list \mathcal{L}_e is a running list for solving discrete transition. As for the two lists above, it contains the discrete location q , an initial time t_0 and the continuous state solution set at time t_0 , but it contains also a final time t_1 . This extra data is needed since we are analyzing the occurrence of a discrete transition over the time interval $[t_0, t_1]$ and are planning to partition time interval as indicated in section 5.3.
4. Running frontier list \mathcal{L} , which is needed to proceed with flow transition with mode q' from $[t_e, t_{j+1}]$, when a discrete transition $e = q \rightarrow q'$ occurs at $t_e \in [t_j, t_{j+1}]$.

Algorithm **Hybrid-Transition** proceeds with continuous transition over integration time step $[t_j, t_{j+1}]$ (steps 5 to 9) for all boxes in the running frontier list. It computes whole flow enclosure at step 5 and updates reachable state list at step 6. It computes solution set at step 7 and computes its intersection with the invariant sets at step 8. If not empty, the filtered solution set is stored in frontier list (step 9).

Algorithm **Hybrid-Transition** then proceeds with discrete transition, which may occur over integration time step $[t_j, t_{j+1}]$. For all boxes in the running jump list, it checks whether a discrete transition exists (step 15) and solves the CSP at step 16. If continuous flow intersects guards sets (step 17) then discrete transition is done only if the size of $[\underline{t}^*, \bar{t}^*]$ is smaller than a given threshold (step 18). In this case, the flow enclosure over $[\underline{t}^*, \bar{t}^*]$, namely χ^* is computed as obtained with χ_0^* as initial state vector at time t_0 (step 18). If this solution set is smaller than a given threshold (step 19), then algorithm proceeds with the discrete transition, the running frontier list is updated (step 20), and eventually a further continuous transition will occur from \underline{t}^* to t_{j+1} (step 3). If $\tilde{\chi}^*$ is too large, then a solution set is computed at time \underline{t}^* and is partitioned (step 23), and the two sub-boxes are used to update the running jump list. Note here that initial and final times are also updated. Now, if the size of $[\underline{t}^*, \bar{t}^*]$ is too large, then this time interval is bisected and the running jump list is updated. Here again note that initial and final times are adapted (step 28-31).

5.5. Comments on Complexity

The computational complexity of our approach to hybrid reachability is mainly driven by the one of solving the numerical CSPs within Algorithm **Hybrid-Transition**. The un-

derlying Taylor series method is of polynomial complexity. The work per step is $O(k^2)$ to compute Taylor coefficients, and $O(n^3)$ for performing the linear algebra. However, it is generally admitted that solving CSPs, either on discrete or on continuous domains, is in theory NP-hard. There have been efforts though to develop solving techniques, whose practical time complexity is better than the exponential worst case. Techniques for solving CSP on discrete domains were successfully used to solve non-trivial issues in several domains such as operations research, scheduling and planning, vehicle routing, component configuration, networks, and bioinformatics [41]. The case of numerical CSP on continuous domains is much harder, because solving techniques rely on exhaustive search techniques that compute solution space coverings by means of multi-dimensional boxes. Nevertheless, several solving techniques were developed by adapting discrete constraint satisfaction techniques to continuous problems. Here again, solving techniques for NCSP were developed and successfully used to solve non trivial problems in control, robotics, and design [27, 42]. Finally, though solving NCSP is NP-hard in theory for the general case, the actual time complexity encountered in practice is generally tractable.

6. Examples

6.1. Example 1

Consider the hybrid transition for a hybrid dynamical system with two modes $q = 1, 2$ and one jump transition $e = 1 \rightarrow 2$ given by

$$\left\{ \begin{array}{l} \text{flow}(1) : f_1(x_1, x_2) = (x_2, -px_2 - g \sin(x_1)) \\ \text{inv}(1) : \nu_1(x_1, x_2) = \cos(x_1) - x_2/10 - 0.7 \\ \text{flow}(2) : f_2(x_1, x_2) = (x_2, -3px_2 - g \sin(x_1)) \\ \text{inv}(2) : \nu_2(x_1, x_2) = -\nu_1(x_1, x_2) \\ \text{guard}(1) : \gamma_1(x_1, x_2) = \nu_1(x_1, x_2) \\ \text{reset}(1) : \rho_1(x_1, x_2) = (\alpha_1 x_1, \alpha_2 x_2) \end{array} \right. \quad (15)$$

with $\alpha_1 = -1$, $\alpha_2 \in [-2.05, -2]$, $g = 10$, $p \in [6, 6.3]$ and $x_0 \in [-0.9, -0.8] \times [3, 3.5]$.

In both Fig. 3 and Fig. 4, we display the set reachable as given by lists \mathcal{L}_j^R ; the frontier sets are given by lists \mathcal{L}_j^F , for hybrid system (15), when time sweeps from $t_0=0$ s to $t_f=0.3$ s. Fig. 3 displays the reachable set in $x_1 \times x_2$ -space, and Fig. 4 displays the reachable set in $t \times x_2$ -space. The top plots in both figures correspond to the reachable set given by the approach that uses a priori solutions only as in [21]. The computations are done in less than 0.1s CPU time (PIV 2GHz). The bottom plots correspond to the reachable set as given by our algorithm `Hybrid-Transition`. They are obtained in 26s CPU time (here only the time interval $[\underline{t}^*, \bar{t}^*]$ was bisected until a threshold $\epsilon_T = 0.005$ s (steps 28-31 of algorithm `Hybrid-Transition`), while the integration time step was taken constant $t_{j+1} - t_j = 0.05$ s). One can see how our algorithm partitions continuous space to proceed with discrete transition from mode 1 to 2. Then, flow transitions occur in mode 2 with the new list of boxes. One can also notice that for frontier boxes, that is, the solution set for (15) in mode 2, time t is now an interval, since the actual time instant t_e when the transition occurs is not known, but enclosed in an interval, according to proposition 2.

It is clear from these plots that our approach yields better results than [21]; that is, more accurate reachable and frontier sets, but with increased computational cost. When bisection is performed on the time step only, computation time remains satisfactory as we found that reducing by half ϵ_T doubles CPU time. To the contrary, and as expected, when the search domain for the continuous state vector is also bisected, then the computation time grows significantly. Future efforts should focus on using more efficient narrowing procedures for state vector domains without using bisection. Also finding a way to merge the obtained list of boxes should also reduce computation time.

6.2. Example 2

Let us consider as second example an artificial hybrid system inspired from a three state biochemical reactor, with two modes $q = 1, 2$ and one jump transition $e = 1 \rightarrow 2$:

$$\left\{ \begin{array}{l} \text{flow}(1) : f_1(x_1, x_2, x_3) = (g_1(x_1, x_2, x_3), g_2(x_1, x_2, x_3), g_3(x_1, x_2, x_3)) \\ \text{inv}(1) : \nu_1(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2 - 30^2 \\ \text{flow}(2) : f_2(x_1, x_2, x_3) = f_1(x_1, x_2, x_3) \\ \text{inv}(2) : \nu_2(x_1, x_2, x_3) = -\nu_1(x_1, x_2, x_3) \\ \text{guard}(1) : \gamma_1(x_1, x_2, x_3) = \nu_1(x_1, x_2, x_3) \\ \text{reset}(1) : \rho_1(x_1, x_2, x_3) = (x_1 + 60, x_2 + 15, x_3 + 30) \end{array} \right. \quad (16)$$

where

$$\begin{aligned} g_1(x_1, x_2, x_3) &= (-0.4 + \mu(x_2, x_3))x_1, \\ g_2(x_1, x_2, x_3) &= -0.4(x_2 + x_f) - (\mu(x_2, x_3)x_1)/(0.4) \\ g_3(x_1, x_2, x_3) &= (0.2 + 2.2\mu(x_2, x_3))x_1 - 0.4x_3 \\ \mu(x_2, x_3) &= (\mu_m x_2(1 - x_3/50))/(k_s + x_2). \end{aligned}$$

Initial domains for state variables are taken as follows: $x_1(t_0) \in [58.5, 71.5]$, $x_2(t_0) \in [47.5, 52.5]$, and $x_3(t_0) \in [142.5, 157.5]$. Uncertainty domain for parameters are $\mu_m \in [0.396, 0.404]$, $k_s \in [1.485, 1.515]$ and $x_f \in [19.8, 20.2]$.

As for the previous example, we compare the outcome of the method in [21] and our **Hybrid-Transition** algorithm. Fig. 5 displays the projection of the reachable and frontier sets in the $x_1 \times x_2$ -space (classical approach: up, our approach: down) and Fig. 6 displays the projection of the reachable and frontier sets in the $t \times x_2$ -space. Though the method in [21] requires less than 0.1s, our method requires 30s CPU time. We used bisection on time step only until a threshold $\epsilon_T = 0.005s$, while the integration time step was taken constant $t_{j+1} - t_j = 0.05s$.

When used with a continuous state vector of dimension 3 with three uncertain parameters, previous conclusions hold, i.e. our method derives more accurate results at the price of a satisfactory computation time, as long as only time is partitioned.

7. Conclusion

In this paper we have addressed the issues of solving the intersection of a nonlinear flow with invariants and guards sets. These issues are at the core of event detection and

localization problems when computing reachable sets with hybrid systems. We have shown that it is possible to derive analytical expression for the boundaries of continuous flows using interval Taylor methods and techniques for controlling the wrapping effect. As a consequence, the problem of flow/sets intersection boils down to solving a numerical constraint satisfaction problem.

We have shown that the latter can be done in an efficient and accurate way using interval methods. However, the complexity and computation time are also increased. In fact, a trade-off between accuracy and computation time can be achieved by an appropriate tuning of algorithm `Hybrid-Transition`. For instance, when the thresholds are chosen very loose, this algorithm boils down to the method originally introduced in [21], which requires small computation time, but achieves only an approximate event localization. Obviously, accurate event localization requires larger computation time.

It remains to study further the complexity of the proposed algorithm, which can be improved by using more efficient constraint satisfaction techniques that do not rely on variable partition.

A drawback of the method introduced in this paper is that we have addressed jump transition in a very simple manner. We need to partition continuous state vector to proceed with discrete transitions, hence we generate a list of boxes, which may provoke a large number of flow transitions starting from every box in the list. Future work will investigate ways to merge these boxes without introducing overestimation. Moreover, we will also consider nonlinear reset functions.

References

- [1] N. Ramdani and N.S. Nedialkov. Computing reachable sets for uncertain nonlinear hybrid systems using interval constraint propagation techniques. In *3rd IFAC ADHS 09 Analysis and Design of Hybrid Systems, September 16-18, 2009, Zaragoza, Spain*, pages 156–161, 2009.
- [2] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.
- [3] E. Asarin, O. Maler, and A. Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138:35–65, 1995.
- [4] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In *HSCC00, vol. 1790 in LNCS*, pages 20–31, 2000.
- [5] O. Botchkarev and S. Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In *HSCC, vol. 1790 in LNCS*, pages 73–88. Springer-Verlag, 2000.
- [6] C. J. Tomlin, I. M. Mitchell, A. M. Bayen, and M. Oishi. Computational techniques for the verification of hybrid systems. *Proceedings of the IEEE*, 91(7):986–1001, 2003.
- [7] A. B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for hybrid dynamics: the reachability problem. In W.P. Dayawansa, A. Lindquist, and Y. Zhou, editors, *New Directions and Applications in Control Theory, Lecture Notes in Control and Information Sciences*, volume 321, pages 193–205. Springer-Verlag, 2005.
- [8] A. Girard. Reachability of uncertain linear systems using zonotopes. In *HSCC, vol 3414 in LNCS*, pages 291–305, 2005.
- [9] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. A time-dependant Hamilton-Jacobi formulation of reachable sets for continuous dynamics games. *IEEE Trans. Automatic Control*, 50(7):947–957, 2005.
- [10] A. Tiwari and G. Khanna. Series abstractions for hybrid automata. In *HSCC, vol 2289 in LNCS*, pages 465–478, 2002.
- [11] O. Stursberg and B.H. Krogh. Efficient representation and computation of reachable sets for hybrid systems. In *HSCC, vol. 2623 in LNCS*, pages 482–497. Springer-Verlag, 2003.
- [12] H. Guéguen and J. Zaytoon. On the formal verification of hybrid systems. *Control Engineering Practice*, 12:1253–1267, 2004.

- [13] L. Doyen, T.A. Henzinger, and J.F. Raskin. Automatic rectangular refinement of affine hybrid systems. In *FORMATS'05, vol. 3829 in LNCS*, pages 144–161, 2005.
- [14] M.-A. Lefebvre and H. Guéguen. Hybrid abstractions of affine systems. *Nonlinear Analysis*, 65(6):1150–1167, 2006.
- [15] M. Kloetzer and C. Belta. Reachability analysis of multi-affine systems. In *HSCC, vol. 3927 in LNCS*, pages 348–362. Springer-Verlag, 2006.
- [16] G. Batt, C. Belta, and R. Weiss. Model checking genetic regulatory networks with parameter uncertainty. In *HSCC, vol. 4416 in LNCS*, pages 61–75. Springer-Verlag, 2007.
- [17] E. Asarin, T. Dang, and A. Girard. Hybridization methods for the analysis of non-linear systems. *Acta Informatica*, 43:451–476, 2007.
- [18] A. Chutinan and B. H. Krogh. Computational techniques for hybrid systems verification. *IEEE T. Automatic Control*, 48(1):64–75, 2003.
- [19] N.S. Nedialkov, K.R. Jackson, and G.F. Corliss. Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*, 105:21–68, 1999.
- [20] N.S. Nedialkov and M. von Mohrenschildt. Rigorous simulation of hybrid dynamic systems with symbolic and interval methods. In *Proceedings of the American Control Conference*, volume 1, pages 140–147, 2002.
- [21] T.A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi. Beyond HYTECH: Hybrid systems analysis using interval numerical methods. In *HSCC, vol. 1790 in LNCS*, pages 130–144, 2000.
- [22] N. Ramdani, N. Meslem, and Y. Candau. Reachability of uncertain nonlinear systems using a nonlinear hybridization. In *HSCC, vol. 4981 in LNCS*, pages 415–428. Springer-Verlag, 2008.
- [23] T. Park and P. Barton. State event location in differential-algebraic models. *ACM transactions on modeling and computer simulation*, 6(2):137–165, 1996.
- [24] J. M. Esposito, V. Kumar, and G. J. Pappas. Accurate event detection for simulating hybrid systems. In *HSCC, vol. 2034 in LNCS*, pages 204–217. Springer-Verlag, 2001.
- [25] E. Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32:281–331, 1987.
- [26] J. C. Cleary. Logical arithmetic. *Future Computing Systems*, 2:125–149, 1987.

- [27] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis: with examples in parameter and state estimation, robust control and robotics*. Springer-Verlag, London, 2001.
- [28] S. Ratschan and Z. She. Safety verification of hybrid systems by constraint propagation-based abstraction refinement. *ACM Trans. Embedded Computing Systems*, 6(1), 2007.
- [29] A. Eggers, M. Fränzle, and C. Herde. SAT modulo ODE: A direct SAT approach to hybrid systems. In *Cha et al. (Eds.): ATVA 2008, LNCS 5311*, pages 171–185, 2008.
- [30] E. Hansen and G.W. Walster. *Global optimization using interval analysis*. Marcel Dekker, 2nd edition, 2004.
- [31] H. Collavizza, F. Delobel, and M. Rueher. Comparing partial consistencies. *Reliable computing*, 5:213–228, 1999.
- [32] A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. In A. Iserles, editor, *Acta Numerica*, chapter 4. Cambridge University Press, 2004.
- [33] D. L. Waltz. *Generating semantic descriptions from drawings of scenes with shadows*, pages 19–91. McGraw-Hill, New York, 1975.
- [34] <http://www-sop.inria.fr/coprin/logiciels/alias/>.
- [35] <http://www2.imm.dtu.dk/~km/fadbad/>.
- [36] <http://www.ti3.tu-harburg.de/software/profilenglisch.html>.
- [37] R.E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, 1966.
- [38] G.F. Corliss and R. Rihm. Validating an a priori enclosure using high-order Taylor Series. In G. Alefeld and A. Formmer, editors, *Scientific computing, Computer Arithmetic, and Validated Numerics*, pages 228–238. Akademie Verlag, Berlin, 1996.
- [39] N.S. Nedialkov, K. Jackson, and J. Pryce. An effective high-order interval method for validating existence and uniqueness of the solution of an IVP for an ODE. *Reliable Computing*, 7(6):449–465, 2001.
- [40] M. Berz and G. Hoffstätter. Computation and application of taylor polynomials with interval remainder bounds. *Reliable Computing*, 4(1):83–97, 1998.
- [41] F. Rossi, P. van Beek, and T. Walsh, editors. *Handbook of Constraint Programming*. Elsevier, 2006.
- [42] F. Benhamou and L. Granvilliers. Continuous and interval constraints. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, chapter 16, pages 571–603. Elsevier, 2006.

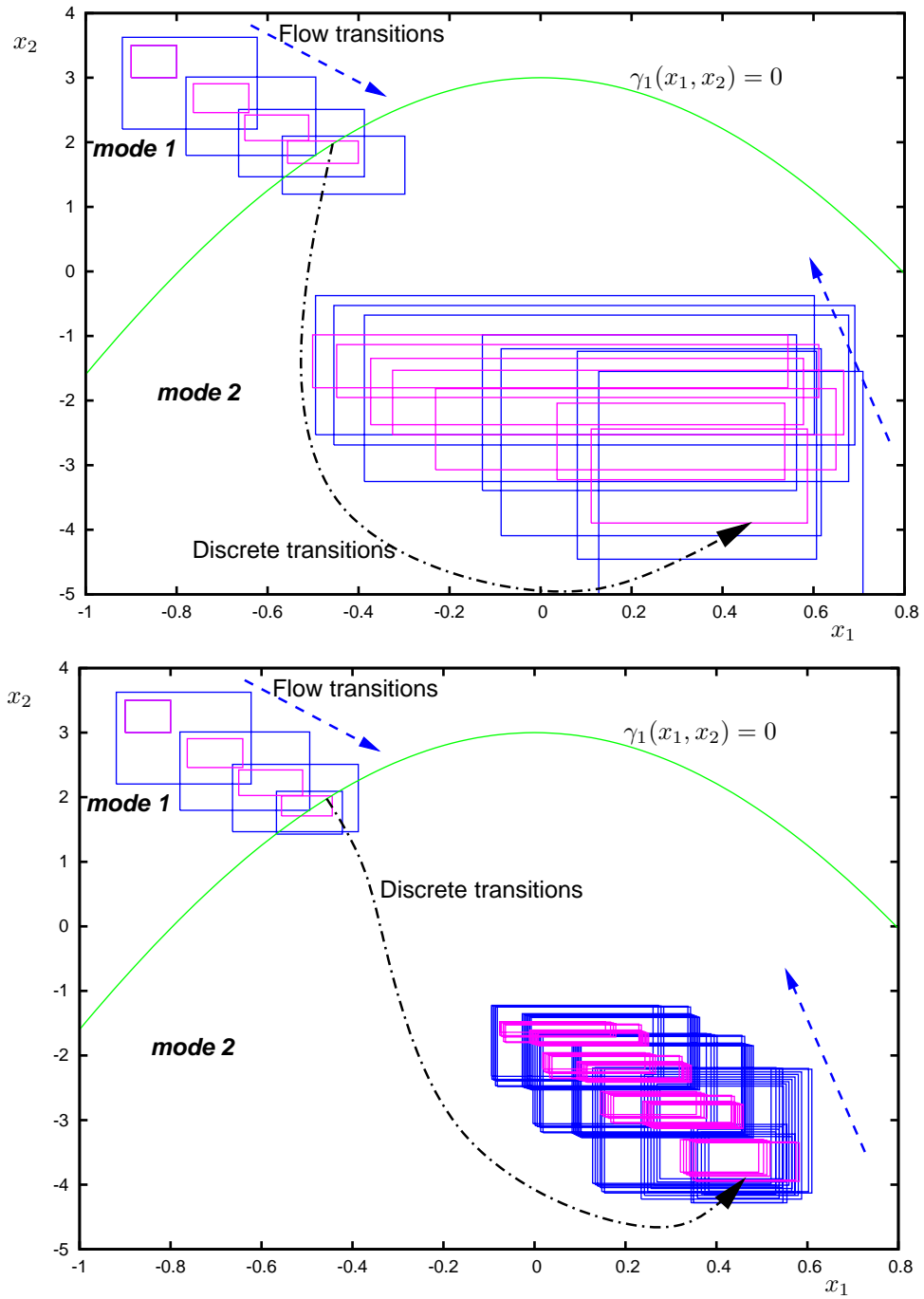


Figure 3: Reachable set of (15) in the $x_1 \times x_2$ -space: [21] approach using a priori solution only (CPU time $< 0.1s$) (top) our method (CPU time 26s) (bottom)

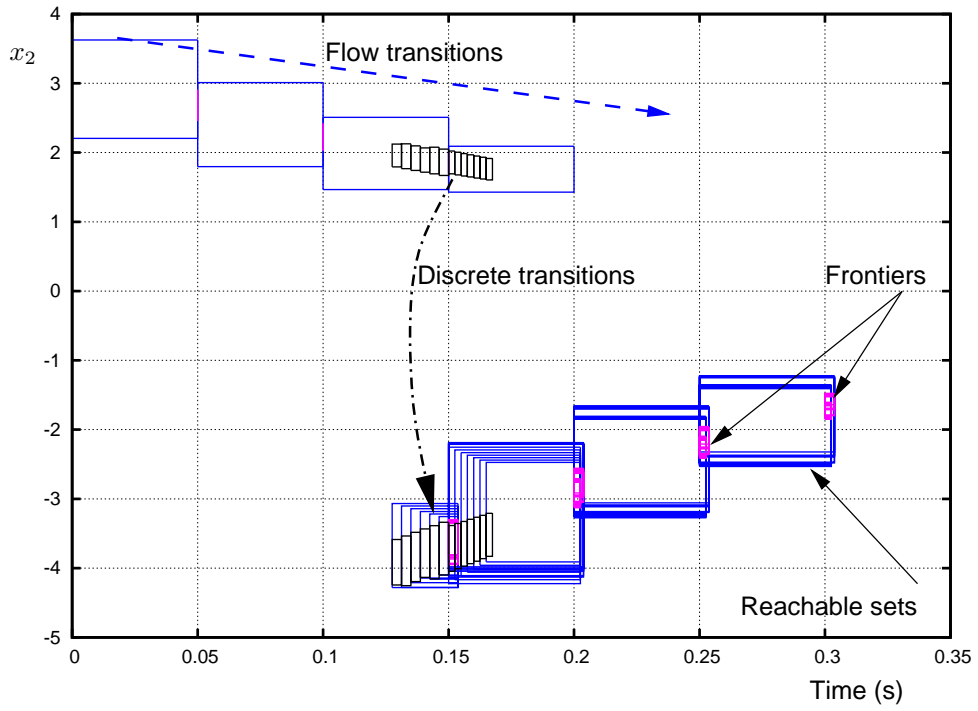
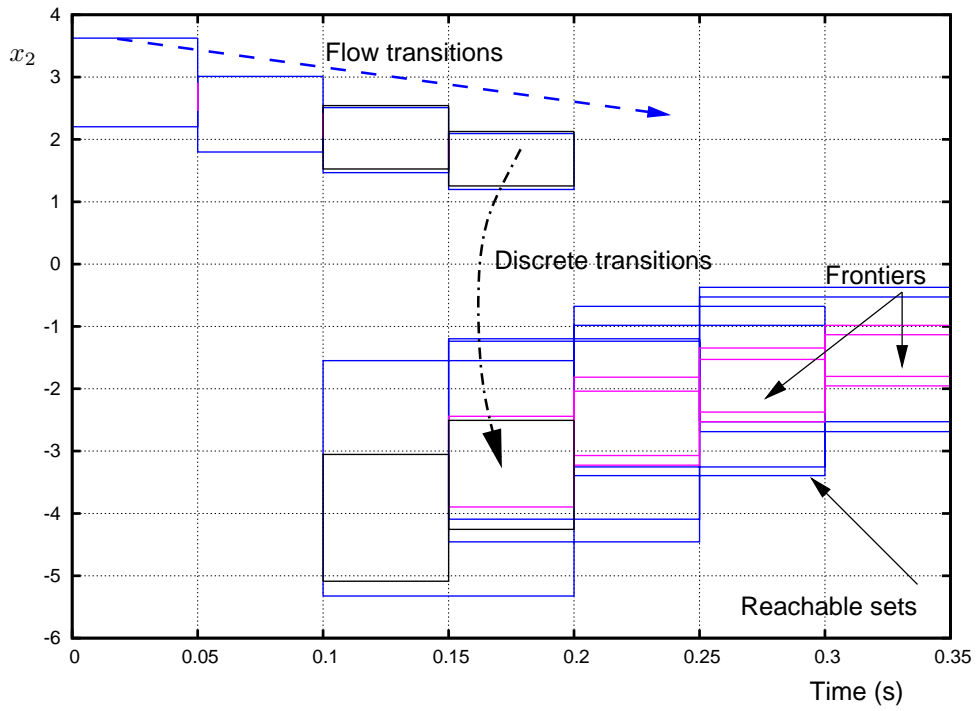


Figure 4: Reachable set of (15) in the $t \times x_2$ -space: [21] approach using a priori solution only (top) our method (bottom)

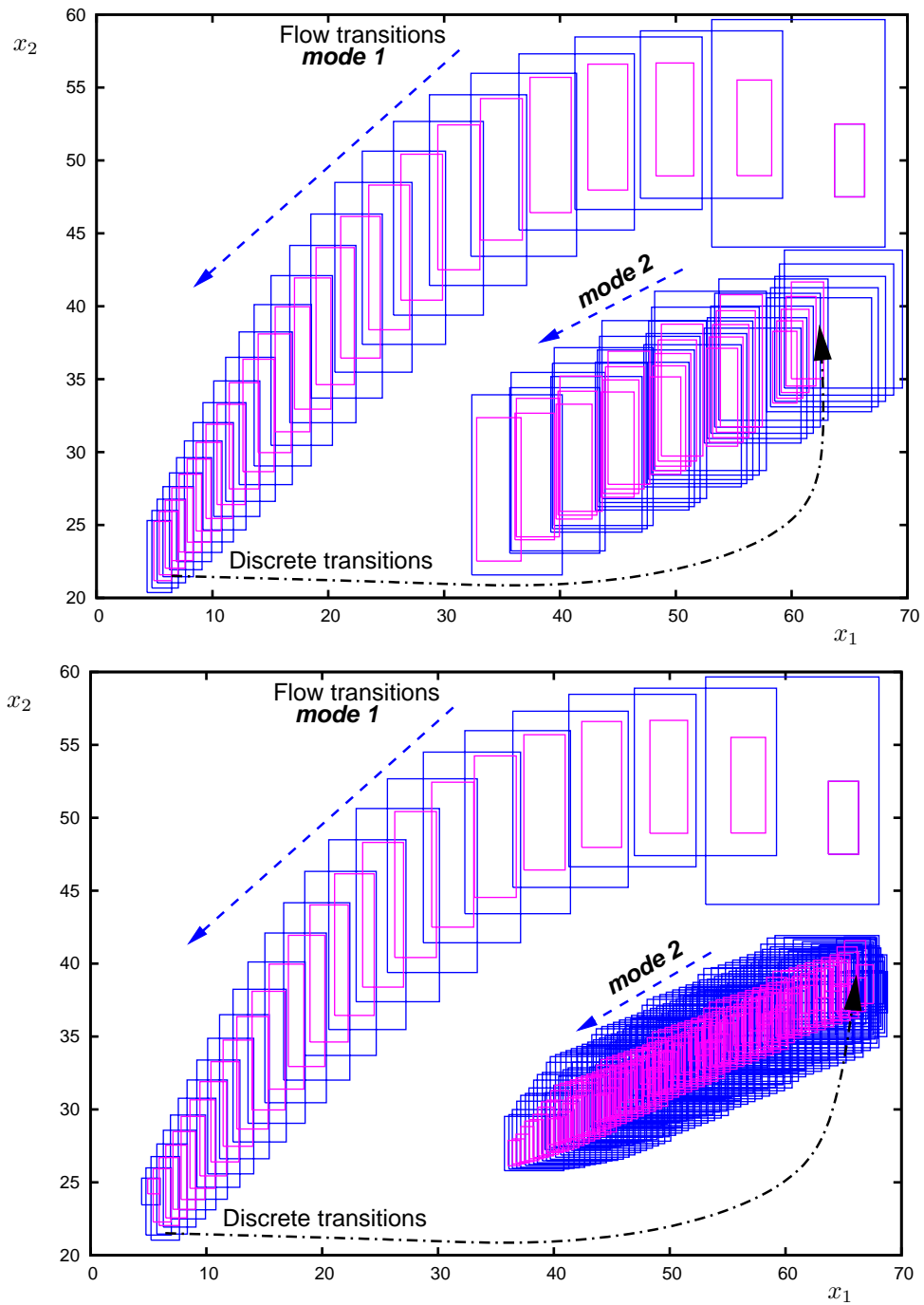


Figure 5: Reachable set of (16) in the $x_1 \times x_2$ -space: [21] approach using a priori solution only (CPU time $< 0.1s$) (up) our method (CPU time 30s)(down)

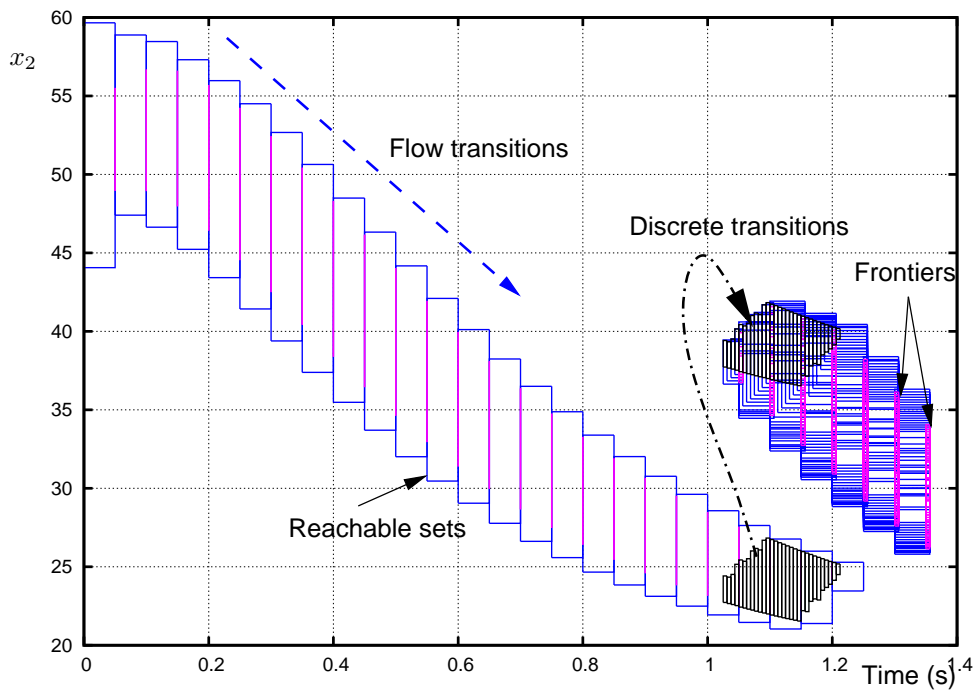
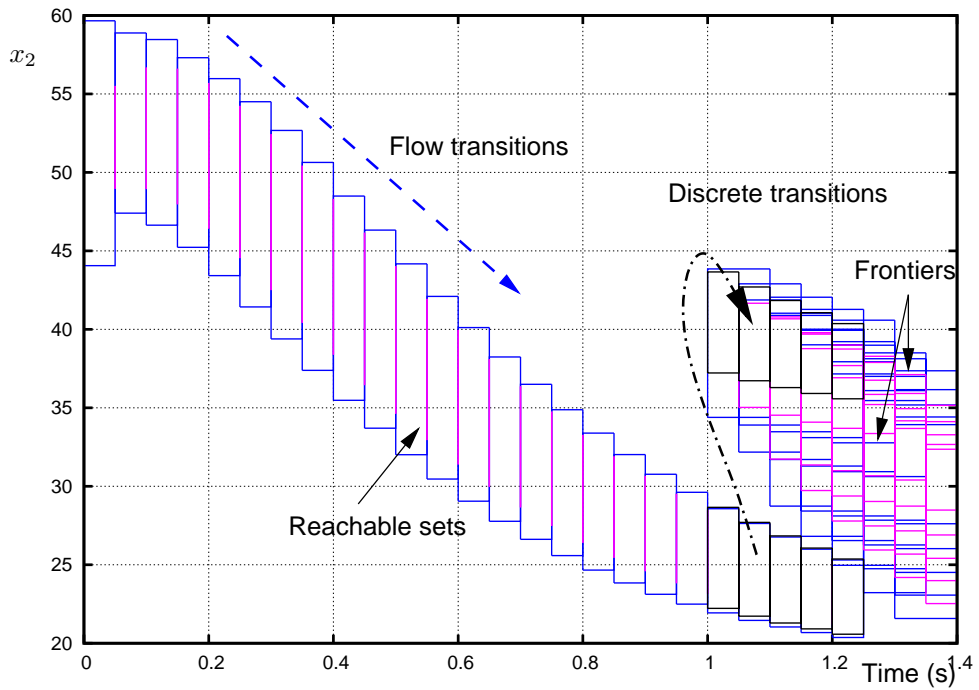


Figure 6: Reachable set of (16) in the $t \times x_2$ -space: [21] approach using a priori solution only (up) our method (down)

Table 1: Algorithm Hybrid-Transition

(input : $\mathcal{L}_j^F, t_{j+1}, \{\varphi_q(), \tilde{\varphi}_q(), \nu_q()\}_{q \in \mathcal{Q}}, \{\gamma_e(), \rho_e()\}_{e \in \mathcal{E}}, \epsilon_T, \epsilon_z$; output : $\mathcal{L}_{j+1}^F, \mathcal{L}_{j+1}^R$)

1. *Initialization* : initialize running frontier list $\mathcal{L} := \mathcal{L}_j^F$;
2. while running frontier list \mathcal{L} not empty, do
3. pick up \mathcal{L} list element $(q, t_0, \boldsymbol{\chi}_0)$;
4. *Continuous transition*
5. compute continuous expansion over $[t_0, t_{j+1}] \rightarrow \tilde{\boldsymbol{z}}_j$;
6. update reached set list $\mathcal{L}_{j+1}^R \leftarrow (q, t_0, t_{j+1}, \tilde{\boldsymbol{z}}_j)$;
7. compute new solution at time $t_{j+1} \rightarrow \boldsymbol{\chi}_{j+1}$;
8. solve CSP (11) to compute $\boldsymbol{\chi}'_{j+1} := \boldsymbol{\chi}_{j+1} \cap \text{inv}(q)$;
9. if $\boldsymbol{\chi}'_{j+1}$ not empty, then update frontier list $\mathcal{L}_{j+1}^F \leftarrow (q, t_{j+1}, \boldsymbol{\chi}'_{j+1})$;
10. *Discrete transition*
11. for all transitions e in \mathcal{E} do
12. initialize running jump list $\mathcal{L}_e := \{(q, t_0, \boldsymbol{\chi}_0, t_{j+1})\}$;
13. while jump list \mathcal{L}_e not empty, do
14. pick up \mathcal{L}_e list element $(q, t_0, \boldsymbol{\chi}_0, t_{j+1})$;
15. if transition $e = (q, q')$ exists, then
16. solve CSP (14) to compute $[\underline{t}^*, \bar{t}^*] \times \boldsymbol{\chi}_0^*$;
17. if $[\underline{t}^*, \bar{t}^*] \times \boldsymbol{\chi}_0^*$ not empty then
18. if $(\bar{t}^* - \underline{t}^*) \leq \epsilon_T$ then compute flow enclosure over $[\underline{t}^*, \bar{t}^*] \rightarrow \tilde{\boldsymbol{\chi}}^*$.
19. if $\text{width}(\tilde{\boldsymbol{\chi}}^*) \leq \epsilon_z$ then
20. jump and update running frontier list $\mathcal{L} \leftarrow (q', \underline{t}^*, \rho_e(\tilde{\boldsymbol{\chi}}^*))$;
21. else
22. compute solution set at $\underline{t}^* \rightarrow \boldsymbol{\chi}_1^*$;
23. partition $\boldsymbol{\chi}_1^* \rightarrow \{^r\boldsymbol{\chi}_1^*, ^l\boldsymbol{\chi}_1^*\}$;
24. update running jump list $\mathcal{L}_e \leftarrow (q, \underline{t}^*, ^r\boldsymbol{\chi}_1^*, t_{j+1})$;
25. update running jump list $\mathcal{L}_e \leftarrow (q, \underline{t}^*, ^l\boldsymbol{\chi}_1^*, t_{j+1})$;
26. end if;
27. else
28. compute solution set at $\underline{t}^* \rightarrow \boldsymbol{\chi}_1^*$;
29. compute solution set at $t_2 := (\underline{t}^* + \bar{t}^*)/2 \rightarrow \boldsymbol{\chi}_2^*$;
30. update running jump list $\mathcal{L}_e \leftarrow (q, \underline{t}^*, \boldsymbol{\chi}_1^*, t_2)$;
31. update running jump list $\mathcal{L}_e \leftarrow (q, t_2, \boldsymbol{\chi}_2^*, t_{j+1})$;
32. end if;
33. end if;
34. end if;
35. end while; % on \mathcal{L}_e
36. end for;
37. end while; % on \mathcal{L}