A fast method for solving guard set intersection in nonlinear hybrid reachability

Moussa Maïga, Nacim Ramdani and Louise Travé-Massuyès

Abstract— **Reachability computation formulates the problem of simulating the behavior of a continuous or hybrid dynamical system in a set-theoretical framework. Compared to the stochastic approach, it provides guaranteed results and has been shown highly valuable for verification or synthesis tasks. This issue is still quite challenging for uncertain nonlinear hybrid dynamical systems.**

Recently, [1] proposed a method for solving the flow/guard intersection problem that is at the core of hybrid reachability. It first derives an analytical expression for the boundaries of continuous flows using interval Taylor methods and techniques for controlling the wrapping effect. It then expresses the event detection and localization problem underlying flow/guard intersection as a constraint-satisfaction problem (CSP).

One of the main issues in interval integration is to control, at each step, the overestimation of the reachable state set due to the wrapping effect. For this purpose, [1] only relies on the geometrical transformation induced by Lohner's QR-factorization method [4], which acts at the integration step. But when dealing with hybrid systems, another source of overestimation exists at the transition step.

This paper describes an efficient method for solving flow/guard intersection: using the standard contractor *HC4Revise* at the transitions step, we will show how to minimize both the overestimation of the flow/guard intersection and the computational complexity, hence computation time. Interestingly, the geometrical transformation introduced by Lohner's QR-factorization method combined with our method, eventually minimizes the overestimation for the whole hybrid flow trajectory. The performance of the new method is illustrated on examples involving typical hybrid systems.

I. INTRODUCTION

The analysis of reachable sets of discrete, continuous and hybrid systems i.e., systems exhibiting both discrete and continuous dynamics plays a fundamental role in many important engineering problems, such as the verification of the correctness and safety of embedded controllers for aircraft, automobiles, medical devices, energy production and distribution plants, and other safety-critical applications [3]. They are often components of safety-critical systems, it is then necessary to have a thorough and guaranteed insight on the properties of the system, such as performance, safety or stability.

Reachability analysis consists in computing the set of states reachable by a system, so it is practically impossible to enumerate all the possible behaviors of a system. A key issue when computing the reachable set of a hybrid dynamical system lays in the calculation of the continuous reachable space for each mode, which boils down to the computation of reachable space for uncertain continuous dynamical systems. Furthermore, in order to compute the reachable set in a given mode, it is necessary to compute a conservative over-approximation of the interesection of the reachable set with a guard condition that can be described by a set.

Using support functions, [5] propose an improvement of the over-approximation error of the image computation of discrete transitions (jumps). The critical operation of this image computation is the intersection of the flowpipe with the guard sets of the transitions, since intersection is in general a difficult operation. Then they propose an approach for computing the intersection of the flowpipe with polyhedral guards.

More recently, [6] introduced a new approach for avoiding geometric intersection operations in reachability analysis by over-approximating the intersection with a nonlinear map for guard sets modeled as halfspaces.

In their previous work, [1] proposed a method for solving the flow/guard intersection problem. It first derives an analytical expression for the boundaries of continuous flows using interval Taylor methods and techniques for controlling the wrapping effect. It then expresses the event detection and localization problem underlying flow/guard intersection as a constraint-satisfaction problem (CSP).

In this paper, we propose technical improvements of the latter method in order to reduce both the computation time while ensuring reduced over-approximation when crossing the guard sets conditions. Our method implements Lohner's QR-factorization method [4] within the guaranteed numerical integration method to control the wrapping effect, applies the standard contractor *HC4Revise* at transition step only, and makes use of bisection in a single dimension to ensure polynomial time complexity. We will illustrate that our method exhibits small computation time while minimizing the overestimation over the whole hybrid flow trajectory.

The paper is organized as follows. Sect. 2 introduces notations 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 interval Taylor methods. Sect. 5 contains the main contribution of this paper. Results are presented in Sect. 6 onto two hybrid systems.

This work is supported by the French National Research Agency under contract ANR 2011 INS 006 MAGIC-SPS (projects.laas.fr/ANR-MAGIC-SPS)

M. Maïga, Université d'Orléans, PRISME, Bourges, France mmaiga@laas.fr

N. Ramdani, Université d'Orléans, PRISME, Bourges, France nacim.ramdani@univ-orleans.fr

L. Travé-Massuyès, CNRS, LAAS, Toulouse, France louise@laas.fr

II. UNCERTAIN NONLINEAR HYBRID AUTOMATA

Dynamical hybrid systems can be represented by a hybrid automaton [2] given by $H = (\mathscr{Q}, \mathscr{D}, \mathscr{P}, \Sigma, \mathscr{A}, \mathrm{Inv}, \mathscr{F})$ and defined as follows:

- $\mathscr{Q}$ is a set of locations $\{q\}$ whose continuous dynamics, i.e. flow transitions, are described by non-autonomous differential equations $f_q \in \mathscr{F}$ of the form
$$\mathrm{flow}(q): \quad \dot{x}(t) = f_q(x, p, t), \tag{1}$$
where $f_q : \mathscr{D} \times \mathscr{P} \times \mathbb{R}^+ \mapsto \mathscr{D}$ is nonlinear and assumed sufficiently smooth over $\mathscr{D} \subseteq \mathbb{R}^n$, with dimension n that may depend on q, and $p \in \mathscr{P}$, where $\mathscr{P}$ 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:
$$\mathrm{Inv}(q): \quad \nu_q(x(t), p, t) < 0, \tag{2}$$
where inequalities are taken componentwise, $\nu_q : \mathscr{D} \times \mathscr{P} \times \mathbb{R}^+ \mapsto \mathbb{R}^m$ is also nonlinear, and the number m of inequalities may also depend on q.
- $\mathscr{A}$ is the set of discrete transitions $\{e = (q \to q')\}$ given by the 5-uple $(q, \mathrm{guard}, \sigma, \rho, q')$, where q and q' represent upstream and downstream locations respectively; guard is a condition of the form:
$$\mathrm{guard}(e): \quad \gamma_e(x(t), p, t) = 0; \tag{3}$$
σ is an event, and ρ is a reset function assumed to be affine.

A transition $q \to q'$ may occur when the continuous state flow reaches the guard set, i.e. when the continuous state satisfies condition (3).

The set reachable in finite time by system (1-3) is illustrated in Fig. 1(a). When starting from an initial state vector $x(t_0)$ taken in $\mathscr{X}(t_0)$, a discrete transition occurs when the continuous flow intersects the guard set at time t_e. Then, the continuous state vector is reset as $x_1(t_e^+) = \rho_0(x_0(t_e^-))$. $\mathscr{X}_e$ is the set of all possible $x(t_e)$ when $x(t_0)$ varies in $\mathscr{X}(t_0)$. The reachable set may intersect a forbidden area as shown in the figure.

Introducing the new state variable $z = (x, p, t)$ with $\dot{z} = (\dot{x}, 0, 1)$, and defining $\mathscr{Z} = \mathscr{D} \times \mathscr{P} \times \mathbb{R}^+$, equations (1–3) are rewritten:

$$\mathrm{flow}(q): \quad \dot{z}(t) = f_q(z), \tag{4}$$
$$\mathrm{Inv}(q): \quad \nu_q(z(t)) < 0 \quad \text{and} \tag{5}$$
$$\mathrm{guard}(e): \quad \gamma_e(z(t)) = 0. \tag{6}$$

so that all uncertain quantities are embedded in the initial state vector.

III. INTERVALS ANALYSIS

A. Interval arithmetics

A real interval $[u] = [\underline{u}, \overline{u}]$ is a closed and connected subset of R where $\underline{u}$ represents the lower bound of $[u]$ and $\overline{u}$ represents the upper bound. The width of an interval $[u]$ is defined by $w(u) = \overline{u} - \underline{u}$, and its midpoint by $Mid(u) = (\overline{u} + \underline{u})/2$. The set of all real intervals of R is denoted IR. Two intervals $[u]$ and $[v]$ are equal if and only if $\underline{u} = \underline{v}$ and $\overline{u} = \overline{v}$.

Real arithmetic operations are extended to intervals [13]. Arithmetic operations on two intervals $[u]$ and $[v]$ can be defined by:

$$\circ \in \{+, -, *, /\}, \quad [u] \circ [v] = \{x \circ y \mid x \in [u],\ y \in [v]\}.$$

An interval vector (or box) $[X]$ is a vector with interval components and may equivalently be seen as a cartesian product of scalar intervals:

$$[X] = [x_1] \times [x_2] \ldots \times [x_n].$$

The set of $n-$dimensional real interval vectors is denoted by IR^n.

An interval matrix is a matrix with interval components. The set of $n \times m$ real interval matrices is denoted by $\mathrm{IR}^{n \times m}$. The width $w(.)$ of an interval vector (or of an interval matrix) is the maximum of the widths of its interval components. The midpoint $Mid(.)$ of an interval vector (resp. an interval matrix) is a vector (resp. a matrix) composed of the midpoint of its interval components.

Classical operations for interval vectors (resp. interval matrices) are direct extensions of the same operations for punctual vectors (resp. point matrices) [13]. Let $f : \mathrm{R}^n \to \mathrm{R}^m$, the range of the function f over an interval vector $[u]$ is given by:

$$f([u]) = \{f(x) | x \in [u]\}.$$

The interval function $[f]$ from IR^n to IR^m is an inclusion function for f if:

$$\forall [u] \in \mathrm{IR}^n, \quad f([u]) \subseteq [f]([u]).$$

An inclusion function of f can be obtained by replacing each occurrence of a real variable by its corresponding interval and by replacing each standard function by its interval evaluation. Such a function is called the natural inclusion function. In practice the inclusion function is not unique, it depends on the formal expression of f.

B. Constraint Satisfaction Problems

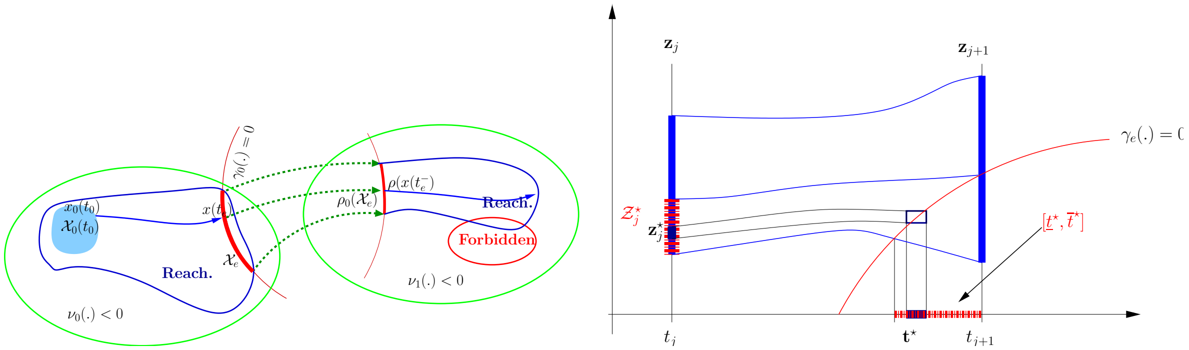
When manipulating sets of values, many problems can be formulated as a CSP. Solving interval CSPs makes use of two main operations:

1) *Bisection* is an operation that partitions a n-dimensional box $[X] \in \mathrm{IR}^n$ along one of its components j in two other n-dimensional boxes $L[X]$ and $R[X]$ ([9]):

$$L[X] \equiv [\underline{x_1}, \overline{x_1}] \times \cdots \times [\underline{x_j}, \frac{\underline{x_j} + \overline{x_j}}{2}] \times \cdots \times [\underline{x_n}, \overline{x_n}]$$
$$R[X] \equiv [\underline{x_1}, \overline{x_1}] \times \cdots \times [\frac{\underline{x_j} + \overline{x_j}}{2}, \overline{x_j}] \times \cdots \times [\underline{x_n}, \overline{x_n}]$$

A classical bissection strategy is to choose $j \overset{\mathrm{def}}{=} \max_n \{i \,|\, \overline{x_i} - \underline{x_i} = w([X])\}$, i.e. the index of the component of greatest length of $[X]$. We have

$$[X] = L[X] \cup R[X].$$



(a) Set reachable in finite time by system (1-3)

(b) Solving flow/guard intersection

Fig. 1. Hybrid reachability.

2) *Contraction* is an operation which reduces an n-dimensional box $[X] \in \mathrm{IR}^n$ to its intersection with respect to another set S. Let S be a set defined by non-linear inequalities, a contractor [9] for S is a function $C_\mathrm{S} : \mathrm{IR}^n \to \mathrm{IR}^n$ defined as:

$$\mathrm{S} \cap [X] \subseteq C_\mathrm{S}([X]) \subseteq [X] \tag{7}$$

IV. ENCLOSING UNCERTAIN HYBRID FLOWS

A. Guaranteed set integration with interval Taylor methods

Consider the *uncertain* dynamical system described by (4)–(6) with $z(t_0) \in \mathscr{Z}_0$ at time $t_0 \geq 0$ and denote by $\mathscr{Z}(t;t_0,\mathscr{Z}_0)$ the set of solutions of (4) at time t originating from each initial condition in $\mathscr{Z}_0$ at t_0. Define a time grid $t_0 < t_1 < t_2 < \ldots < t_{n_T}$, which is taken equally spaced in this paper, and assume $\mathscr{Z}_0 = z_0 = [\underline{z}_0, \bar{z}_0]$.

Guaranteed set integration via interval Taylor methods computes interval vectors $[z_j], j = 1,\ldots,n_T$, that are *guaranteed* to contain the set of solutions $\mathscr{Z}(t_j;t_0,\mathscr{Z}_0)$ of (4) at times t_j, $j = 1,\ldots,n_T$ in three steps:

- verify the existence and uniqueness of the solution using the Banach fixed point theorem and the Picard-Lindelöf operator [13], [14], [15].
- compute an a priori enclosure $[\tilde{z}_j]$ such that $[\tilde{z}_j] \supseteq \mathscr{Z}(t)$ for all t in $[t_j,t_{j+1}]$.
- compute a tighter enclosure for the set of solutions of (4) at t_{j+1} using a Taylor series expansion of order k of the solution at t_j, where $[\tilde{z}_j]$ is used to enclose the remainder term:

$$\mathscr{Z}(t;t_j,z_j) \subseteq [z](t;t_j,[z_j]) = \\ [z_j] + \sum_{i=1}^{k-1} (t-t_j)^i \mathrm{f}_q^{[i]}([z_j]) + (t-t_j)^k \mathrm{f}_q^{[k]}([\tilde{z}_j]), \tag{8}$$

where t can be taken as t_{j+1} or any $t \in [t_j,t_{j+1}]$, which is not necessarily on the time-grid, and the $\mathrm{f}_q^{[i]}([z_j])$ are the Taylor coefficients.

In [7], (8) is turned into a computationally acceptable scheme that controls the wrapping effect by using the *mean-value* approach complemented by QR-factorization as proposed by Lohner [7]. The solution enclosure at time $t \in [t_j,t_{j+1}]$ can be computed in the form:

$$\mathscr{Z}(t;t_j,\mathscr{Z}_0) \in \{v(t) + A(t)r(t) \,|\, v(t) \in [v](t), r(t) \in [r](t)\},$$

Defining:

$$[\chi](t) \equiv \{[z](t),\, \hat{z}(t),\, [v](t),\, [r](t),\, A(t)\}, \tag{9}$$

where $\hat{z}(t) := \mathrm{Mid}([z](t))$, the algorithm used in this paper to compute the solution set of (4) at time $t \in [t_j, t_{j+1}]$ is the following:

```
Algorithm $\varphi^{l_{qr}}$(input : $[\chi_j]$, $t_j$, $t$, $[\tilde{z}_j]$, output : $[\chi](t)$)
  1) $[v](t) := \hat{z}_j + \sum_{i=1}^{k-1} (t-t_j)^i \mathrm{f}_q^{[i]}(\hat{z}_j) + (t-t_j)^k \mathrm{f}_q^{[k]}([\tilde{z}_j])$
  2) $[S](t) := \mathbb{I} + \sum_{i=1}^{k-1} (t-t_j)^i \frac{\partial \mathrm{f}_q^{[i]}}{\partial z}([z_j])$
  3) $[q](t) := ([S](t)A_j)[r_j] + [S](t)([v_j] - \hat{z}_j)$
  4) $[z](t) := [v](t) + [q](t)$
  5) $A(t) = l_{qr}(A_j, S_j, r_j)$ see Algorithm Lohner's QR factorization
  6) $[r](t) := A(t)^{-1}([S](t)A_j)r_j + (A(t)^{-1}[S])([v_j] - \hat{z}_j)$
  7) $\hat{z}(t) :=$ Mid($[v](t)$)
  8) $[\chi](t) := \{[z](t),\, \hat{z}(t),\, [v](t),\, [r](t),\, A(t)\}$
```

Then, the solution enclosure at time t_{j+1} is given by $[\chi_{j+1}] = \varphi^{l_{qr}}([\chi_j], t_j, t_{j+1}, [\tilde{z}])$.

In Algorithm $\varphi^{l_{qr}}(.)$, which implements the integration step, QR-factorization plays an essential role in controlling the wrapping effect by choosing an appropriate coordinate transformation. This step is discussed in details in section IV-B.

B. Lohner's QR factorization

Lohner's QR-factorization method works as follows. It rotates the coordinates frame so that the first coordinate is parallel to the edge of greatest length of the parallelepiped $A[r]$ (where $[r]$ is computed on line 6 of Algorithm $\varphi^{l_{qr}}$). This operation reduces the spurious zones introduced by set enclosure in a significant way, hence reduce the wrapping effect and therefore improve solution approximation. Prior to QR-factorization, Lohner's method computes the lengths of each edge of the parallelepiped $A[r]$ then rearranges the columns of A in decreasing length order.

Let $\hat{A}_{j+1} = \tilde{A}_{j+1}.P_{j+1}$, with P_{j+1} be a permutation matrix. We have in the general case:

$$\hat{A} \in \{S_j \cdot A_j |\ S_j \in [S_j]\} \tag{10}$$

In this paper we take (see line 5, Algorithm $\varphi^{l_{qr}}$) :

$$\tilde{A}_{j+1} \in Mid(S_j \cdot A_j), \text{ with } S_j \in [S_j].$$

TABLE I

ALGORITHM `Lohner's QR factorization`

```
Algorithm $l_{qr}$(input : $A_j, S_j, r_j$, output : $A(t), l_i$)
1) $A(t) := Mid((S_j)A_j)$;
2)       Compute $l_i$ via 11
3)           sort($l_i$) and $A_{perm}(t) := Permute(A(t))$
4)               Compute $Q(t) \leftarrow A_{perm}(t) = Q(t)R(t)$
5)       Choice $A(t) := Q(t)$
```

By computing the length of each edge of the parallelepiped $A[r]$ by the following formula:

$$l_i = ||A^i_{j+1}||_2.w([r_j]^i) \tag{11}$$

where A^i_{j+1} the ith column of A, which we denote in the remainder of this paper $col(A^i)$, $[r_j]^i$ the ith component of r_j. $||.||_2$ represents the Euclidean norm.

Example 1: To illustrate Lohner's method, let us consider matrix $\tilde{A}$ and parallelepiped $[r]$ below:

$$\tilde{A} = \begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix} \quad [r] = \begin{pmatrix} [1,2] \\ [1,4] \end{pmatrix}$$

By calculating the lengths of the edges of the parallelepiped following formula (11), we have:

$$l_1 = \sqrt{(1^2+2^2)}.(2-1); \quad l_2 = \sqrt{(1^2+1^2)}.(4-1)$$

We have $l_2 > l_1$, therefore, according to Lohner's criterion, we permute columns 1 and 2 of matrix $\tilde{A}$, using permutation matrix P:

$$\hat{A} = \begin{pmatrix} 1 & 1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$$

with $P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.

Having determined $[\chi_{j+1}]$, [1] proposes to compute the flow/invariant intersection at time t_{j+1} by solving the CSP:

$$([v_{j+1}] \times r_{j+1}, \text{`}v_q([v_{j+1}] + A_{j+1}[r_{j+1}]) < 0'). \tag{12}$$

If $[v_{j+1}]' \times [r_{j+1}]'$ is the set of solutions of CSP (12), the solution set $[z'_{j+1}] = \text{inv}(q) \cap [z_{j+1}]$ is given by:

$$[z]'_{j+1} = \{v_{j+1} + A_{j+1}r_{j+1}, |\, v_{j+1} \in [v_{j+1}]',\ r_{j+1} \in [r_{j+1}]'\}.$$

Numerical implementation: For the experimentation purposes of this paper, the above system-solving methods have been implemented in the `IBEX` C++ library [10]. We use the `Profil/Bias` C++ class library [11] for interval computation, the `FABDAB++` package [12] for automatic differentiation, and `AML++` package for Linear algebra.

V. Minimizing the overestimation at flow-guard intersection

The main contribution of this paper targets reducing the duration of guard crossing by minimizing the overestimation of the flowpipe-guard sets intersection at transition times. We use the HC4revise contractor as implemented by IBEX library.

A. Detecting and localizing hybrid transitions

In their previous work [1], given Algorithm $\varphi^{l_{qr}}(.)$, detecting and localizing hybrid transition is shown to be equivalent to finding the set $[\underline{t}^\star, \overline{t}^\star] \times \mathscr{Z}^\star_j$, where $\mathscr{Z}^\star_j$ is the initial set consisting of the initial state vectors $z^\star_j$ that lead to a $z(t_e; t_j, z^\star_j)$ that satisfies (6) at t_e and $t_e \in [\underline{t}^\star, \overline{t}^\star]$, as depicted on Fig. 1(b). Accounting for the mean value form introduced in section IV-A, finding such set is performed by solving the following CSP:

$$\left([t_j, t_{j+1}] \times r_j, \text{`}\gamma_e\big(\varphi(\chi_j(.), ., t, .)\big) = 0'\right). \tag{13}$$

If the set $[\underline{t}^\star, \overline{t}^\star] \times [r^\star_j]$ is not empty, one can assume that the event $e = q \rightarrow q'$ occurs at $t_e = \underline{t}^\star$ and that $[\chi](t^-_e) = [\chi]([\underline{t}^\star, \overline{t}^\star]$.

The discrete transition can then be computed from $[z(t^-_e)]$ thanks to the reset function ρ. [1] enhances the method by proposing a bisection strategy that account for the size of $[\underline{t}^\star, \overline{t}^\star]$ and $[\mathscr{Z}^\star_j]$.

In this paper, we make a relaxation on state variables, i.e. we do not search all initial states vectors that lead the flow to satisfy the guard condition. Indeed, we merely compute a priori solution $[\tilde{z}]$ over $[t_0, t_1]$, and check if this solution is verified by guard condition.

$$0 \in \gamma_e([\tilde{z}]) \tag{14}$$

If this is the case, we compute a new solution $[\chi]^\star$ from initial solution $[\chi]_0$ at t_0. And then, we use a contractor $\mathbb{C}_e$ (see section V-B) to contract the domain of solution $[\chi]^\star$ according to guard condition.

The event $e = q \rightarrow q'$ occurs at $t_e = \underline{t}^\star$, if the width of $[\underline{t}^\star, \overline{t}^\star]$, and $\mathbb{C}_e([\chi]^\star)$ are lower than given thresholds, and the domain of solution $[\chi]^\star$is sufficiently contracted.

B. Domain Contraction

HC4Revise or forward-backward contractor computes the projection of a constraint onto all the variables and updates the domains of each variable by pruning the parts inconsistant with the result of the corresponding projection. In this paper, we build for each guard condition, a contractor. We contract flow $[\chi]^\star$ enclosure over $[\underline{t}^\star, \overline{t}^\star]$ before performing jump operation. In order to control computational complexity, we allow several bisection of the time variable only, whereas the state vector $[\chi]^\star$ is not bisected (or only into a very small number of subboxes). Doing so, we can keep compuational complexity polynomial as bisection is performed in a single direction only. Furthermore, we also control the overall over-approximation introduced by reset functions $\rho_e([\chi]^\star_c))$ by tuning threshold ε_T to a small value.

To summarize our method, we use for the continuous expansion of the hybrid system a combination of the interval Taylor method and the Lohner's one, and in the vicinity of the guard condition we use a contractor and a single-dimension bisection to enclose as precisely as possible the flow-guard sets intersection solution set, then we perform a jump and apply the reset functions. When the flow has completely crossed the guard condition, we carry on in the new mode in a similar way until the flow intersects a new guard sets condition.

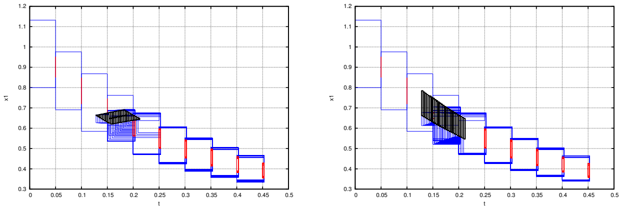
VI. NUMERICAL EVALUATIONS

A. Example 1

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

$$\begin{cases} \text{flow}(1) : f_1 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1-(b_1+1)x_1+a_1x_1^2x_2 \\ b_1x_1-a_1x_1^2x_2 \end{pmatrix} \\ \text{inv}(1) : \nu_1(x_1,x_2) = -4x_1+x_2+2 \\ \text{flow}(2) : f_2 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1-(b_2+1)x_1+a_2x_1^2x_2 \\ b_2x_1-a_2x_1^2x_2 \end{pmatrix} \\ \text{inv}(2) : \nu_2(x_1,x_2) = -\nu_1(x_1,x_2) \\ \text{guard}(1) : \gamma_1(x_1,x_2) = \nu_1(x_1,x_2) \\ \text{reset}(1) : \rho_1(x_1,x_2) = (\alpha_1x_1,\alpha_2x_2) \end{cases} \tag{15}$$

with $\alpha_1 = \alpha_2 = (1;1)$, $a_1 = 1.5, a_2 = 3.5$, $b_1 = 1$, $b_2 = 3.5$ and $x_0 \in [1,1.1] \times [0.1,0.3]$. We took for this simulation a constant integration time step $h = 0.05$, the time interval was bisected until a threshold $\varepsilon_T = 0.005$, and the continuous state vector was bisected until a threshold $\varepsilon_z = 0.2$. When we compare the results obtained in (Figures 3). We see that our method with the contractor yields better computation times than without the contactor. When one focuses on the computation time required for crossing the guard sets, we can see that the use of the contractor halves computation time. We also obtain a smaller number of generated boxes. (see Table II)



(a)

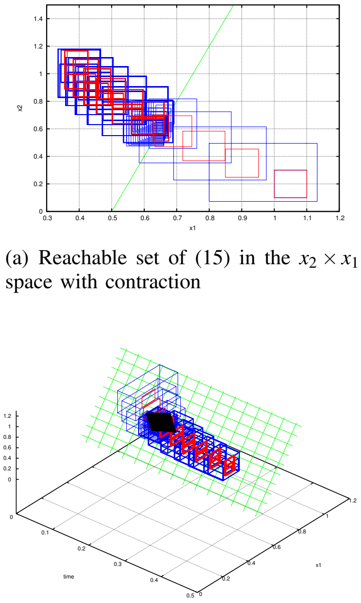
(b)

Fig. 2. Reachable set in the $x_1 \times t$ space : Fig.2(a) with contraction, CPU times=0.308s, TCG=0.148s, NBox=38; Fig. 2(b) without contraction, CPU times=0.376s, TCG=0.192s, NBox=44

B. Example 2 : Vehicle Model

The dynamics of a non-holonomic vehicle[1] is given as follows :

$$\begin{array}{lll} \frac{dx}{dt} = vc_t; & \frac{dy}{dt} = vs_t; & \frac{dv}{dt} = u_1 \\ \frac{dc_t}{dt} = \sigma v^2 s_t; & \frac{ds_t}{dt} = -\sigma v^2 c_t; & \frac{d\sigma}{dt} = u_2 \end{array} \tag{16}$$



(a) Reachable set of (15) in the $x_2 \times x_1$ space with contraction

(b) Reachable set of (15) in 3D with contraction

Fig. 3. Reachable set in the $x_2 \times x_1$ space and 3D representation.

Where u_1, u_2 are control inputs. We consider the case of a vehicle with three control modes m_1, m_2, m_3. The control inputs are given by (u_1,u_2) = $(-0.05,-0.1)$ for mode m_1, $(0,0)$ for m_2 and $(0.05,0.1)$ for m_3. The transitions between modes are shown in Figure 4. We start our simulation from m_1 with the initial variable values :

$$\begin{array}{ccc} x \in [1,1.2] & y \in [1,1.2] & v \in [0.8,0.81] \\ s_t \in [0.6,0.61] & c_t \in [0.7,0.71] & \sigma = [0,0.01] \end{array}$$

We took for this simulation a constant integration time step $h = 0.5$, the time interval was bisected until a threshold $\varepsilon_T = 0.05$, and the continuous state vector was bisected until a loose threshold $\varepsilon_z = 1$. We obtain for this simulation, when we use contractor without merging the boxes (see section VI-C). All results are ploted on Figures 6(a) for the case with contraction but without merging the solution boxes, and on Figures 6(b) for the case with contraction and while merging the solution boxes (see section VI-C).

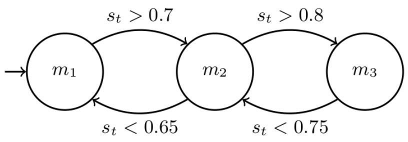


Fig. 4. Hybrid automaton of the vehicle model

C. Computing the hull of the solution boxes

Even though bisection is performed on a single direction only, our method generates in practice a large number of boxes to cover the solution set, i.e. the hybrid reachable set (see Table II). The natural idea to reduce this number is to merge these boxes and compute a box that contains

[1] Bench proposed by Xin Chen and Sriram Sankaranarayanan

TABLE II

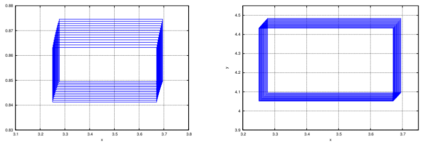
SIMULATION WITH DIFFERENT THRESHOLD ε_z, **TTCPU**=TOTAL TIME CPU,**TCG**=TOTAL TIME CROSSING GUARD CONDITION,**VOLUME**=$\prod_{i=1}^{n} w(z_i)$, **NBOX**= NUMBER OF BOXES GENERATED.

	TTCPU(s)	**TCG** (s)	**Volume**	**Nbox**		**TTCPU**(s)	**TCG** (s)	**Volume**	**Nbox**
With contraction					Without contraction				
$\varepsilon_z = 0.04$	1.452	0.656	0.0100893	161	$\varepsilon_z = 0.04$	2.084	1.032	0.00341105	206
$\varepsilon_z = 0.05$	1.324	0.648	0.0104943	156	$\varepsilon_z = 0.05$	1.892	1.020	0.00341105	206
$\varepsilon_z = 0.06$	1.276	0.532	0.0107828	150	$\varepsilon_z = 0.06$	1.644s	0.740	0.00784587	171
$\varepsilon_z = 0.07$	0.620	0.332	0.0157061	77	$\varepsilon_z = 0.07$	0.764	0.404	0.00537186	94
$\varepsilon_z = 0.08$	0.596	0.320	0.015505	76	$\varepsilon_z = 0.08$	0.856	0.432	0.00537186	94

their convex hull. But this may introduce some spurious overapproximation.

We merge at a given time, boxes belonging to the same mode (see Figure 5(a)). We will apply this box-merging algorithm to the vehicle benchmark and compare the simulation results with and without boxes merging.

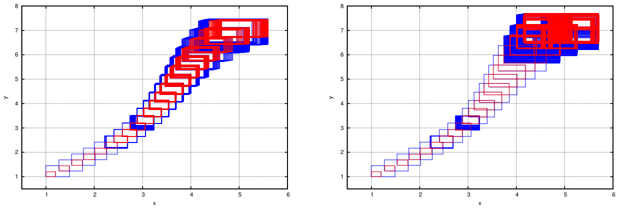
The operation of merging of boxes has advantages and disadvantages. Its advantage is that it reduces list length, at a given time, hence contribute to reducing the overall computation time. Its disadvantage lays in the significant spurious wrapping effect that may be introduced when merging a set of boxes. This overapproximation accumulates over time.



(a) St vs X (b) Y vs X

Fig. 5. Solution boxes captured at t=6s

These effects can be seen in Figure 6(b). When the boxes are fairly close to one another as shown in Figure 5(b), merging the boxes does not introduce over-approximation. Merging boxes depends strongly on the arrangement of the boxes.



(a) Reachable set of in the $y \times x$ space with contraction only

(b) Reachable set of in the $y \times x$ space with contraction and boxes merging

Fig. 6. Reachable set of (16).

VII. CONCLUSIONS AND FUTURE WORKS

In this paper, we proposed an effective method based on the geometrical transformation introduced by Lohner's QR-factorization method combined with HC4Revise contractor for solving flowpipe-guard set intersection in a fast and effective way. On a simple 2-dim example, it already shows better performance, smaller computation times and lower complexity than methods without contractor. We also tested our algorithm on a 6-dim non-holonomic vehicle benchmark, and the performance of our method are very promising, showing fairly small overapproximation and very acceptable computation time.

Future work will investigate ways to merge solution boxes while controlling wrapping effect.

REFERENCES

[1] N. Ramdani and N.S. Nedialkov. Computing reachable sets for uncertain nonlinear hybrid systems using interval constraint propagation techniques. In *Nonlinear Analysis Hybrid Systems*, , pages 149–162, 2011.

[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] R. Alur, T. Dang, J. Esposito, Y. Hur, F. Ivancic, V. Kumar, I. Lee, P. Mishra, G.J. Pappas, and Sokolsky. Hierarchical modeling and analysis of embedded systems, 2003.

[4] R. J. LOHNER, *Enclosing the solutions of ordinary initial and boundary value problems*, in Computer Arithmetic: Scientific Computation and Programming Languages, E. W. Kaucher, U. W. Kulisch, and C. Ullrich, eds., Wiley-Teubner Series in Computer Science, Stuttgart, 1987, pp. 255–286.

[5] G. Frehse, R. Rajarshi Flowpipe-Guard Intersection for Reachability Computations with Support Functions. In *ADHS'12 (2012)* .

[6] Althoff, M. and Krogh, B.H. (2012). Avoiding geometric intersection operations in reachability analysis of hybrid systems. In *HSCC*, 45–54.

[7] 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.

[8] 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.

[9] Jaulin L., Kieffer M., Didrit O., and Walter E. *Applied Interval Analysis*. Springer-Verlag, 2001.

[10] http://www.ibex-lib.org/.

[11] http://www2.imm.dtu.dk/˜km/fadbad/.

[12] http://www.ti3.tu-harburg.de/software/profilenglisch.html.

[13] R.E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, 1966.

[14] 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 Velag, Berlin, 1996.

[15] 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.