Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
Electronic Notes in Theoretical Computer Science 223 (2008) 87–102
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

The Reach-and-Evolve Algorithm for Reachability Analysis of Nonlinear Dynamical Systems

Pieter Collins[2]

Centrum voor Wiskunde en Informatica
Amsterdam, The Netherlands

Alexandre Goldsztejn[3]

CNRS, LINA, UMR 6241
Nantes, France

Abstract

This paper introduces a new algorithm dedicated to the rigorous reachability analysis of nonlinear dynamical systems. The algorithm is initially presented in the context of discrete time dynamical systems, and then extended to continuous time dynamical systems driven by ODEs. In continuous time, this algorithm is called the Reach and Evolve algorithm. The Reach and Evolve algorithm is based on interval analysis and a rigorous discretization of space and time. Promising numerical experiments are presented.

Keywords: Reachability analysis, nonlinear dynamical systems, interval analysis.

1 Introduction

We consider the computation of the reachable set of a dynamical system. For a set of initial conditions $\mathbf{X}_0 \subseteq \mathbb{R}^n$ and a system $\mathbf{f}$ with time axis $\mathbb{T}$, the reachable set is defined as

$$\mathrm{Reach}(\mathbf{f},\mathbf{X}_0) := \{\mathbf{y} \mid \exists \text{ solution } \mathbf{x}(\cdot) \text{ of } \mathbf{f} \text{ and } t \in \mathbb{T} \text{ s.t. } \mathbf{x}(0) \in \mathbf{X}_0 \text{ and } \mathbf{x}(t) = \mathbf{y}\}. \quad (1)$$

[1] P. Collins was partially supported by the Nederlandse Wetenschappelijk Organisatie (NWO) through VIDI project number 639.032.408 .

[2] Email: Pieter.Collins@cwi.nl

[3] Email: Alexandre.Goldsztejn@univ-nantes.fr

doi:10.1016/j.entcs.2008.12.033

We present a high-level algorithm for computing over-approximations reachable set, which can be effectively implemented in different ways. We consider both discrete-time systems defined by iterated maps

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k)), \tag{2}$$

where $\mathbf{x}(k) \in \mathbb{R}^n$ and $\mathbf{f} : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is a continuous function, and continuous-time systems defined by differential equations

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)), \tag{3}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ and $\mathbf{f} : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is a locally Lipschitz function.

In this paper, we present a high-level algorithm for rigorously computing an enclosure of the reachable set (1) for dynamical systems (2,3) defined by arbitrary nonlinear continuous functions. In continuous time, we call our algorithm the *Reach-and-Evolve* algorithm (*R&E*) since it involves computing *reachability steps* and *evolution steps* for the system. The algorithms are based on interval analysis which is introduced in Section 2. Section 3 presents the R&E algorithm, and some of its properties. Promising experiments are finally presented in Section 4.

2 Interval Analysis

The modern interval analysis was born in the 60's with [14]. Since, it has been widely developed and is today one important branch of numerical analysis (see [1,16,8,7] and extensive references). The main concepts of interval analysis that will be used in the sequel are now presented.

2.1 Interval and Interval Vectors

Interval analysis usually considers only closed intervals. The set of these intervals is denoted by $\mathbb{IR}$. An interval is usually denoted using brackets. As often as possible, the bounds of the interval $[x]$ are denoted by $\underline{x}$ and $\overline{x}$, i.e. $[x] = [\underline{x}, \overline{x}]$, and an element of $[x]$ by x. Interval vectors (boxes) can be defined in two equivalent ways: First as a vector of intervals $[\mathbf{x}] = ([x_1], \ldots, [x_n])$. In this case $\mathbf{x} \in [\mathbf{x}]$ is defined by $x_1 \in [x_1]$, ..., $x_n \in [x_n]$. Second, as a interval of vectors $[\mathbf{x}] = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$ where $\underline{\mathbf{x}}, \overline{\mathbf{x}} \in \mathbb{R}^n$ such that $\underline{\mathbf{x}} \leq \overline{\mathbf{x}}$, the inequality being defined component-wise. In this case $\mathbf{x} \in [\mathbf{x}]$ is defined by $\underline{\mathbf{x}} \leq \mathbf{x} \leq \overline{\mathbf{x}}$. Both definitions are obviously equivalent, and used indifferently. Interval matrices are defined in the same way.

2.2 Interval Extensions

The main concept of interval analysis is the extension of real functions to intervals, which is defined as follows:

Definition 2.1 Let $\mathbf{f} : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ be a continuous real function, and $[\mathbf{f}] : \mathbb{IR}^n \longrightarrow \mathbb{IR}^m$ be an interval function. Then $[\mathbf{f}]$ is an interval extension of $\mathbf{f}$ if and only if for every $[\mathbf{x}] \in \mathbb{IR}^n$, $\{\mathbf{f}(\mathbf{x}) : \mathbf{x} \in [\mathbf{x}]\} \subseteq [\mathbf{f}]([\mathbf{x}])$.

Remark 2.2 It is important to note the difference between $\mathbf{f}([\mathbf{x}])$ which is the exact range of the function over $[\mathbf{x}]$, i.e. $\mathbf{f}([\mathbf{x}]) = \{\mathbf{f}(\mathbf{x}) : \mathbf{x} \in [\mathbf{x}]\}$ and $[\mathbf{f}]([\mathbf{x}])$ which is the evaluation of an interval function. Usually, the symbol $[\mathbf{f}]$ is used for an interval extension of the real function $\mathbf{f}$, and in this case $\mathbf{f}([\mathbf{x}]) \subseteq [\mathbf{f}]([\mathbf{x}])$.

Hence, an interval extension allows computing enclosures of the image of boxes by real functions. This definition is very useful in many contexts, obviously including reachability analysis. It now remains to show how to compute such extensions.

The first step is to compute formally the interval extension of elementary functions. For example, we define $[x] + [y] := [\underline{x} + \underline{y}, \overline{x} + \overline{y}]$. Similar simple expressions are obtained for other functions like $-, \times, \div, x^n, \sqrt{x}, \exp, \ldots$. This process gives rise to the so-called *interval arithmetic*.

Then, an interval extension for real functions compound of these elementary operations is simply obtained changing the real operations to their interval counterparts. This interval extension is called the *natural interval extension*.

Example 2.3 Let $f(x, y) = x(y - x)$. The interval function $[f]([x], [y]) = [x]([y] - [x])$ is the natural interval extension of f. Hence for example

$$[f]([0, 1], [-1, 1]) = [-2, 1] \supseteq \{f(x, y) : x \in [0, 1], y \in [-1, 1]\}. \tag{4}$$

Note that the exact range is $f([0, 1], [-1, 1]) = [-2, 1/4]$, and the natural interval extension is thus pessimistic. One main issue of interval analysis is to fight this pessimism introduced by the interval evaluation of a function.

There are other interval extensions, in particular the mean-value interval extension which uses the natural extension of the derivatives to try improving the enclosure. See [16] for details.

Finally, the interval arithmetic also allows extending vector/matrix and matrix/matrix multiplications. Such definitions preserve the enclosing property of interval extensions (they are actually special cases of interval extensions).

Remark 2.4 When one represents numbers using a finite precision [5], the previous operations cannot be computed in general. The outer rounding is then used so as to keep valid the interpretations. For example, $[1, 2]/[3, 7]$ would result of $[0.142, 0.667]$ if rounded with a three decimal accuracy. Such an outer rounding is implemented in most interval libraries, e.g. [9,20,21].

2.3 Rigorous Iteration of Maps

While it is possible to iterate a dynamical system $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ using an interval extension $[\mathbf{f}] : \mathbb{IR}^n \to \mathbb{IR}^n$ directly, the errors of this approximation tend to grow exponentially due to the so-called *wrapping effect*. There are many methods used in the literature to reduce the wrapping effect, all of which rely on using higher-order *enclosures* of the set of evolved points. Such methods include ellipsoidal calculus [11], orthogonal parallelotopes [12], zonotopes [10] and Taylor models [13]. We henceforth assume a class $\mathbb{E}$ of *enclosure sets* which are used to enclose the

current state set, and an enclosure extension $\langle \mathbf{f} \rangle : \mathbb{E} \to \mathbb{E}$ of $\mathbf{f}$ such that

$$\langle \mathbf{f} \rangle(\langle \mathbf{x} \rangle) \supset \mathbf{f}(\langle \mathbf{x} \rangle). \tag{5}$$

Of course, we want the enclosure extension to be better than the interval extension i.e. $\langle \mathbf{f} \rangle([\mathbf{x}]) \subset [\mathbf{f}]([\mathbf{x}])$ for $[\mathbf{x}] \in \mathbb{IR}^n \cap \mathbb{E}$.

2.4 Rigorous Integration of Ordinary Differential Equations

The rigorous integration of continuous time dynamical systems driven by ODEs is an important application of interval analysis. It can be formalized as follows. The ODE $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t))$ gives rise to an operator $\Phi : \mathbb{R} \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$ which associate $\mathbf{x}(t)$ to $\mathbf{x}(0)$, i.e. $\Phi(t, \mathbf{x}(0)) = \mathbf{x}(t)$. This operator is called the *solution operator* of the ODE. Note that in general the solution operator is not defined inside the whole space $\mathbb{R} \times \mathbb{R}^n$ but instead in a subspace. For clarity, we will use $\mathbb{R} \times \mathbb{R}^n$.

Interval analysis offers a wide variety of methods dedicated to computing interval extensions and enclosure extensions of Φ. This means that given a time interval $[t]$ and an enclosure $\langle \mathbf{x} \rangle$, the evaluation of an enclosure extension $\langle \Phi \rangle$ gives rise to an enclosure that contains all solutions which start inside $\langle \mathbf{x} \rangle$ after evolution for a duration included inside $[t]$. This rigorous enclosure of the flow is basically obtained using a truncated Taylor series with a rigorous bound on the remainder. For details on parallelepiped methods, see the survey paper [15] and references therein. For the Taylor method see [13] and references therein. It is also very useful to rigorously compute the space derivatives of the flow [22,6].

For completeness, a simple interval integrator is presented in Appendix A. It is based on a first order Taylor expansion with rigorous bound of the remainder. Higher order expansions are much more efficient, but the method presented in Appendix A is simple to present and gives rise to interesting results.

3 The Reach and Evolve Algorithm

3.1 Bounded Reachability Analysis

In discrete-time, the reachable set can be written

$$\mathrm{Reach}(\mathbf{f}, \mathbf{X}_0) \;=\; \bigcup_{k \in \mathbb{N}} \mathbf{f}^k(\mathbf{X}_0). \tag{6}$$

In order to compute the reachable set with the unbounded time horizon using a purely numerical algorithm, we need the trajectories of (2,3) starting in $\mathbf{X}_0$ to be bounded in space. To force this property in general, we introduce the *bounded reachable set* as follows: Let $\mathbb{U}$ be a bounded subset of $\mathbb{R}^n$ (called the *universe*), then

$$\mathrm{Reach}_{\mathbb{U}}(\mathbf{f}, \mathbf{X}_0) \;:=\; \bigcup_{k \in \mathbb{N}} (\mathbf{f}_{\mathbb{U}})^k(\mathbf{X}_0 \cap \mathbb{U}), \tag{7}$$

where $\mathbf{f}_{\mathbb{U}}(\mathbf{X}) = \mathbf{f}(\mathbf{X}) \cap \mathbb{U}$.

In general $\mathrm{Reach}_{\mathbb{U}}(\mathbf{f}, \mathbf{X}_0)$ is only a subset of $\mathrm{Reach}(\mathbf{f}, \mathbf{X}_0)$. However, in several situations this relationship can be stronger. When $\mathbb{U}$ happens to be a trapping region that contains $\mathbf{X}_0$ then the equality holds. A weaker condition can be verified a posteriori:

$$\mathbf{f}\big(\mathrm{Reach}_{\mathbb{U}}(\mathbf{f}, \mathbf{X}_0)\big) \subseteq \mathbb{U} \implies \mathrm{Reach}_{\mathbb{U}}(\mathbf{f}, \mathbf{X}_0) = \mathrm{Reach}(\mathbf{f}, \mathbf{X}_0). \tag{8}$$

Also, when every trajectory which leaves $\mathbb{U}$ remains outside $\mathbb{U}$ then $\mathrm{Reach}_{\mathbb{U}}(\mathbf{f}, \mathbf{X}_0) = \mathrm{Reach}(\mathbf{f}, \mathbf{X}_0) \cap \mathbb{U}$. This allows tackling a large class of reachability problems by computing an enclosure of the bounded reachable set. Furthermore, in some situations such a universe is meaningful because physical solutions outside this universe do not exist, and the bounded reachable set is then useful by itself.

In continuous time, the bounded reachable set is

$$\mathrm{Reach}_{\mathbb{U}}(\Phi, \mathbf{X}_0) := \bigcup_{t \in \mathbb{R}^+} \Phi_{\mathbb{U}}(t, \mathbf{X}_0 \cap \mathbb{U}), \tag{9}$$

where $\mathbb{R}^+ = \{x \in \mathbb{R} : x \geq 0\}$ and $\Phi_{\mathbb{U}}$ is defined similarly to $\mathbf{f}_{\mathbb{U}}$:

$$\Phi_{\mathbb{U}}(t, \mathbf{X}_0) := \Big\{\mathbf{y} \in \mathbb{R}^n : \exists \mathbf{x} \in \mathbb{X}, \big(\mathbf{y} = \Phi(t, \mathbf{x}) \wedge \forall t' \leq t, \phi(t, \mathbf{x}) \in \mathbb{U}\big)\Big\}. \tag{10}$$

3.2 Boxes and Enclosure Sets

In order to perform a global analysis of our system in some universe $\mathbb{U}$, we take a finite subset $\mathcal{B}$ of the set of all boxes $\mathbb{B}$ in $\mathbb{R}^n$ such that the elements of $\mathcal{B}$ have disjoint interiors and form a cover of the universe $\mathbb{U} \subset \bigcup \mathcal{B}$. We perform numerical computation of the system evolution on a class of enclosure sets $\mathbb{E}$.

To convert from boxes to enclosure sets, we assume that we have access to a function $\mathrm{Enclose} : \mathbb{B} \to \mathbb{E}$ such that

$$[\mathbf{x}] \subset \mathrm{Enclose}([\mathbf{x}]). \tag{11}$$

In many cases we have $\mathbb{B} \subset \mathbb{E}$, and can take Enclose to be the identity. We also assume we have access to a function $\mathrm{Cover} : \mathbb{E} \times \mathcal{P}(\mathbb{B}) \to \mathcal{P}(\mathbb{B})$ such that $\mathrm{Cover}(\langle x \rangle, \mathcal{B})$ returns a cover of $\langle x \rangle$ by boxes in $\mathcal{B}$. Formally, Cover is required to satisfy

$$\{[\mathbf{u}] \in \mathcal{B} : [\mathbf{u}] \cap \langle \mathbf{x} \rangle \neq \emptyset\} \subseteq \mathrm{Cover}(\langle \mathbf{x} \rangle, \mathcal{B}) \subseteq \mathcal{B}. \tag{12}$$

If $\langle \mathbf{x} \rangle \subset \bigcup \mathcal{B}$, then $\langle \mathbf{x} \rangle \subseteq \mathrm{Cover}(\langle \mathbf{x} \rangle, \mathcal{B})$.Such a function is easily implemented as soon as a sufficient condition for $[\mathbf{u}] \cap E$ is available.

Remark 3.1 If implemented naively, the function Cover has a complexity proportional to the number of elements in $\mathcal{B}$ even for small sets $\langle \mathbf{x} \rangle$. This is impracticable since $\mathcal{B}$ is huge and the function Cover will be intensively called for small sets $\langle \mathbf{x} \rangle$. An indexation of the elements of $\mathcal{B}$, which reduces the complexity of Cover to $\log(\mathrm{card}\, \mathcal{B})$ for small sets $\langle \mathbf{x} \rangle$, is therefore necessary.

3.3 Reachability Algorithm for Discrete Time Dynamical Systems

Let us consider the discrete-time dynamical system (2). The reachable set is given by

$$\mathrm{Reach}(\mathbf{f}, \mathbf{X}_0) = \bigcup_{k \in \mathbb{N}} \mathbf{f}^k(\mathbf{X}_0) \tag{13}$$

We also consider a finite set of boxes $\mathcal{B}$ whose union [4] $\bigcup \mathcal{B}$ will play the role of the universe. These boxes represent the discretization of space, and hence need to be taken sufficiently small. On the other hand, the smaller the more expensive will be the computation on this discretization. The reachability algorithm is given in Algorithm 1.

Algorithm 1 In: $\mathbf{f}$, $\mathbf{X}_0$, $\mathcal{B}$ Out: $\mathcal{R} = \mathrm{RA}(\mathbf{f}, \mathbf{X}_0, \mathcal{B})$

1: $\mathcal{I} \leftarrow \mathrm{Cover}(\mathbf{X}_0, \mathcal{B})$
2: $\mathcal{R} \leftarrow \mathcal{I}$ and $\mathcal{P} \leftarrow \emptyset$
3: `while` $\mathcal{R} \backslash \mathcal{P} \neq \emptyset$
4: $[\mathbf{x}] \leftarrow \mathrm{Element}(\mathcal{R} \backslash \mathcal{P})$
5: $\langle \mathbf{x} \rangle \leftarrow \mathrm{Enclose}([\mathbf{x}])$
6: $\langle \mathbf{y} \rangle \leftarrow \langle \mathbf{f} \rangle (\langle \mathbf{x} \rangle)$
7: $\mathcal{R} \leftarrow \mathcal{R} \cup \mathrm{Cover}(\langle \mathbf{y} \rangle, \mathcal{B})$
8: $\mathcal{P} \leftarrow \mathcal{P} \cup \{[\mathbf{x}]\}$
9: `end while`
10: `return` $\mathcal{R}$

Informally, the algorithm works as follows. The initial condition $\mathbf{X}_0$ is discretized as a union of boxes $\mathcal{I}$ from $\mathcal{B}$ i.e. $\mathbf{X}_0 \subset \bigcup \mathcal{I}$ and $\mathcal{I} \subseteq \mathcal{B}$. The set $\mathcal{I}$ is computed from the initial set $\mathbf{X}_0$ using a function $\mathrm{Cover} : \mathcal{P}(\mathbb{R}^n) \times \mathcal{P}(\mathbb{B}) \to \mathcal{P}(\mathbb{B})$. The over-approximation to the (bounded) reachable set computed by the reachability algorithm is also union of boxes $\mathcal{R}$ from $\mathcal{B}$. The principle of the algorithm is simple: $\mathcal{R}$ contains all the boxes that have been reached so far. These boxes needs to be propagated computing their image. However, the key point is to note that if a box has already been propagated, which means that its image is already covered by the boxes of $\mathcal{R}$, then it is useless to propagate it again. Therefore, the reachability algorithm maintains a list of boxes $\mathcal{P}$ which contains the boxes that have already been propagated. So, the boxes that actually remain to be propagated are the boxes of $\mathcal{R} \backslash \mathcal{P}$, and the algorithm stops when $\mathcal{R} \backslash \mathcal{P} = \emptyset$, which means that no more box has to be propagated. The propagation itself is performed in Lines 5-7.

Remark 3.2 The set difference $\mathcal{R} \backslash \mathcal{P}$ at Line 3 can be expensive to compute when $\mathcal{R}$ and $\mathcal{P}$ are large. In practice, the algorithm is slightly modified to prevent computing this set difference by updating incrementally a set of boxes which remains equal to $\mathcal{R} \backslash \mathcal{P}$.

[4] The union of a set of boxes contains every vector which belong to at least one of these boxes, e.g. $\cup\{[0,1],[2,4],[3,5]\} = \{x \in \mathbb{R} : x \in [0,1] \vee x \in [2,5]\}$.

Remark 3.3 The same algorithm can be used to compute the reachable set for a multivalued map $\mathbf{F} : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$, as long as an enclosure extension $\langle \mathbf{F} \rangle : \mathbb{E} \rightrightarrows \mathbb{E}$ satisfying $\mathbf{F}(\langle x \rangle) \subset \bigcup \big(\langle \mathbf{F} \rangle (\langle x \rangle) \big)$ is available.

3.4 Properties of the Reachability Algorithm

In this section, Algorithm 1 is first prove to halt, and then to be correct.

Proposition 3.4 (Halting) *The Reachability Algorithm 1 halts after a finite number of executions of the while-loop.*

Proof. Using a simple induction, $\mathcal{P}$ starts empty (Line 2) and is added elements of $\mathcal{R}$ at Line 8 (because $[\mathbf{x}] \in \mathcal{R}$ in Line 4). Therefore, $\mathcal{P} \subseteq \mathcal{R}$ holds during the whole execution. Now, $\mathrm{Cover}(\langle \mathbf{y} \rangle, \mathcal{B}) \subseteq \mathcal{B}$ as a consequence of (12). A simple induction using Line 2 and Line 8 thus shows that $\mathcal{R} \subseteq \mathcal{B}$ during the whole execution. Therefore $\mathcal{P} \subseteq \mathcal{B}$ holds during the whole execution. Finally, the cardinality of $\mathcal{P}$ increases of one at each execution of the while-loop, but is bounded by the cardinality of $\mathcal{B}$. Therefore, $\mathcal{P}$ reaches its limit in a finite number of executions of the while-loop. Then, $\mathcal{R} \backslash \mathcal{P}$ has to be empty, otherwise $\mathcal{P}$ would increase more. The while-loop halts at this moment. □

The following Lemma will be used in the proof of correctness of Algorithm 1.

Lemma 3.5 *If* $\mathbf{f}_{\mathbb{U}}(\mathbf{X}) \subseteq \mathbf{X}$ *holds then* $\mathrm{Reach}_{\mathbb{U}}(\mathbf{f}, \mathbf{X}) \subseteq \mathbf{X}$.

Proof. By definition of the bounded reachable set (7), it is sufficient to prove that for all $k \in \mathbb{N}$ the inclusion $(\mathbf{f}_{\mathbb{U}})^k(\mathbf{X} \cap \mathbb{U}) \subseteq \mathbf{X}$ holds. Let us prove this by induction. For $k = 0$, $(\mathbf{f}_{\mathbb{U}})^0(\mathbf{X} \cap \mathbb{U}) = \mathbf{X} \cap \mathbb{U} \subseteq \mathbf{X}$. Now, let us suppose that $(\mathbf{f}_{\mathbb{U}})^k(\mathbf{X} \cap \mathbb{U}) \subseteq \mathbf{X}$ holds. Then $(\mathbf{f}_{\mathbb{U}})^{k+1}(\mathbf{X} \cap \mathbb{U}) = \mathbf{f}_{\mathbb{U}}((\mathbf{f}_{\mathbb{U}})^k(\mathbf{X} \cap \mathbb{U})) \subseteq \mathbf{f}_{\mathbb{U}}(\mathbf{X}) \subseteq \mathbf{X}$, which concludes the proof. □

Proposition 3.6 (Correctness) *If* $\mathbb{U} \subset \bigcup \mathcal{B}$, *then the following inclusion holds:*

$$\mathrm{Reach}_{\mathbb{U}}(\mathbf{f}, \mathbf{X}_0) \;\subseteq\; \bigcup \mathrm{RA}(\mathbf{f}, \mathbf{X}_0, \mathcal{B}). \tag{14}$$

Proof. First, let us prove by induction that

$$\Big(\mathbf{x} \in (\bigcup \mathcal{P}) \wedge \mathbf{f}(\mathbf{x}) \in (\bigcup \mathcal{B}) \Big) \implies \mathbf{f}(\mathbf{x}) \in (\bigcup \mathcal{R}) \tag{15}$$

holds during the whole execution of the algorithm. Entering the first time in the while-loop, $\mathcal{P}$ is empty and hence (15) trivially holds. Now suppose that (15) holds when entering the while-loop. Then leaving the while-loop, $\mathcal{P}$ has been added the box $[\mathbf{x}]$. But in the same time, $\mathcal{R}$ has been added $\mathrm{Cover}(\mathbf{f}, [\mathbf{x}], \mathcal{B})$, and therefore (15) holds when finishing the while-loop.

Now, when the algorithm halts $\mathcal{P} = \mathcal{R}$ and thus (15) gives rise to

$$\Big(\mathbf{x} \in (\bigcup \mathcal{R}) \wedge \mathbf{f}(\mathbf{x}) \in (\bigcup \mathcal{B}) \Big) \implies \mathbf{f}(\mathbf{x}) \in (\bigcup \mathcal{R}). \tag{16}$$

This reads $\mathbf{f}_{\bigcup\mathcal{B}}(\bigcup\mathcal{R}) \subseteq (\bigcup\mathcal{R})$. Therefore, Lemma 3.5 can be applied to prove that $\mathrm{Reach}_{\mathbb{U}}(\mathbf{f}, \bigcup\mathcal{R}) \subseteq (\bigcup\mathcal{R})$ holds. Finally, as $\mathbf{X}_0 \cap \mathbb{U} \subseteq \bigcup\mathcal{I} \subseteq \bigcup\mathcal{R}$ the inclusion $\mathrm{Reach}_{\mathbb{U}}(\mathbf{f}, \bigcup\mathcal{I}) \subseteq \mathrm{Reach}_{\mathbb{U}}(\mathbf{f}, \bigcup\mathcal{R})$ holds, which concludes the proof. □

In [3,4], it is shown that any algorithm which relies on numerical approximations cannot in general compute arbitrarily accurate over-approximations to the reachable set itself, but only convergence to the *chain-reachable set* can be achieved. It can be shown that the result of Algorithm 1 converges to the chain reachable set as the maximum width of the boxes of $\mathcal{B}$ converges to zero, assuming convergence conditions on the enclosure extension $\langle\mathbf{f}\rangle$ and the function Enclose and Cover. The experiments presented in Section 4 show sharp enclosures of the bounded reachable sets.

3.5 The Reach-and-Evolve Algorithm for Discrete Time Dynamical Systems

We now consider the R&E algorithm for systems in discrete time. The main problem with the basic reachability algorithm is that at each step, we perform an over-approximation of the enclosure set on a grid. This operation introduces zero-order errors, and hence causes a greater loss of accuracy than the higher-order enclosure computation. To reduce the errors, we can modify the algorithm so that we only perform an over-approximation on a grid every M steps. We call the new algorithm the *discrete-time reach-evolve algorithm.*

Fix $M > 0$ and define the *evolution step* operator $\mathbf{e}_M : \mathbb{R}^n \to \mathbb{R}^n$ and the *reachability step* operator $\mathbf{r}_M : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ by

$$\mathbf{e}_M(\mathbf{x}) := \mathbf{f}^M(\mathbf{x}); \quad \mathbf{r}_M(\mathbf{x}) := \bigcup_{m=0}^{M-1} \mathbf{f}^m(\mathbf{x}). \tag{17}$$

The following elementary proposition shows how to re-write the reachability algorithm (7).

Proposition 3.7

$$\mathrm{Reach}(\mathbf{f}, \mathbf{X}_0) \;=\; \bigcup_{k\in\mathbb{N}} \mathbf{e}_M^k(\mathbf{r}_M(\mathbf{X}_0)) = \mathrm{Reach}(\mathbf{e}_M, \mathbf{r}_M(\mathbf{X}_0)). \tag{18}$$

We can therefore use Algorithm 1 to compute the bounded reachable set $\mathrm{Reach}_{\bigcup\mathcal{B}}(\mathbf{f}, \Phi)$ as follows. First, a set of boxes $\mathcal{I}$ is computed such that $\bigcup\mathcal{I} \supseteq \mathbf{X}_0 \cap \mathbb{U}$ (this is easily done using the function Cover). Next, an over-approximation of $\mathcal{H} \bigcup_{m=0}^{M-1} \mathbf{f}^m(\bigcup\mathcal{I})$ is computed using en enclosure extension $\langle\mathbf{r}_M\rangle$ of $\mathbf{r}_M$. Finally, Algorithm 1 is applied to $\mathbf{e}_M$, $\bigcup\mathcal{H}$ and $\mathcal{B}$, returning a superset of $\mathrm{Reach}_{\mathbb{U}}(\mathbf{f}, \mathbf{X}_0)$ by Proposition 3.9. This yield the Reach-and-Evolve Algorithm 2.
The "reach" part of the algorithm is in Lines 3-9, and the "evolve" part in Lines 10-16.

Remark 3.8 By increasing M, the accuracy of the algorithm is increased, since the number of evolution steps between each over-approximation on the grid is large.

Algorithm 2 In: $\mathbf{f}$, $\mathbf{X}_0$, $\mathcal{B}$, M Out: $\mathcal{R} = \mathrm{RE}(\mathbf{f}, \mathbf{X}_0, \mathcal{B}, M)$

1: $\mathcal{I} \leftarrow \mathrm{Cover}(\mathbf{X}_0, \mathcal{B})$
2: $\mathcal{R} \leftarrow \emptyset$ and $\mathcal{P} \leftarrow \emptyset$
3: `for` $[\mathbf{x}]$ `in` $\mathcal{I}$
4: $\quad\langle \mathbf{x}_0 \rangle \leftarrow \mathrm{Enclose}([\mathbf{x}])$
5: $\quad$`for` $m = 0$ `to` $M - 1$
6: $\quad\quad\mathcal{R} \leftarrow \mathcal{R} \cup \mathrm{Cover}(\langle \mathbf{x}_m \rangle, \mathcal{B})$
7: $\quad\quad\langle \mathbf{x}_{m+1} \rangle \leftarrow \langle \mathbf{f} \rangle(\langle \mathbf{x}_m \rangle)$
8: $\quad$`end for`
9: `end for`
10: `while` $\mathcal{R} \backslash \mathcal{P} \neq \emptyset$
11: $\quad[\mathbf{x}] \leftarrow \mathrm{Element}(\mathcal{R} \backslash \mathcal{P})$
12: $\quad\langle \mathbf{x} \rangle \leftarrow \mathrm{Enclose}([\mathbf{x}])$
13: $\quad\langle \mathbf{y} \rangle \leftarrow \langle \mathbf{f} \rangle^M(\langle \mathbf{x} \rangle)$
14: $\quad\mathcal{R} \leftarrow \mathcal{R} \cup \mathrm{Cover}(\langle \mathbf{y} \rangle, \mathcal{B})$
15: $\quad\mathcal{P} \leftarrow \mathcal{P} \cup \{[\mathbf{x}]\}$
16: `end while`
17: `return` $\mathcal{R}$

However, the algorithm will essentially evolve many times over the same part of state space, which is computationally more expensive. We expect the complexity of the algorithm to be roughly proportional to M. However, decreasing the size of the state-space grid can be even more expensive, typically exponential in the dimension d.

3.6 The Reach-and-Evolve Algorithm for Continuous Time Dynamical Systems

We now consider the R&E algorithm for systems in continuous time. Let $\Phi(t, \mathbf{x})$ be the solution operator of the ODE (3). The reachable set can then be written as

$$\mathrm{Reach}(\Phi, \mathbf{X}_0) := \bigcup_{t \in \mathbb{R}^+} \Phi(t, \mathbf{X}_0). \tag{19}$$

Fix $h > 0$ and define the *evolution step* operator $\mathbf{e}_h : \mathbb{R}^n \to \mathbb{R}^n$ and the *reachability step* operator $\mathbf{r}_h : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ by

$$\mathbf{e}_h(\mathbf{x}) := \Phi(h, \mathbf{x}); \quad \mathbf{r}_h(\mathbf{x}) := \Phi([0, h], \mathbf{x}) \tag{20}$$

The following proposition shows how to discretize rigorously time in order to apply the reachability algorithm to enclose (9).

Proposition 3.9

$$\mathrm{Reach}(\Phi, \mathbf{X}_0) = \bigcup_{k \in \mathbb{N}} \mathbf{e}_h^k(\mathbf{r}_h(\mathbf{X}_0)) = \mathrm{Reach}(\mathbf{e}_h, \mathbf{r}_h(\mathbf{X}_0)). \tag{21}$$

We can therefore use Algorithm 1 to compute the bounded reachable set $\mathrm{Reach}_{\bigcup \mathcal{B}}(\mathbf{f}, \Phi)$ as follows. First, a set of boxes $\mathcal{I}$ is computed such that $\bigcup \mathcal{I} \supseteq \mathbf{X}_0 \cap \mathbb{U}$ (this is easily done using the function Cover). Next, an over-approximation $\mathcal{H}$ of $\Phi([0, h], \bigcup \mathcal{I})$ is computed using en enclosure extension $\langle \mathbf{r}_h \rangle$ of $\mathbf{r}_h$. Finally, Algorithm 1 is applied to $\mathbf{e}_h$, $\bigcup \mathcal{H}$ and $\mathcal{B}$, returning a superset of $\mathrm{Reach}_{\mathbb{U}}(\Phi, \mathbf{X}_0)$ by Proposition 3.9. This yield the Reach-and-Evolve Algorithm 3.
The "reach" part of the algorithm is in Lines 3-7, and the "evolve" part in Lines 8-14.

Algorithm 3 In: $\mathbf{f}$, $\mathbf{X}_0$, $\mathcal{B}$, h Out: $\mathcal{R} = \mathrm{RE}(\mathbf{f}, \mathbf{X}_0, \mathcal{B}, h)$

1: $\mathcal{I} \leftarrow \mathrm{Cover}(\mathbf{X}_0, \mathcal{B})$
2: $\mathcal{R} \leftarrow \emptyset$ and $\mathcal{P} \leftarrow \emptyset$
3: `for` $[\mathbf{x}]$ `in` $\mathcal{I}$
4: $\quad \langle \mathbf{x} \rangle \leftarrow \mathrm{Enclose}([\mathbf{x}])$
5: $\quad \langle \mathbf{y} \rangle \leftarrow \langle \mathbf{r}_h \rangle(\langle \mathbf{x} \rangle)$
6: $\quad \mathcal{R} \leftarrow \mathcal{R} \cup \mathrm{Cover}(\langle \mathbf{y} \rangle, \mathcal{B})$
7: `end for`
8: `while` $\mathcal{R} \backslash \mathcal{P} \neq \emptyset$
9: $\quad [\mathbf{x}] \leftarrow \mathrm{Element}(\mathcal{R} \backslash \mathcal{P})$
10: $\quad \langle \mathbf{x} \rangle \leftarrow \mathrm{Enclose}([\mathbf{x}])$
11: $\quad \langle \mathbf{y} \rangle \leftarrow \langle \mathbf{e}_h \rangle(\langle \mathbf{x} \rangle)$
12: $\quad \mathcal{R} \leftarrow \mathcal{R} \cup \mathrm{Cover}(\langle \mathbf{y} \rangle, \mathcal{B})$
13: $\quad \mathcal{P} \leftarrow \mathcal{P} \cup \{[\mathbf{x}]\}$
14: `end while`
15: `return` $\mathcal{R}$

Remark 3.10 A naïve way of computing the reachable set is to use the formula $\mathrm{Reach}(\mathbf{f}, \mathbf{X}_0) = \bigcup_{k \in \mathbb{N}} \mathbf{r}_h^k(\mathbf{X}_0)$ and to apply Algorithm 1 to $\mathbf{r}_h$. This approach fails because of the inherent over-approximation involved in computing $\mathrm{Cover}(\mathbf{r}_h(\mathrm{Enclose}([\mathbf{x}])), \mathcal{B})$. Note that $\mathbf{x} \in \Phi([0, h], \mathbf{x})$ for all x. Hence the over-approximation of $\mathbf{r}_h([x])$ is likely to contain all boxes touching $[x]$. Iteration of this operator will therefore ultimately result in *all* boxes being found!

Remark 3.11 An alternative way to compute the reachable set is to use the formula $\mathrm{Reach}(\mathbf{f}, \mathbf{X}_0) = \mathbf{r}_h\big(\bigcup_{k \in \mathbb{N}} \mathbf{e}_h^k(\mathbf{X}_0)\big)$. However, the ordering (21) used in Algorithm 3 appears to be more efficient in practice.

4 Numerical Experiments

We report experiments on two non linear dynamical systems. Algorithm 1 has been implemented in C++, using the interval library PROFIL/BIAS [9], and ran on an Intel Dual Core 2.2Ghz processor.

In all cases, the universe is a box. This box is covered by a set of boxes $\mathcal{B}$, which is used in the algorithm. The initial conditions are disks. In order to be used in the algorithm, these disks are covered using boxes of $\mathcal{B}$, resulting in a list of box $\mathcal{I}$ whose union contains the initial condition. The enclosure of the bounded reachable set computed by the reachability algorithm is a subset $\mathcal{R}$ of $\mathcal{B}$. However, the number of boxes in $\mathcal{R}$ is too big to obtain nice graphical representations. Therefore a post-processing that reduces the number of boxes is performed .

4.1 The Hénon Map

The Hénon map is defined by

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} x_2 + 1 - a x_1^2 \\ b x_1 \end{pmatrix}. \tag{22}$$

We use here the standard parameter values $a = 1.4$ and $b = 0.3$ for which the dynamical system is chaotic. The universe is chosen to be $\mathbb{U} = \{\mathbf{x} \in \mathbb{R}^2 : -2 \leq x_1 \leq 2, -2 \leq x_2 \leq 2\}$. The bounded reachable set computed by the reachability algorithm is shown in Figure 1, the initial condition being the disk centered at

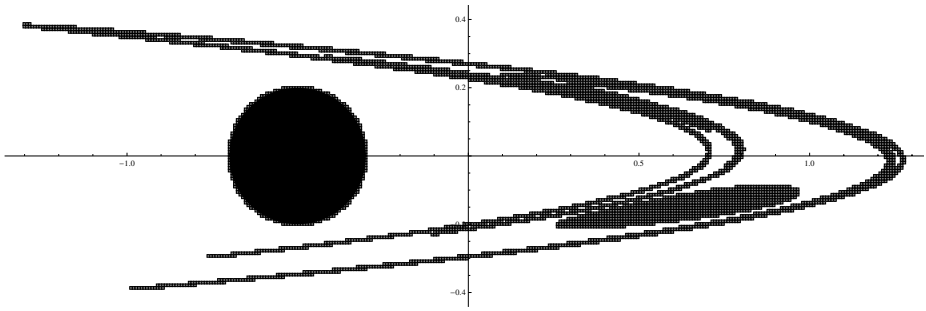

Fig. 1. Enclosure of the bounded reachable set (gray boxes) for the Hénon map and the black disk as initial condition.

$(-0.5, 0)$ of radius 0.2. It has been computed in 7.5 seconds. Furthermore, the initial condition is inside the well known trapping region of the Hénon map, and therefore the bounded reachable set is equal to the reachable set.

4.2 Continuous Time Dynamical System with a Limit Cycle

The ODE $\mathbf{x}'(t) = \mathbf{f}(\mathbf{x}(t))$ with

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} x_1 - x_2 - x_1(x_1^2 + x_2^2) \\ x_1 + x_2 - x_2(x_1^2 + x_2^2) \end{pmatrix} \tag{23}$$

has an attracting cycle. It is interesting to verify that the R&E algorithm allows separating the two areas of the universe that are separated by this cycle. The bounded reachable sets computed by the R&E algorithm are shown in Figure 2 for two initial conditions (the disks centered at $(1.5, 1.5)$ and $(0, 0)$ respectively, and of radius 0.2). The universe is $\mathbb{U} = \{\mathbf{x} \in \mathbb{R}^2 : -2 \leq x_1 \leq 2, -2 \leq x_2 \leq 2\}$ and the step size used for discretization of time is $h = 0.005$. They both have been computed in 250 seconds. Furthermore, we can again check a posteriori that $\Phi(h, \bigcup \mathcal{R}) \subseteq (\bigcup \mathcal{B})$, which implies that the bounded reachable is is equal to the reachable set. In both cases, the R&E algorithm is able to separate the two area delimited by the attracting cycle. Up to the authors knowledge, the R&E algorithm is the first numerical algorithm which is able to prove rigorously this separation for such a nonlinear dynamical system.

4.3 Comparison with HSolver

HSolver [18,19] is dedicated to the safety analysis for nonlinear hybrid systems. This include ODE driven dynamical systems. In this case, HSolver also tackles bounded reachable sets. It is based on interval analysis and constraint propagation. Although HSolver tackles a larger class of problem than the R&E algorithm presented in this paper, the following example shows that it lacks efficiency for the reachability analysis of ODE driven dynamical systems.

Let us consider the linear ODE $\mathbf{x}'(t) = \mathbf{x}(t)$, the universe $\mathbb{U} = \{\mathbf{x} \in \mathbb{R}^2 : 0 \leq x_1 \leq 10, 0 \leq x_2 \leq 10\}$ and the initial condition $\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^2 : 1 \leq x_1 \leq 10, 0 \leq x_2 \leq 1\}$.

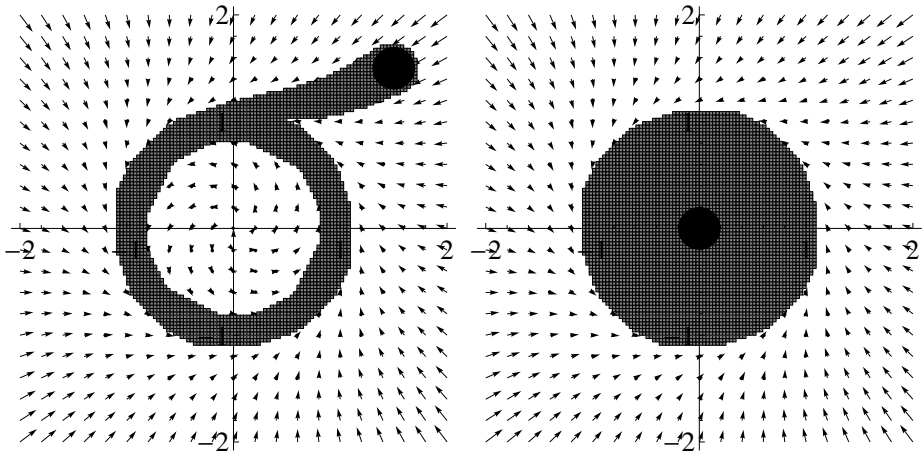

Fig. 2. Enclosure of the bounded reachable set (gray boxes) for an ODE with an attracting cycle, and two different initial conditions (displayed in black).

We wish to decide whether the vector $(7, 10)$ is inside the bounded reachable set or not. Because of the simplicity of the system, it is easy to see that it is not. However, HSolver fails to prove this property after 10 minutes of computations [5]. The enclosure of the bounded reachable set computed by the R&E algorithm in 36 seconds is represented in Figure 3 with a step size $h = 0.1$. It clearly shows that $(7, 10)$ is not reachable inside the universe $\mathbb{U}$. Note that the enclosure shown on Figure 3 has been computed using the simple integrator described in Appendix A. The usage of a more sophisticated interval integrator should improve a lot the enclosure, while HSolver cannot benefit of the usage of higher order Taylor expansions.

5 Conclusion

Dealing with an infinite time horizon within a purely numerical algorithm is difficult, but necessary for enclosing rigorously reachable sets of nonlinear dynamical systems. In this paper, we introduced the bounded reachable set, which forces the space variables to be bounded. Once space is bounded, the infinite time horizon can be handled rigorously through the Reach and Evolve algorithm proposed in this paper. The inclusion of the reachable set inside the bounded reachable set in several situations allows tackling a large variety of reachability problems.

Presented experiments have shown that the R&E algorithm is able to compute accurate enclosures of the reachable set for non-trivial nonlinear dynamical systems. It has also been compared to Stefan Ratschan's HSolver, and shows a much better behavior. It can be noted that on the one hand HSolver is dedicated to a wider class of hybrid problems. On the other hand, the R&E algorithm will directly benefit of the usage of high order Taylor expansion of flows while HSolver's method

[5] Stefan Ratschan has pointed out in a personal communication that the improvements presented in [17] to HSolver can solve this linear problem. However, these improvements are restricted to linear ODE.

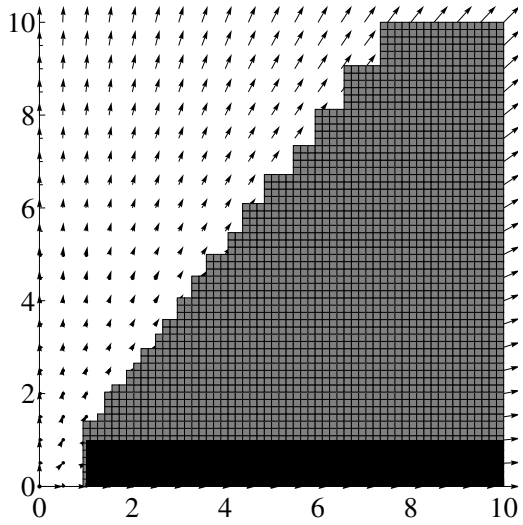

Fig. 3. Enclosure of the bounded reachable set (gray boxes) for a linear ODE, and the black rectangle as initial condition.

is intrinsically restricted to first order integration.

Future works include using higher order Taylor expansions. Although we of course expect drastic improvements, this also raises several issues, like tuning the step size, the space discretization size and the order of the Taylor expansion. Furthermore, the R&E algorithm will be extended to hybrid systems. A simple version of the algorithm has been implemented in the software package Ariadne [2].

References

[1] Alefeld, G. and J. Herzberger, "Introduction to Interval Computations," Computer Science and Applied Mathematics, 1974.

[2] Benvenuti, L., D. Bresolin, A. Casagrande, P. Collins, A. Ferrari, E. Mazzi, A. Sangiovanni-Vincentelli and T. Villa, *Reachability computation for hybrid systems with ariadne*, in: *Proceedings of the 17th IFAC World Congress*, 2008.

[3] Collins, P., *Continuity and computability of reachable sets*, Theor. Comput. Sci. **341** (2005), pp. 162–195.

[4] Collins, P., *Optimal semicomputable approximations to reachable and invariant sets*, Theory Comput. Syst. **41** (2007), pp. 33–48.

[5] Goldberg, D., *What every computer scientist should know about floating-point arithmetic*, Computing Surveys **23** (1991), pp. 5–48.

[6] Goldsztejn, A. and W. Hayes, *Reliable Inner Approximation of the Solution Set to Initial Value Problems with Uncertain Initial Value*, in: *Proceedings of SCAN 2006*.

[7] Hayes, B., *A Lucid Interval*, American Scientist **91** (2003), pp. 484–488.

[8] Jaulin, L., M. Kieffer, O. Didrit and E. Walter, "Applied Interval Analysis with Examples in Parameter and State Estimation, Robust Control and Robotics," Springer-Verlag, 2001.

[9] Knueppel, O., *PROFIL/BIAS - A Fast Interval Library*, Computing **53** (1994), pp. 277–287.

[10] Kühn, W., *Rigorously Computed Orbits of Dynamical Systems Without the Wrapping Effect*, Computing **61** (1998), pp. 47–67.

[11] Kurzhanski, A. B. and P. Varaiya, *On ellipsoidal techniques for reachability analysis. I. External approximations*, Optim. Methods Softw. **17** (2002), pp. 177–206.

[12] Lohner, R., *Enclosing the solutions of ordinary initial and boundary value problems*, Computer Arithmetic: Scientific Computation and Programming Languages, Wiley-Teubner Series in Computer Science, Stuttgart (1987), pp. 255–286.

[13] Makino, K. and M. Berz, *Suppression of the Wrapping Effect by Taylor Model-based Verified Integrators: Long-term Stabilization by Preconditioning*, International Journal of Differential Equations and Applications **10** (2005), pp. 353–384.

[14] Moore, R., "Interval Analysis," Prentice-Hall, 1966.

[15] Nedialkov, N. S., K. R. Jackson and G. F. Corliss, *Validated Solutions of Initial Value Problems for Ordinary Differential Equations*, Applied Mathematics and Computation **105** (1999), pp. 21–68.

[16] Neumaier, A., "Interval Methods for Systems of Equations," Cambridge Univ. Press, 1990.

[17] Ratschan, S. and Z. She, *Constraints for continuous reachability in the verification of hybrid systems*, in: *Proc. 8th Int. Conf. on Artif. Intell. and Symb. Comp., AISC'2006*, number 4120 in LNCS (2006), pp. 196–210.

[18] Ratschan, S. and Z. She, *Safety verification of hybrid systems by constraint propagation based abstraction refinement*, ACM Transactions in Embedded Computing Systems **6** (2007).

[19] Ratschan, S. and Z. She, *Recursive and backward reasoning in the verification on hybrid systems*, in: *Proceedings of the 5th Int. Conf. on Informatics in Control, Automation and Robotics*, 2008.

[20] Rump, S. M., *INTLAB - Interval Laboratory*, in: T. Csendes, editor, *Developments in Reliable Computing*, Kluwer, 1999 p. 77104.

[21] *Wolfram Research inc., Mathematica*, Champaign, Illinois (2007).

[22] Zgliczynski, P., *C^1-Lohner Algorithm*, Foundations of Computational Mathematics **2** (2002), pp. 429–465.

A A First-Order Interval Integrator

The aim of this section is to describe a very simple first order interval integrator for ODE. We consider an ODE $\mathbf{x}'(t) = \mathbf{f}(\mathbf{x}(t))$ and a set of initial conditions $[\mathbf{x}]$. The aim of the rigorous integration is to compute a box $[\mathbf{y}]$ which contains $\Phi([h], [\mathbf{x}])$ for a given interval $[h] \geq 0$, that is, which contains all states reached starting in $[\mathbf{x}]$ and evolving for a duration $h \in [h]$ (note that the interval $[h]$ can be degenerated, i.e $\underline{h} = \overline{h}$). An interval integration is usually performed in two steps. The first consists of computing a crude enclosure of $\Phi([0, \overline{h}], [\mathbf{x}])$. Then this crude enclosure is used to compute a sharper enclosure of $\Phi([h], [\mathbf{x}])$. These computations are based on the first order Taylor expansion of the solution of the ODE:

$$\mathbf{x}(h) = \mathbf{x}(0) + h\,\mathbf{f}(\mathbf{x}(0)) + \mathbf{r}, \tag{A.1}$$

where the remainder $\mathbf{r} = (r_1 \ldots, r_n)$ is

$$r_i = \frac{h^2}{2}\,(\nabla f_i)(\mathbf{x}(\xi_i)) \cdot \mathbf{f}(\mathbf{x}(\xi_i)) \tag{A.2}$$

with $\xi_i \in [0, h]$ for $i \in \{1, \ldots n\}$.

A.1 Crude Enclosure

In this section, we show how to build a box $[\mathbf{x}_C]$ which contains $\Phi([0, \overline{h}], [\mathbf{x}])$. This crude enclosure computation relies on the following idea: Let us first suppose that $[\mathbf{x}_C]$ is such an enclosure. Then by (A.1)

$$[\mathbf{x}'_C] := [\mathbf{x}] + [0, \overline{h}]\,[\mathbf{f}]([\mathbf{x}]) + \frac{[0, \overline{h}]^2}{2}\,[D\mathbf{f}]([\mathbf{x}_C]) \cdot \mathbf{f}([\mathbf{x}_C]) \tag{A.3}$$

is also an enclosure of $\Phi([0, \overline{h}], [\mathbf{x}])$. If the inclusion $[\mathbf{x}'_C] \subseteq [\mathbf{x}_C]$ does hold, then this confirms the hypothesis that $[\mathbf{x}_C]$ encloses $\Phi([0, \overline{h}], [\mathbf{x}])$. This is due to the Picard operator which can be proved to be contracting because of the inclusion $[\mathbf{x}'_C] \subseteq [\mathbf{x}_C]$. If $\overline{h}$ is small enough then we expect the fixed point iteration

$$[\mathbf{x}_C] \leftarrow [\mathbf{x}] + [0, \overline{h}]\,[\mathbf{f}]([\mathbf{x}]) + \frac{[0, \overline{h}]^2}{2}\,[D\mathbf{f}]([\mathbf{x}_C]) \cdot \mathbf{f}([\mathbf{x}_C]) \tag{A.4}$$

to be contracting, its limit satisfying the equality, and hence the wanted inclusion. However, finite precision computations prevent from computing the exact limit and an inflation process has to be interleaved with the fixed point computation (A.4). We obtain Algorithm 4 for the computation of a crude enclosure of $\Phi([0, \overline{h}], [\mathbf{x}])$.

Note that Line 5 simply performs a 1% inflation of the box $[\mathbf{x}'_C]$. If Algorithm 4 returns the emptyset then the crude enclosure process has failed. Otherwise, it returns an enclosure of $\Phi([0, \overline{h}], [\mathbf{x}])$.

Algorithm 4 *In:* $\mathbf{f}$, $[h]$, $[\mathbf{x}]$ *Out:* $[\mathbf{x}_C]$
1: $[\mathbf{x}'_C] \leftarrow [\mathbf{x}]$
2: $k_{max} \leftarrow 10$
3: $k \leftarrow 0$
4: Repeat
5: $\quad [\mathbf{x}_C] \leftarrow \hat{\mathbf{x}}'_C + 1.01([\mathbf{x}'_C] - \hat{\mathbf{x}}'_C)$ /⋆ $\hat{\mathbf{x}}'_C$ *is the midpoint of* $[\mathbf{x}'_C]$ ⋆/
6: $\quad [\mathbf{x}'_C] \leftarrow [\mathbf{x}] + [0, \overline{h}]\,[\mathbf{f}]([\mathbf{x}]) + \frac{[0,\overline{h}]^2}{2}\,[D\mathbf{f}]([\mathbf{x}_C]) \cdot \mathbf{f}([\mathbf{x}_C])$
7: While(not$([\mathbf{x}'_C] \subseteq [\mathbf{x}_C]) \wedge k \leq k_{max}$)
8: If$([\mathbf{x}'_C] \subseteq [\mathbf{x}_C])$ Return$([\mathbf{x}'_C])$ Else Return$(\emptyset)$

A.2 Sharper Enclosure

Once a crude enclosure $[\mathbf{x}_C]$ of $\Phi([0, \overline{h}], [\mathbf{x}])$ has been computed using Algorithm 4 a sharper enclosure $[\mathbf{y}]$ of $\Phi([h], [\mathbf{x}])$ is easily obtained in the following way:

$$[\mathbf{y}] = [\mathbf{x}] + [h]\,[\mathbf{f}]([\mathbf{x}]) + \frac{[h]^2}{2}\,[D\mathbf{f}]([\mathbf{x}_C]) \cdot \mathbf{f}([\mathbf{x}_C]). \tag{A.5}$$

When the crude enclosure process fails, the normal action is to reduce h and try again. However, this step size reduction is not implemented in the presented experiments. The initial step size was chosen small enough to obtain the success of the crude enclosure process everywhere in the universe.

B Extra Proofs

Proof. [Proposition 3.9] Note that $\Phi([0, h], \mathbf{X}_0) = \bigcup_{t\in[0,h]} \Phi(t, \mathbf{X}_0)$. Hence,

$$\mathrm{Reach}(\mathbf{f}, \Phi([0, h], \mathbf{X}_0)) = \mathrm{Reach}(\mathbf{f}, \bigcup_{t\in[0,h]} \Phi(t, \mathbf{X}_0)). \tag{B.1}$$

Now, since in general $\mathrm{Reach}(\mathbf{f}, \mathbf{A} \cup \mathbf{B}) = \mathrm{Reach}(\mathbf{f}, \mathbf{A}) \bigcup \mathrm{Reach}(\mathbf{f}, \mathbf{B})$,

$$\mathrm{Reach}(\mathbf{f}, \Phi([0, h], \mathbf{X}_0)) = \bigcup_{t\in[0,h]} \mathrm{Reach}(\mathbf{f}, \Phi(t, \mathbf{X}_0)). \tag{B.2}$$

By definition of the reachable set,

$$\mathrm{Reach}(\mathbf{f}, \Phi([0, h], \mathbf{X}_0)) = \bigcup_{k\in\mathbb{N},\ t\in[0,h]} \mathbf{f}^k(\Phi(t, \mathbf{X}_0)). \tag{B.3}$$

As Φ is an ODE solution operator, it satisfies $\Phi(t', \Phi(t'', \mathbf{x})) = \Phi(t' + t'', \mathbf{x})$. Therefore

$$\mathrm{Reach}(\mathbf{f}, \Phi([0, h], \mathbf{X}_0)) = \bigcup_{k\in\mathbb{N},\ t\in[0,h]} \Phi(hk, \Phi(t, \mathbf{X}_0)) = \bigcup_{k\in\mathbb{N},\ t\in[0,h]} \Phi(hk + t, \mathbf{X}_0). \tag{B.4}$$

This latter is obviously equal to $\mathrm{Reach}(\Phi, \mathbf{X}_0)$. □