Template-Based Computation of Barrier Certificates of Continuous Dynamical Systems using Interval Constraints

Adel Djaballah

ENSTA ParisTech, U2IS
joint work with Olivier Bouissou(CEA), Alexandre Chapoutot(ENSTA) and Michel Kieffer(Supelec)

Reunion of GT MEA 14/11/2013

1 Context

2 Barrier certificate

3 Approach

4 Examples

5 Conclusion and future work

1 Context

2 Barrier certificate

3 Approach

4 Examples

5 Conclusion and future work

Formal verification

- Formal verification is a key aspect of the analysis of systems, it goal is to prove that certain properties are respected.
- In particular safety properties which ensure that the system will never have an unsafe behavior.
- Proving a safety property can be translated as proving that an unsafe region can never be reached from an initial region.

Context

Dynamical system

A dynamical system which state $\mathbf{x} \in \mathbb{R}^n$ evolves according to :

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t)) \tag{1}$$

Solution

Given an initial state $\mathbf{x}(0) = \mathbf{x_0}$, a solution for the previous system is a continuous derivable function $\Omega(t)$ such that $\Omega(0) = x_0$ and $\dot{\Omega}(t) = f(\Omega(t)) \; \forall t \geq 0$

Problematic

Consider an initial region $X_i \subset \mathbb{R}^n$, an unsafe region $X_u \subset \mathbb{R}^n$. The dynamical system remains in the safe region (or is safe).
If $\forall \mathbf{x}_i \in X_i$ and $\forall t > 0$, $\mathbf{x}(t, \mathbf{x}_i) \notin X_u$, i.e.,the system cannot reach X_u starting from X_i.

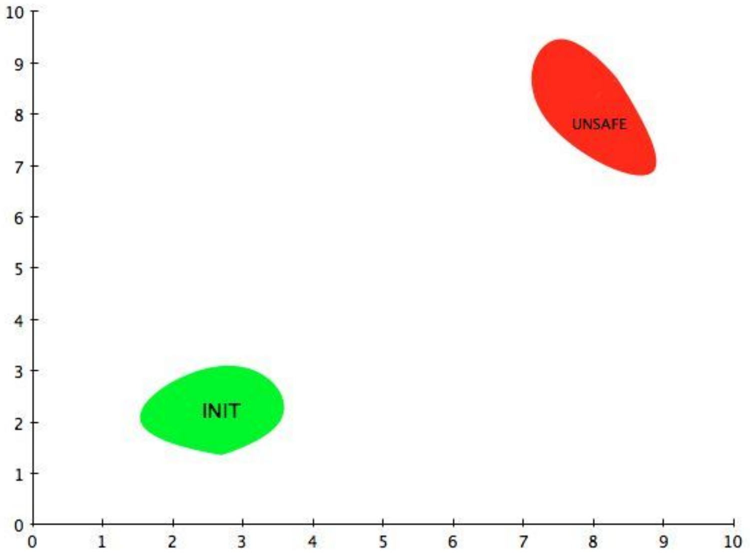
Different Approach

- Computation of the reachable set (SpaceEx,Althoff al.).
- Finding an invariant for the system(Tiwari al.).
- **Barrier certificate**(Prajna al.).

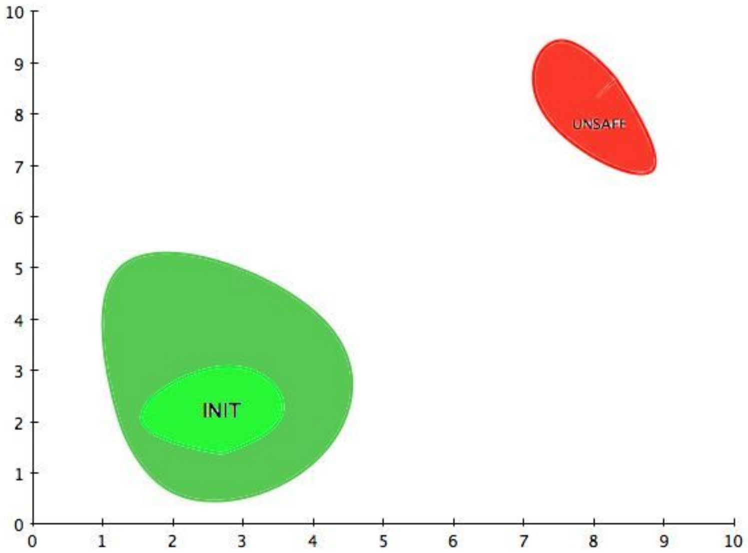
Computation of the reachable set

- This approach consist of explicit computation of the reachable set starting from an initial region.
- It tries to compute an over-approximation of the reachable set using geometrical representation propagated through the dynamical system.
- And if the computed set will not intersect with the unsafe region that will mean that the system is safe.

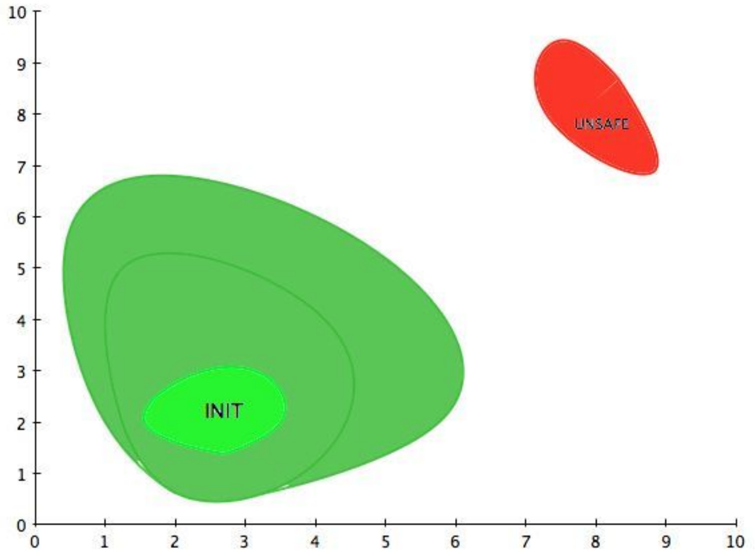
Computation of the reachable set



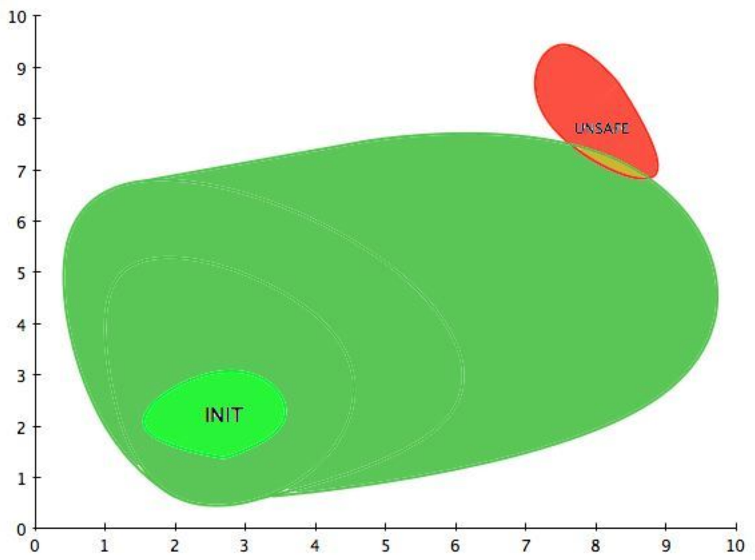
Computation of the reachable set



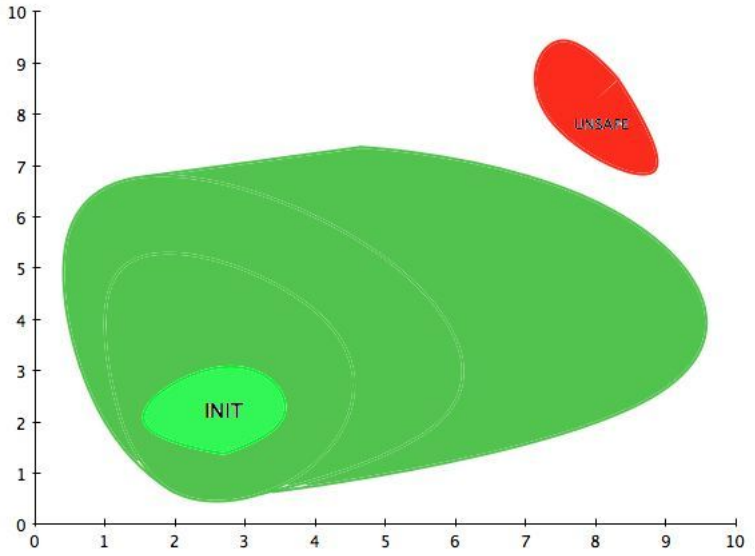
Computation of the reachable set



Computation of the reachable set



Computation of the reachable set



Computation of the reachable set

Limitation

- Can compute the reachable set only for a bounded time.
- The computation of the reachable set can be computationally heavy for non linear dynamics.

Invariant

Definition

Consider the dynamical system : $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t))$ with $\mathbf{x} \in \mathbb{R}^n$. An invariant set $S \subseteq \mathbb{R}^n$ verifies :

$$\forall \mathbf{x_0} \in S \text{ and } \forall t \geq 0,\ \mathbf{x}(t) \in S \quad (2)$$

And if $S \cap X_u = \emptyset$ then system is safe.

Example

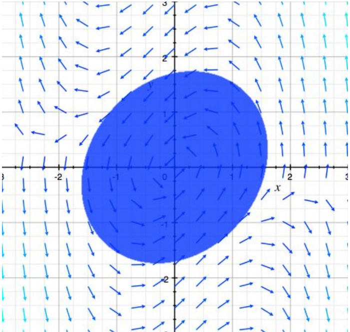
- Equilibrium points.
- Limit cycles.
- Level sets of Lyapunov function i.e.. $\{x : V(x) \leq V_0\}$ for a constant V_0.

Example

Let consider the Van-der-pol equation :

$$\begin{pmatrix} \dot{x_0} \\ \dot{x_1} \end{pmatrix} = \begin{pmatrix} -x_1 \\ x_0 - (1 - x_0^2)x_1 \end{pmatrix}$$

The following invariant is given by the Lyapounov inequality $x_0^2 - 0.34x_0x_1 + 0.85x_1^2 \leq 2$



1 Context

2 Barrier certificate

3 Approach

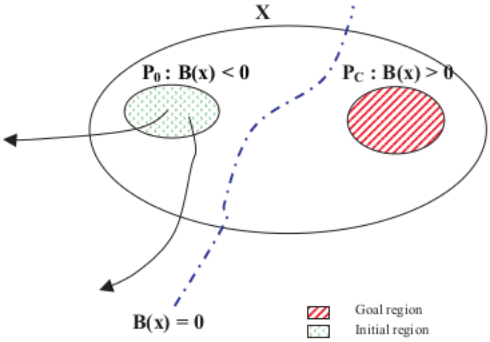
4 Examples

5 Conclusion and future work

Barrier certificate

Approach

The barrier certificate approach **does not require** the computation of the reachable set, instead it searches a **function** that separates an unsafe region from all the trajectories starting form a given initial region.



Example

Example

Let us consider the dynamical system :

$$\begin{pmatrix} \dot{x}_0 \\ \dot{x}_1 \end{pmatrix} = \begin{pmatrix} x_0 \\ -x_1 \end{pmatrix} \quad .$$

With the initial region $X_i = [-2.5, -2.1] \times [3, 3.5]$ and the unsafe region $X_u = [-1, -0.5] \times [1.5, 2]$. A valid barrier certificate is : $B(\mathbf{x}) = 1.79x_0 - 0.86x_1 + 6.1607$

Barrier certificate

Definition

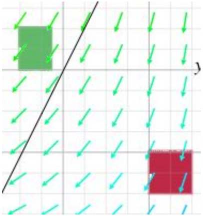
A barrier certificate is a function $B : \mathbb{R}^n \to \mathbb{R}$ defined by those constraints :

Constraints

$$\forall \mathbf{x} \in X_i, B(\mathbf{x}) \leq 0 \quad (3)$$

$$\forall \mathbf{x} \in X_u, B(\mathbf{x}) > 0 \quad (4)$$

$$\forall \mathbf{x} \in X_s \; s.t, B(\mathbf{x}) = 0, \quad \left\langle \frac{\partial B}{\partial x}(\mathbf{x}), f(\mathbf{x}) \right\rangle \leq 0 \quad (5)$$



Template Barrier Certificate

Problematic

To find such function it implies to search over the functional spaces which can be hard.

Template

A template of a barrier certificate $B(\mathbf{x}, \mathbf{p})$ defined $B : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, can be an approach to solve such problem. We will have to search for parameter $\mathbf{p}$ that satisfies the constraints (3)-(5)

Reformulation

So the constraints (3)-(5) can be reformulate as :

$$\exists \mathbf{p} \in \mathbb{R}^m \ : \quad \begin{cases} \forall \mathbf{x} \in X_i & B(\mathbf{x}, \mathbf{p}) \leq 0 \\ \forall \mathbf{x} \in X_u & B(\mathbf{x}, \mathbf{p}) > 0 \\ \forall \mathbf{x} \in X_S \text{ s.t. } B(\mathbf{x}, \mathbf{p}) = 0 & \left\langle \frac{\partial B}{\partial x}(\mathbf{x}, \mathbf{p}), f(\mathbf{x}) \right\rangle \leq 0 \end{cases} . \quad (6)$$

Example

For example consider the template $B(\mathbf{x}, \mathbf{p}) = p_0 x_0 + p_1 x_1 + p_2$ to solve the barrier certificate, we just have to find some parameters $(p_0, p_1, p_2) \in R^3$ that satisfy all the constraints of (6)

1 Context

2 Barrier certificate

3 Approach

4 Examples

5 Conclusion and future work

Interval

Interval analysis

We use interval analysis to solve the constraints, so all the variables are defined by intervals.

Definition

An interval is represented by $[\underline{x}, \overline{x}] = \{x \in \mathbb{R} / \underline{x} \leq x \leq \overline{x}\}$. We denote $\mathbb{I}$ by the set of the bounded interval over $\mathbb{R}$.
We call a box an interval vector e.g.,([1,2],[3,4])

Interval arithemitic

- All the classical operations of the classical arithmetic have there equivalent in interval we define :

$$[\underline{x}, \overline{x}] + [\underline{y}, \overline{y}] = [\underline{x} + \underline{y}, \overline{x} + \overline{y}]$$

$$[\underline{x}, \overline{x}] - [\underline{y}, \overline{y}] = [\underline{x} - \underline{y}, \overline{x} - \overline{y}]$$

$$[\underline{x}, \overline{x}] * [\underline{y}, \overline{y}] = [min\{\underline{x} * \underline{y}, \underline{x} * \overline{y}, \overline{x} * \underline{y}, \overline{x} * \overline{y}\}, max\{\underline{x} * \underline{y}, \underline{x} * \overline{y}, \overline{x} * \underline{y}, \overline{x} * \overline{y}\}]$$

$$[\underline{x}, \overline{x}] / [\underline{y}, \overline{y}] = [min\{\underline{x}/\underline{y}, \underline{x}/\overline{y}, \overline{x}/\underline{y}, \overline{x}/\overline{y}\}, max\{\underline{x}/\underline{y}, \underline{x}/\overline{y}, \overline{x}/\underline{y}, \overline{x}/\overline{y}\}]$$

$$with\ 0 \notin [\underline{y}, \overline{y}]$$

Inclusion function

- Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ an inclusion fonction $F : \mathbb{I}^n \rightarrow \mathbb{I}$ is define :

$$\{f(a_1, ..., a_n) | \exists a_1 \in I_1, .., \exists a_n \in I_n\} \subseteq F(I_1, ..., I_n) \qquad (7)$$

Example

Let take the real function $f(x, y) = x(x - y)$ and the extension F over the interval. If we evaluate f with $0 \leq x \leq 2$ and $0 \leq y \leq 2$ the result will be $-1 \leq f(x, y) \leq 4$, but for the interval version $F([0, 2], [0, 2]) = [-4, 4]$

Problematic

Given a inclusion function f , a box [z] and [x] finding :

$$\exists \mathbf{x} \in [\mathbf{x}], \quad f(\mathbf{x}) \in [\mathbf{z}] \qquad (8)$$

Interval

Contractor

A contractor $\mathcal{C}_{[f],[\mathbf{z}]}$ associated with the generic constraint is a function taking a box $[\mathbf{x}]$ as input and returning a box

$$\mathcal{C}_{[f],[\mathbf{z}]}\left([\mathbf{x}]\right) \subseteq [\mathbf{x}] \tag{9}$$

such that

$$f\left([\mathbf{x}]\right) \cap [\mathbf{z}] = f\left(\mathcal{C}_{[f],[\mathbf{z}]}\left([\mathbf{x}]\right)\right) \cap [\mathbf{z}] \quad . \tag{10}$$

Example

Let take the constraint $x^2 - 1 \leq 0$ and $x = [0.5, 4]$, using the forward backward contractor found in the toolbox **Ibex**, the contraction gave the interval $[0.5, 2]$. To note that it includes the real contraction which is $[0.5, 1]$

Algorithm

Constraints

$\exists \mathbf{p} \in \mathbb{R}^m$

$$\forall \mathbf{x} \in X_i \quad B(\mathbf{x}, \mathbf{p}) \leq 0 \tag{11}$$

$$\forall \mathbf{x} \in X_u \quad B(\mathbf{x}, \mathbf{p}) > 0 \tag{12}$$

$$\forall \mathbf{x} \in X_S \text{ s.t. } B(\mathbf{x}, \mathbf{p}) = 0 \left\langle \frac{\partial B}{\partial x}(\mathbf{x}, \mathbf{p}), f(\mathbf{x}) \right\rangle \leq 0 \tag{13}$$

CSC-FPS

To find the parameters that satisfy the constraints we used a branch and bound algorithm found in (L Jaulin, E Walter 1996) , based on two procedures.

- FPS : searches for vector of parameters $\mathbf{p_0}$ from an initial box $[\mathbf{p}]$ that satisfies all the constraints.
- CSC : validates or invalidates a box of parameters candidate, it checks if the middle of box satisfy (6), or tries to invalidate the whole box.

FPS

```
Input: [p],[xi] , [xu], [xs]
queue Q := [P];
decidable := true;
while Q not empty do
    [p] := dequeue(Q);
    [pc] := contract(B([xi] , [p]) > 0);
    [pc] := contract(B([xu], [pc]) ≤ 0);
    [pc] := contract( ∂B/∂x f([xs], [pc]) > 0 and B([xs], [Pc]) = 0);
    code := CSC([pc],[xi] , [xu], [xs]);
    if code = true then
        return(mid([pc]));
    else
        if code = undetermined then
            if width([p]) < ε_fps then
                decidable := false
            else
                ([Pc, 1],[Pc, 2]) := bisect([Pc]);
                enqueue(Q, [Pc, 1]) ;
                enqueue(Q, [Pc, 2]);
            end
        end
    end
end
if decidable=true then
    return(∅);
else
    return(undetermined);
end
```

Algorithm 1: FPS

Input: $[\mathbf{p}]$, $\{[\mathbf{x_i}]$, $[\mathbf{x}_u]$, $[\mathbf{x}_s]\}$

$t_i := CSCInit([\mathbf{x_i}] , [\mathbf{p}])$;

$t_u := CSCUnsafe([\mathbf{x}_u], [\mathbf{p}])$;

$t_b := CSCBorder([\mathbf{x}_s], [\mathbf{p}])$;

if $t_i = \mathtt{true}$ *and* $t_u = \mathtt{true}$ *and* $t_i = \mathtt{true}$ **then**

  return(`true`);

else

  if $t_i = \mathtt{false}$ *or* $t_u = \mathtt{false}$ *or* $t_b = \mathtt{false}$ **then**

    return(`false`);

  else

    return(`undermined`);

  end

end

Algorithm 2: CSC

CSCInit

```
Input: [x_i] , [p]
m := mid([p]) ; decidable := true ; stack S := [x_i];
while S not empty do
    [x] := unstack(S);
    [x_c] := contract(B([x] , [p]) ≤ 0);
    if [x_c] ≠ [x] then
        return(false);
    end
    if B(mid([x]), [p])>0 then
        return(false);
    end
    if B([x] , m) ≤ 0 then
        continue;
    else
        [x_c] := contract(B([x] , m) > 0) ;
        if [x_c] ≠ ∅ then
            if width([x]_c) < ε_csc then
                decidable := false;
            else
                ([x_{c,1}] , [x_{c,2}]) := bisect([x_c]);
                stack(S, [x_c, 1]) ;
                stack(S, [x_c, 2]) ;
            end
        end
    end
end
if decidable = false then
    return(undermined);
else
    return(true);
end
```

Algorithm 3: CSCInit

Execution

Example

Let consider the following system :

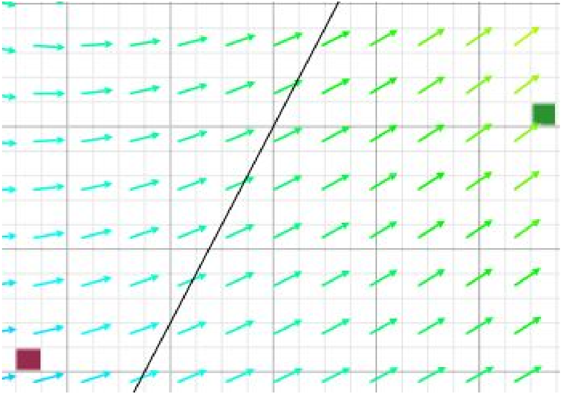
$$\begin{pmatrix} \dot{x_0} \\ \dot{x_1} \end{pmatrix} = \begin{pmatrix} x_0 + x_1 \\ x_0 x_1 - 0.5 x_1^2 \end{pmatrix}$$

$X_i = [3, 3.1] \times [2, 2.1]$, $X_u = [1, 1.1] \times [1, 1.1]$. The template $B(\mathbf{x}, \mathbf{p}) = p_0 x_0 + p_1 x_1 + p_2$ and $\mathbf{p} \in [-10, 10]^3$

solution

The resulting barrier is $B(\mathbf{x}) = -5x_0 + 2.5x_1 + 5$

Example



Example of execution

```
DATA :

Barrier: B:(x0,x1,a,b,c)->(((a*x0)+(b*x1))+c)
Differentiated barrier: B:(x0,x1,a,b,c)->((a*(x0+x1))+(b*((x0*x1)-(0.5*x1^2))))

Init: ([3, 3.1] ; [2, 2.1])
Unsafe: ([1, 1.1] ; [1, 1.1])
State-space: ([1, 3.1] ; [1, 2.1])
Parameters: ([-10, 10] ; [-10, 10] ; [-10, 10])

Iteration : 1

Start FPS
currentParams at the beginning ([-10, 10] ; [-10, 10] ; [-10, 10])
FPS: after contraction ([-10, 10] ; [-10, 10] ; [-10, 10])

middle of parameter : (0,0,0)

Start CSC
CSC Init ... Done and returned True
CSC Unsafe ... Done and returned Undetermined
CSC Border ... Done and True
CSC Result is Undetermined
```

Example of execution

```
Iteration : 2

FPS result is Undetermined: split parameter box
currentParams at the beginning ([-10, 0] ; [-10, 10] ; [-10, 10])
FPS: after contraction ([[-10, 0] ; [-5, 10] ; [-10, 10])
middle of parameter : (-5,2.5,0)


Start CSC
CSC Init ... Done and returned True
CSC Unsafe ... Done and returned Undetermined
CSC Border ... Done and True
CSC Result is Undetermined
```

Example of execution

```
Iteration : 3

FPS result is Unknown: split parameter box
currentParams at the beginning ([0, 10] ; [-10, 10] ; [-10, 10])
FPS: after contraction ([0, 10] ; [-10, 10] ; [-10, 10])
middle of parameter : (5,0,0)

Start CSC
CSC Init ... Done and returned Undetermined
CSC Unsafe ... Done and returned True
CSC Border ... Done and returned True
CSC Result is Undetermined
```

Example of execution

```
Iteration : 4

FPS result is Undetermined: split parameter box
currentParams at the beginning ([-10, 0] ; [-5, 10] ; [-10, 0])
FPS: after contraction ([-6.999999682, 0] ; [-0, 10] ; [-10, 0])
middle of parameter : (3.5,5,-5)

Start CSC
CSC Init ... Done and returned True
CSC Unsafe ... Done and returned Undetermined
CSC Border ... Done and returned True
CSC Result is Unknown
```

Example of execution

```
Iteration : 5

FPS result is Unknown: split parameter box
currentParams at the beginning ([-10, 0] ; [-5, 10] ; [0, 10])
FPS: after contraction ([-10, 0] ; [-5, 10] ; [0, 10])
middle of parameter : (-5,2.5,5)
Start CSC

CSC Init ... Done and returned1
CSC Unsafe ... Done and returned1
CSC Border ... Done and returned1
CSC Result is True

FPS result is True: we found solution
Solution found with the following parameters: (-5 ; 2.5 ; 5)
```

Implementation

- The algorithm was implemented in C++ using Ibex(G.Chabert) interval liberary
- The test was made using a 2.7 ghz intel core i5 processor
- The state space was taken as the convex hull of the initial region and the unsafe region
- $\epsilon_{csc} = 10^{-1}$ and $epsilon_{fps} = 10^{-5}$

1. Context
2. Barrier certificate
3. Approach
4. Examples
5. Conclusion and future work

Example

Example

Consider the perturbed dynamical system

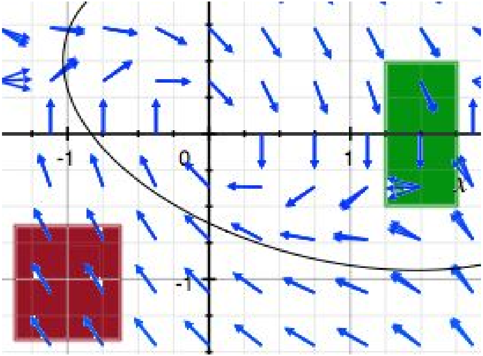
$$\begin{pmatrix} \dot{x}_0 \\ \dot{x}_1 \end{pmatrix} = \begin{pmatrix} x_1 \\ -x_0 + \frac{\mathbf{d}}{3} x_0^3 - x_1 \end{pmatrix}$$

With $d \in [0.9, 1.1]$, $X_i = [1, 2] \times [-0.5, 0.5]$ and $X_u = [-1.4, -0.6] \times [-1.4, -0.6]$

Result

The algorithm finds the following barrier in 5 sec
$B(x) = 2.5x_0^2 + 7.5x_1^2 + 2.5x_0x_1 - 5x_0 - 5x_1 - 5.9$

Example



With $d = \{0.9, 1, 1.1\}$

Example

Example

Consider the perturbed dynamical system

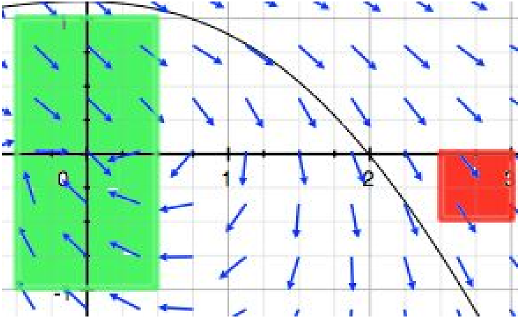
$$\begin{pmatrix} \dot{x}_0 \\ \dot{x}_1 \end{pmatrix} = \begin{pmatrix} -x_0 + x_1 + 0.5(\exp(x_0) - 1) \\ -x_0 - x_1 + x_0 x_1 + x_0 \cos(x_0) \end{pmatrix}$$

With $X_i = [-0.5, 0.5] \times [-1, 1]$ and $X_u = [2.5, 3] \times [-0.5, 0]$

Result

The algorithm finds the following barrier in 5m40 sec
$B(\mathbf{x}) = 0.825x_0^2 - 0.625x_1^2 - 1.25x_0x_1 + 1.5x_0 + 6.25x_1 - 6.25$

Example



Example

Example

Consider the perturbed dynamical system

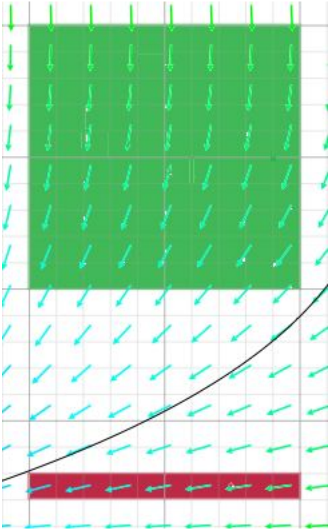
$$\begin{pmatrix} \dot{x_0} \\ \dot{x_1} \end{pmatrix} = \begin{pmatrix} -x_0 + x_0 x_1 \\ -x_1 \end{pmatrix}$$

With $X_i = [0.5, 1] \times [0.5, 1]$ and $X_u = [0.5, 1] \times [0.1, 0.15]$

Result

The algorithm finds the following barrier in 3.868 sec
$B(\mathsf{x}) = ln(\frac{-2.47725x_0}{-9.6875x_1}) + 1.25216x_1$

Example



Bench

Table : Computation results

Example	Barrier	Time (in sec.)	Memory
P0	$1.39583x - 1.25y - 7.5 = 0$	0.7	2.5kb
P1	$-7.5x^2 + 4.04762y^2 + 5.14286xy + 5x + 5y + 5 = 0$	2.3	18.3kb
P2	$-0.0639947t^2 + 0.820312t + 5.60238x - 5.32227 = 0$	104.8	3.9Mb
P3	$2.5x^2 + 7.5y^2 + 2.5xy - 5x - 5y - 5.9 = 0$	16.2	36.9kb
P4	$1.07143t + 3.75y - 7.5 = 0$	0.13	1.5kb
P5	$-1.25x - 1.25y - 2.5 = 0$	0.49	1.9kb
P6	$-7.8125x - 6.875y + 9.375z + 2.4375 = 0$	16.7	0.4Mb
P7	$-0.625x^2 - 1.25y^2 - 3.75xy + 6.25x + 8.75y - 8.75 = 0$	1184.8	5.8Mb
P8	$-2.5x^2 - 7.5y^2 - 2.5xy + 2.5x + 7.5y + 7.5 = 0$	55.5	0.17Mb

1. Context
2. Barrier certificate
3. Approach
4. Examples
5. Conclusion and future work

Conclusion

- We presented a new method to find barrier certificate, based on the search of parameters of a function.
- The main advantage of our technique is that it does not restrict the dynamics nor the template of the barrier certificate.
- We were able to find barrier certificates for a large class of dynamical systems.

Future work

- Find a better strategy for the search of the parameters.
- Find an automatic way to chose a well suited template for each dynamics.
- Make an extension to handle hybrid systems.

Thank you for your attention.