

Bounded and Unbounded Safety Verification using Bisimulation Metrics^{*}

Gang Zheng and Antoine Girard

Laboratoire Jean Kuntzmann, Université de Grenoble
{Gang.Zheng,Antoine.Girard}@imag.fr

Abstract. In this paper, we propose an algorithm for bounded safety verification for a class of hybrid systems described by metric transition systems. The algorithm combines exploration of the system trajectories and state space reduction using merging based on a bisimulation metric. The main novelty compared to an algorithm presented recently by Lerda et.al. lies in the introduction of a tuning parameter that makes it possible to increase the performances drastically. The second significant contribution of this work is a procedure that allows us to derive, in some cases, a proof of unbounded safety from a proof of bounded safety via a refinement step. We demonstrate the efficiency of the approach via numerous experimental results.

1 Introduction

Formal methods are now well established in the domain of embedded systems, both for software and hardware design. Model checking [1] has been used successfully in various industrial settings. Still, algorithmic verification of computing systems embedded in a physical environment, involving interactions of discrete and continuous dynamics, remains a great challenge. Several approaches have emerged from hybrid systems research, ranging from approximate reachability analysis, to abstraction techniques and safety certificates. We refer the reader to the proceedings of the conference on *Hybrid Systems: Computation and Control* for a fair overview of the area. Recently, several safety verification techniques based on exploration of individual trajectories of a system have been proposed [2–6], arguing the moderate cost of the numerical simulation of individual trajectories relatively to the complex computations involved in the previously mentioned techniques.

In this paper, we first consider the problem of bounded safety verification for a class of hybrid systems described by metric transition systems. We propose an algorithm that determines if there exists a trajectory of bounded length that can reach a set of unsafe states. It combines exploration of the system trajectories and state space reduction using merging based on a bisimulation metric [3, 7]. Intuitively, a bisimulation metric measures how far two states of the system are from being bisimilar. A similar approach has recently been proposed in [6]; there

^{*} This work was supported by the ANR SETIN project VAL-AMS.

are significant differences though. The most important one lies in the introduction of a tuning parameter ρ that determines the opportunity of merging states. For $\rho = 0$, states are never merged and we make an exhaustive exploration of the trajectories of the system. For $\rho = 1$, states are merged whenever it is possible and we essentially get the algorithm presented in [6]. Interestingly, intermediate choices for the parameter ρ may increase drastically the performances of the algorithm. The second significant contribution is that we establish a procedure that makes it possible, in some cases, to derive a proof of unbounded safety (no trajectory of any length can reach the set of unsafe states) from a proof of bounded safety via a refinement step. Finally, we use our approach for the verification of a discrete-time switched linear system where the set of admissible switching sequences is specified by an automaton. We demonstrate the efficiency of the approach via numerous experimental results.

2 Problem Formulation

We first introduce the class of metric transition systems and the associated bisimulation metrics. Then, we define the bounded and unbounded safety properties.

2.1 Metric transition systems

In this paper, we will use metric transition systems as an abstract formalism for describing hybrid systems. Essentially, metric transition systems are “classical” transition systems whose set of states is equipped with a pseudo-metric¹.

Definition 1. *A metric transition system (MTS) is a tuple $T = (Q, \rightarrow, Q_0, d)$ consisting of:*

- A set of states Q ,
- A transition relation $\rightarrow \subseteq Q \times Q$,
- A set of initial states $Q_0 \subseteq Q$,
- A pseudo-metric $d : Q \times Q \rightarrow \mathbb{R}^+ \cup \{+\infty\}$

A transition $(q, q') \in \rightarrow$ will be denoted $q \rightarrow q'$. Let us remark that the set of states can be discrete, continuous or hybrid. For all $q \in Q$, we will denote

$$\text{succ}(q) = \{q' \in Q \mid q \rightarrow q'\}.$$

For simplicity, we shall assume that the system is *non-blocking* (for all $q \in Q$, $\text{succ}(q)$ has at least one element) and possibly *non-deterministic* ($\text{succ}(q)$ may have more than one element). We will further assume that the system is *finitely branching* ($\text{succ}(q)$ have a finite number of elements).

A *trajectory* of a MTS is a finite sequence of states $s = q_0 \dots q_K$ such that $q_k \rightarrow q_{k+1}$, for all $0 \leq k < K - 1$. K is referred to as the *length* of s . In addition, we say that s is *initialized* if $q_0 \in Q_0$.

¹ A pseudo-metric over a set Q is a function $d : Q \times Q \rightarrow \mathbb{R}^+ \cup \{+\infty\}$ which satisfies the following properties: for all $q_1, q_2, q_3 \in Q$, (i) $d(q_1, q_1) = 0$, (ii) $d(q_1, q_2) = d(q_2, q_1)$, (iii) $d(q_1, q_3) \leq d(q_1, q_2) + d(q_2, q_3)$.

Example 1. Let us introduce a simple example that we will use for illustration throughout the paper. We consider a discrete-time switched linear system where the set of admissible switching sequences is specified by an automaton. Let $T = (Q, \rightarrow, Q_0, d)$ be a MTS where the hybrid set of states is $Q = \{1, 2, 3, 4\} \times \mathbb{R}^2$. For two states, $q = (\sigma, x)$ and $q' = (\sigma', x')$ there is a transition $q \rightarrow q'$ if and only if

$$\begin{cases} \sigma = 1 \text{ and } \sigma' = 2 & \text{and } x' = A_1x + b_1 \\ \sigma = 2 \text{ and } \sigma' = 3 & \text{and } x' = A_2x + b_2 \\ \sigma = 3 \text{ and } \sigma' \in \{1, 4\} & \text{and } x' = A_2x + b_2 \\ \sigma = 4 \text{ and } \sigma' \in \{1, 3\} & \text{and } x' = A_2x + b_2 \end{cases}$$

where A_1, A_2, b_1 and b_2 are matrices and vectors given by :

$$A_1 = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, A_2 = \begin{pmatrix} 0 & 2 \\ 0.1 & 0 \end{pmatrix}, b_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, b_2 = \begin{pmatrix} -1 \\ 0.9 \end{pmatrix}.$$

We can see that the system is non-deterministic. The set of initial states is $Q_0 = \{(1, x_0)\}$ with $x_0 = (0 \ 0)^T$ and the pseudo-metric over the set of states is $d(q, q') = \|x - x'\|$ where $q = (\sigma, x)$ and $q' = (\sigma', x')$. The MTS T is represented on Figure 1. This system belongs to a class of models that has been studied recently in [8] in the context of control mode scheduling for switched systems.

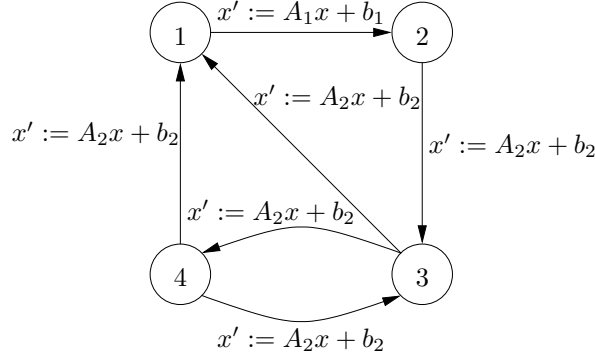


Fig. 1. Example of metric transition system.

2.2 Bisimulation metrics

We first define the notion of bisimulation relation [9] in the framework of metric transition systems.

Definition 2. A relation $\sim \subseteq Q \times Q$ is a bisimulation relation for the MTS $T = (Q, \rightarrow, Q_0, d)$ if for all $q_1 \sim q_2$ the following conditions hold:

1. $d(q_1, q_2) = 0$;
2. For all $q_1 \rightarrow q'_1$, there exists $q_2 \rightarrow q'_2$ such that $q'_1 \sim q'_2$;
3. For all $q_2 \rightarrow q'_2$, there exists $q_1 \rightarrow q'_1$ such that $q'_1 \sim q'_2$.

The notion of bisimulation function has been introduced in [7] as a quantitative generalization of the usual bisimulation relation. In this paper, we use bisimulation metrics which are both pseudo-metrics and bisimulation functions [3]:

Definition 3. A function $V : Q \times Q \rightarrow \mathbb{R}^+ \cup \{+\infty\}$ is a bisimulation metric for the MTS $T = (Q, \rightarrow, Q_0, d)$ if the following conditions hold:

1. V is pseudo-metric;
2. For all $q_1, q_2 \in Q$,

$$V(q_1, q_2) \geq d(q_1, q_2); \quad (1)$$

3. There exists $\lambda \in \mathbb{R}^+$ such that, for all $q_1, q_2 \in Q$,

$$\lambda V(q_1, q_2) \geq \max_{q_1 \rightarrow q'_1} \min_{q_2 \rightarrow q'_2} V(q'_1, q'_2). \quad (2)$$

If $\lambda < 1$, then the MTS T is said to be contractive with respect to bisimulation.

Remark 1. In the original definition of the bisimulation function, there is, in addition to equations (1) and (2), the symmetrical of equation (2). It is not stated explicitly here because symmetry is implied by V being a pseudo-metric.

It is straightforward to check that the zero set of V is a bisimulation relation for the MTS T . Then, the intuitive meaning of a bisimulation metric is a measure of how far two states are from being bisimilar. Describing a systematic way to compute a bisimulation metric for arbitrary metric transition systems is out of the scope of this paper. However, for the system considered in example 1, we show how a bisimulation metric can be obtained. A similar approach can be used for other systems of the same class.

Example 2. Let $T = (Q, \rightarrow, Q_0, d)$ be the MTS defined in example 1. We start by remarking that the relation, given for $q_1 = (\sigma_1, x_1)$, $q_2 = (\sigma_2, x_2)$ by $q_1 \sim q_2$ if and only if $x_1 = x_2$ and either $\sigma_1 \in \{1, 2\}$ and $\sigma_2 = \sigma_1$, or $\sigma_1 \in \{3, 4\}$ and $\sigma_2 \in \{3, 4\}$; is a bisimulation relation for the MTS T . Then, let us search for a bisimulation metric of the form

$$V(q_1, q_2) = \begin{cases} \|x_1 - x_2\|_1 & \text{if } \sigma_1 = \sigma_2 = 1; \\ \|x_1 - x_2\|_2 & \text{if } \sigma_1 = \sigma_2 = 2; \\ \|x_1 - x_2\|_3 & \text{if } \sigma_1 \in \{3, 4\} \text{ and } \sigma_2 \in \{3, 4\}; \\ +\infty & \text{for all other cases.} \end{cases}$$

where the norms $\|\cdot\|_i$ ($i = 1, 2, 3$) are given by $\|x\|_i = \sqrt{x^T M_i x}$ with M_i positive definite symmetric matrices. Let us remark that the zero set of V is the bisimulation relation \sim . It is easy to check that V is a pseudo-metric. Equations (1) and (2) in Definition 3 translate to the following linear matrix inequalities :

$$M_i \succeq I, \quad i = 1, 2, 3$$

and

$$\begin{cases} \lambda^2 M_1 \succeq A_1^T M_2 A_1, \\ \lambda^2 M_2 \succeq A_2^T M_3 A_2, \\ \lambda^2 M_3 \succeq A_2^T M_3 A_2 \text{ and } \lambda^2 M_3 \succeq A_2^T M_1 A_2. \end{cases}$$

This set of inequalities can be solved using semidefinite programming. For $\lambda = \sqrt{0.8}$, we find the following matrices

$$M_1 = \begin{pmatrix} 6.25 & 0 \\ 0 & 1.25 \end{pmatrix}, M_2 = \begin{pmatrix} 1 & 0 \\ 0 & 5 \end{pmatrix}, M_3 = \begin{pmatrix} 1 & 0 \\ 0 & 31.25 \end{pmatrix}.$$

Since $\lambda < 1$, the MTS T is contractive with respect to bisimulation.

2.3 Bounded/unbounded safety

In this paper, we will only consider safety verification whose objective is to determine whether there exists a trajectory reaching a predefined undesirable region of the set of states.

Definition 4. *Given a MTS $T = (Q, \rightarrow, Q_0, d)$, a set of unsafe states $Q_u \subseteq Q$ and $N \in \mathbb{N}$, we say that:*

- *A state $q \in Q$ is safe for bound N if for every trajectory $s = q_0 \dots q_K$ with $q_0 = q$ and of length $K \leq N$, we have $q_k \notin Q_u$, for all $0 \leq k \leq K$. The MTS T is safe for bound N , if all $q_0 \in Q_0$ are safe for bound N .*
- *A state $q \in Q$ satisfies the unbounded safety property if for every trajectory $s = q_0 \dots q_K$ with $q_0 = q$, we have $q_k \notin Q_u$, for all $0 \leq k \leq K$. The MTS T satisfies the unbounded safety property if all $q_0 \in Q_0$ satisfy the unbounded safety property.*

In the following, we will assume that the set of initial states Q_0 has only a finite number of elements. Since in addition, we consider finitely-branching systems, there exists only a finite number of trajectories of length N or less. Then, the bounded safety property can always be verified by complete exploration of the trajectories of the system. However, for non-deterministic systems, this approach may be computationally untractable as the number of trajectories to be explored generally grows exponentially with respect to the bound N . Further, since the set of states Q can be infinite, it is generally not possible to verify the unbounded safety property by exploring all reachable states. In the following section, we will show how the use of bisimulation metrics makes it possible to overcome these limitations.

3 Safety Verification using Bisimulation Metrics

In this section, we propose an algorithm for bounded safety verification that does not explore completely the trajectories of the system. In some cases, it allows us to prove unbounded safety.

Definition 5. *Let V be a bisimulation metric, $q \in Q$, $\delta \in \mathbb{R}^+$, the neighborhood $[q]_\delta \subseteq Q$ is defined by*

$$[q]_\delta = \{r \in Q \mid V(q, r) \leq \delta\}.$$

We say that $[q]_\delta$ is safe for bound $N \in \mathbb{N}$, if all $r \in [q]_\delta$ are safe for bound N .

Our algorithm combines exploration of the system trajectories and state space reduction using merging, it works as follows. We start the exploration of the trajectories of T in a depth-first search manner; for each state q , whose safety is verified for some bound M , we compute a neighborhood $[q]_\delta$ safe for bound M . When reaching a new state q , if we previously determined a neighborhood $[p]_\gamma$ safe for a bound L such that $q \in [p]_\gamma$, the safety of q for a bound $M \leq L$ can be determined without exploring further the trajectories starting from q . This is the merging operation.

A similar approach, inspired by the classical explicit state model checking algorithm [1], combining depth-first search and merging based on proximity has recently been proposed in [6]. There are significant differences with our work though. Firstly, we use bisimulation metrics instead of bisimulation functions. The fact that we use metrics allows us to replace some geometrical considerations involving online numerical optimization problems by a simple use of the triangular inequality. Secondly and more importantly, we add a tuning parameter $\rho \in [0, 1]$ to the algorithm that determines the opportunity to merge states. If there is a neighborhood $[p]_\gamma$ safe for a bound L , such that $V(p, q) \leq \rho\gamma$, then q is safe for all bounds $M \leq L$ and we do not explore the trajectories starting from q . Let us remark, that if $\rho\gamma < V(p, q) \leq \gamma$, then q is safe as well; however, in that case we shall explore the trajectories starting from q . For $\rho = 0$ (i.e. states are never merged), we make an exhaustive exploration of the trajectories of the system. For $\rho = 1$ (i.e. states are merged whenever it is possible), we essentially get, as a special case, the algorithm presented in [6]. Interestingly, intermediate choices for the parameter ρ may increase drastically the performances of the algorithm as will be shown later. Another significant contribution compared to [6] is that we establish a procedure that makes it possible, in some cases, to derive a proof of unbounded safety from a proof of bounded safety.

Bisimulation metrics have first been used for bounded safety verification in [3]. However, the algorithm presented in [3] is quite different as it uses breadth-first search and neighborhood splitting. Its performances are far behind those of the algorithm presented in this paper.

3.1 Bounded safety verification algorithm

Algorithm 1 verifies whether a MTS $T = (Q, \rightarrow, Q_0, d)$ is safe for a given bound N . It returns “Safe” if T is safe for bound N . It returns (“Unsafe”, s) if T is unsafe for bound N where $s = q_0 \dots q_K$ is a counterexample (i.e. an initialized trajectory of T of length $K \leq N$ and such that $q_K \in Q_u$).

The algorithm calls recursively the function `check_safety` which verifies if a state $q \in Q$ is safe for a bound $M \leq N$. If q is not safe for bound M , then the function returns (“Unsafe”, s) where s is a counterexample. Otherwise, it returns (“Safe”, $[p]_\gamma, L$) where the couple $([p]_\gamma, L)$ is such that the neighborhood $[p]_\gamma$ is safe for bound $L \geq M$ and $q \in [p]_\gamma$.

We maintain a list of safe neighborhoods (with their safety bound) stored in the global variable \mathcal{N} . Initially, \mathcal{N} is empty. Each time a safe neighborhood is determined, it is added to \mathcal{N} . We compute a transition relation over the set

Algorithm 1 Bounded safety verification algorithm

1: **Input:** MTS T , unsafe set Q_u , bisimulation metric V and bound N .
2: **Output:** “Safe” or (“Unsafe”, s) where s is a counterexample.

3: **Global** $\mathcal{N} \leftarrow \emptyset$; $i \leftarrow 0$;
4: **Global** $\rightsquigarrow \subseteq \mathcal{N} \times \mathcal{N}$; $\text{merge} : \mathcal{N} \rightarrow 2^{\mathcal{Q}}$; $\text{num} : \mathcal{N} \rightarrow \mathbb{N}$;

5: **Main:** ▷ Main procedure to check bounded safety of MTS T
6: **for** each $q \in Q_0$ **do**
7: $\text{result} \leftarrow \text{check_safety}(q, N)$;
8: **if** $\text{result} = (\text{“Unsafe”}, s)$ **then**
9: **return** result ;
10: **end if**
11: **end for**
12: **return** “Safe”;

13: **function** $\text{check_safety}(q, M)$ ▷ Check whether q is safe for bound M
14: **if** $q \in Q_u$ **then** ▷ q is in the unsafe set
15: **return** (“Unsafe”, q);
16: **else if** $\exists ([p]_{\gamma}, L) \in \mathcal{N}$ such that $M \leq L$ and $V(p, q) \leq \rho\gamma$ **then**
17: $\text{merge}([p]_{\gamma}, L) \leftarrow \text{merge}([p]_{\gamma}, L) \cup \{q\}$; ▷ Merging
18: **return** (“Safe”, $[p]_{\gamma}, L$);
19: **else if** $M = 0$ **then** ▷ q is safe for bound 0
20: $\delta \leftarrow d(q, Q_u)$;
21: $\mathcal{N} \leftarrow \mathcal{N} \cup \{([q]_{\delta}, 0)\}$; $i \leftarrow i + 1$;
22: $\text{merge}([q]_{\delta}, 0) \leftarrow \{q\}$; $\text{num}([q]_{\delta}, 0) \leftarrow i$;
23: **return** (“Safe”, $[q]_{\delta}, 0$);
24: **else** ▷ Need to explore the successors of q
25: $\delta \leftarrow d(q, Q_u)$; $\mathcal{S} \leftarrow \emptyset$;
26: **for** each $q' \in \text{succ}(q)$ **do**
27: $\text{result} \leftarrow \text{check_safety}(q', M - 1)$;
28: **if** $\text{result} = (\text{“Unsafe”}, s)$ **then**
29: **return** (“Unsafe”, qs);
30: **else** ▷ Then $\text{result} = (\text{“Safe”}, [p']_{\gamma'}, L')$
31: $\delta \leftarrow \min\left(\delta, \frac{\gamma' - V(p', q')}{\lambda}\right)$;
32: $\mathcal{S} \leftarrow \mathcal{S} \cup \{([p']_{\gamma'}, L')\}$;
33: **end if**
34: **end for**
35: $\mathcal{N} \leftarrow \mathcal{N} \cup \{([q]_{\delta}, M)\}$; $i \leftarrow i + 1$;
36: $\text{merge}([q]_{\delta}, M) \leftarrow \{q\}$; $\text{num}([q]_{\delta}, M) \leftarrow i$;
37: **for** each $([p']_{\gamma'}, L') \in \mathcal{S}$ **do**
38: Set $([q]_{\delta}, M) \rightsquigarrow ([p']_{\gamma'}, L')$;
39: **end for**
40: **return** (“Safe”, $[q]_{\delta}, M$);
41: **end if**

of safe neighborhoods \mathcal{N} , denoted \rightsquigarrow . Additionally, we compute the function $\text{merge} : \mathcal{N} \rightarrow 2^{\mathcal{Q}}$ that keeps track of merging operations, and the function $\text{num} : \mathcal{N} \rightarrow \mathbb{N}$ that records in which order the safe neighborhoods have been computed. Though the transition relation \rightsquigarrow and functions merge , num are not technically necessary for bounded safety verification, they help for the proof of correctness and will be useful for the extension to unbounded safety verification.

The function check_safety works as follows. Given a state q and a bound M , there are four possible cases:

1. If $q \in Q_u$, then q is not safe. The trajectory of length 0 consisting of q is a counterexample. The function returns (“Unsafe”, q).
2. If there exists $([p]_{\gamma}, L) \in \mathcal{N}$, such that $M \leq L$ and $V(p, q) \leq \rho\gamma$, then q is safe for bound M and we merge states. We insert q in $\text{merge}([p]_{\gamma}, L)$ and the function returns (“Safe”, $[p]_{\gamma}, L$).
3. If $M = 0$ and $q \notin Q_u$, then q is safe for bound 0. Let δ be given by

$$\delta = d(q, Q_u) = \inf_{r \in Q_u} d(q, r). \quad (3)$$

We insert $([q]_{\delta}, 0)$ in \mathcal{N} , we set $\text{merge}([q]_{\delta}, 0) = \{q\}$ and the function returns (“Safe”, $[q]_{\delta}, 0$).

4. In the other cases, we need to check the safety for bound $M - 1$ of the successors of q :
 - If one of the successors q' is not safe for bound $M - 1$, then the function $\text{check_safety}(q', M - 1)$ returns a counterexample s . In that case, q is not safe for bound M and a counterexample is given by the trajectory qs . The function returns (“Unsafe”, qs).
 - If all the successors q'_1, \dots, q'_n of q are safe for bound $M - 1$, then q is safe for bound M . Let δ be given by

$$\delta = \min \left(d(q, Q_u), \frac{\gamma'_1 - V(p'_1, q'_1)}{\lambda}, \dots, \frac{\gamma'_n - V(p'_n, q'_n)}{\lambda} \right) \quad (4)$$

where couple $([p'_i]_{\gamma'_i}, L'_i) \in \mathcal{N}$ is returned by $\text{check_safety}(q'_i, M - 1)$, for $i = 1, \dots, n$. We insert $([q]_{\delta}, M)$ in \mathcal{N} , set $\text{merge}([q]_{\delta}, M) = \{q\}$ and $([q]_{\delta}, M) \rightsquigarrow ([p'_i]_{\gamma'_i}, L'_i)$, for $i = 1, \dots, n$. Then, the function returns (“Safe”, $[q]_{\delta}, M$).

Remark 2. We want to point out the following simple but useful properties. Firstly, for all transitions $([q]_{\delta}, M) \rightsquigarrow ([p']_{\gamma'}, L')$, we have, by construction, that $L' \geq M - 1$ and $\text{num}([q]_{\delta}, M) > \text{num}([p']_{\gamma'}, L')$. Secondly, for all $([q]_{\delta}, M) \in \mathcal{N}$, with $M \geq 1$, for all $q \rightarrow q'$, there exists, by construction, $([q]_{\delta}, M) \rightsquigarrow ([p']_{\gamma'}, L')$, such that $q' \in \text{merge}([p']_{\gamma'}, L')$.

Lemma 1. *Let $([q]_{\delta}, M) \in \mathcal{N}$ and $r \in Q$, such that $r \in [q]_{\delta}$, then:*

- $r \notin Q_u$;
- if $M \geq 1$, for all $r \rightarrow r'$, there is $([q]_{\delta}, M) \rightsquigarrow ([p']_{\gamma'}, L')$ such that $r' \in [p']_{\gamma'}$.

Proof. From equations (3) and (4), we have that $\delta \leq d(q, Q_u)$. Then, from equation (1), $d(q, r) \leq V(q, r) \leq \delta \leq d(q, Q_u)$ which implies that $r \notin Q_u$. The first property holds. Let us assume that $M \geq 1$ and let $r \rightarrow r'$, from equation (2), there exists $q \rightarrow q'$, such that $V(q', r') \leq \lambda V(q, r) \leq \lambda \delta$. From equation (4), there exists $([q]_\delta, M) \rightsquigarrow ([p']_{\gamma'}, L')$ such that $\delta \leq (\gamma' - V(p', q'))/\lambda$. Then, it follows from the triangular inequality

$$V(p', r') \leq V(p', q') + V(q', r') \leq V(p', q') + \lambda \delta \leq \gamma'.$$

Hence, $r' \in [p']_{\gamma'}$ and the second property holds. ■

The correctness of algorithm 1 is a direct consequence of Lemma 1.

Theorem 1. *Algorithm 1 is correct: if it returns (“Unsafe”, s) then T is not safe for bound N and the trajectory s is counterexample; if it returns “Safe”, then the MTS T is safe for bound N .*

Proof. The proof of the first part of the theorem is straightforward. Let us assume that the algorithm returns “Safe”. Let $r_0 \dots r_K$ be an initialized trajectory of T of length $K \leq N$. Since algorithm 1 returns “Safe”, there exists $([q_0]_{\delta_0}, N) \in \mathcal{N}$, such that $r_0 \in [q_0]_{\delta_0}$. Let us prove, by induction, that for all $k \in \{0, \dots, K\}$, there exists $([q_k]_{\delta_k}, M_k) \in \mathcal{N}$ such that $r_k \in [q_k]_{\delta_k}$ and $M_k \geq N - k$. This is clearly true for $k = 0$. Let us assume this is true for some $k \in \{0, \dots, K - 1\}$ and show that it is true for $k + 1$. We have $r_k \rightarrow r_{k+1}$, then from Lemma 1, and since $M_k \geq N - k \geq 1$, it follows that there exists $([q_k]_{\delta_k}, M_k) \rightsquigarrow ([q_{k+1}]_{\delta_{k+1}}, M_{k+1})$ such that $r_{k+1} \in [q_{k+1}]_{\delta_{k+1}}$. In addition, we have $M_{k+1} \geq M_k - 1 \geq N - k - 1$. This completes the induction. Further, from Lemma 1, it follows that $r_k \notin Q_u$ for $k \in \{0, \dots, K\}$. Therefore, r_0 is safe for bound N and the MTS T is safe for bound N as well. ■

Remark 3. The termination of algorithm 1 is an obvious consequence of T being finitely branching with a finite set of initial states.

3.2 Unbounded safety verification result

We now move to an interesting result that makes it possible, in some cases, to derive a proof of unbounded safety from the proof of bounded safety provided by algorithm 1. The main idea is the following. Let us assume that a MTS T has been proved safe for some bound N using algorithm 1 and that for all $([p]_\gamma, 0) \in \mathcal{N}$, there exists $([q]_\delta, M) \in \mathcal{N}$ with $M \geq 1$ such that $[p]_\gamma \subseteq [q]_\delta$. Thus, the neighborhoods $[p]_\gamma$ which are safe for bound 0, are included in neighborhoods $[q]_\delta$ safe for bound $M \geq 1$. This implies that the neighborhoods $[p]_\gamma$ are safe for bound 1 which in turns implies that the neighborhoods $[q]_\delta$ are safe for bound $M + 1$. Then, by induction, it can be shown that the MTS T satisfies the unbounded safety property. Unfortunately, the elements of \mathcal{N} generally do not satisfy the previous condition. The reason is that for all $([p]_\gamma, 0) \in \mathcal{N}$, we

have $\gamma = d(p, Q_u)$. Then, $[p]_\gamma$ is generally not included in another safe neighborhood $[q]_\delta$, since such a neighborhood would most probably intersect Q_u which contradicts the fact that it is safe.

Therefore, prior to apply the simple test described previously, we need to refine the neighborhoods in \mathcal{N} . In algorithm 1, the safe neighborhoods are computed backward, the safe neighborhood around a state is determined from the safe neighborhoods around its successors. Our refinement step consists in reevaluating forward these safe neighborhoods. We keep the safe neighborhoods around the initial states unchanged. Then, the safe neighborhoods around the other states are updated according to the safe neighborhoods around their predecessors. More precisely, we use the function $\text{refine} : \mathcal{N} \rightarrow \mathbb{R}^+$ defined recursively as follows. Let $([p']_{\gamma'}, L') \in \mathcal{N}$, there are two different cases:

1. If $L' = N$, then $([p']_{\gamma'}, L')$ corresponds to an initial state of T , we keep it unchanged and $\text{refine}([p']_{\gamma'}, L') = \gamma'$.
2. If $L' \neq N$, we denote $([q_1]_{\delta_1}, M_1), \dots, ([q_n]_{\delta_n}, M_n)$ the elements of \mathcal{N} such that $([q_i]_{\delta_i}, M_i) \rightsquigarrow ([p']_{\gamma'}, L')$, $i = 1, \dots, n$. Then,

$$\text{refine}([p']_{\gamma'}, L') = \max_{i=1}^n \max_{j=1}^{m_i} (\lambda \text{refine}([q_i]_{\delta_i}, M_i) + V(q'_{i,j}, p')). \quad (5)$$

where $\{q'_{i,1}, \dots, q'_{i,m_i}\} = \text{succ}(q_i) \cap \text{merge}([p']_{\gamma'}, L')$.

Lemma 2. For all $([p']_{\gamma'}, L') \in \mathcal{N}$, $\text{refine}([p']_{\gamma'}, L')$ is well defined and satisfies $\text{refine}([p']_{\gamma'}, L') \leq \gamma'$.

Proof. The proof is done by induction on $\text{num}([p']_{\gamma'}, L')$. For $([p']_{\gamma'}, L') \in \mathcal{N}$, $\text{num}([p']_{\gamma'}, L')$ ranges from 1 to $\text{Card}(\mathcal{N})$. Further, it is easy to see that the last couple $([p']_{\gamma'}, L')$ added by algorithm 1, which verifies $\text{num}([p']_{\gamma'}, L') = \text{Card}(\mathcal{N})$ is such that $L' = N$. In that case, $\text{refine}([p']_{\gamma'}, L') = \gamma'$. Now, let us assume that there exists $k \in \{2, \dots, \text{Card}(\mathcal{N})\}$ such that for all $([p']_{\gamma'}, L') \in \mathcal{N}$, satisfying $\text{num}([p']_{\gamma'}, L') \geq k$, $\text{refine}([p']_{\gamma'}, L')$ is well defined and $\text{refine}([p']_{\gamma'}, L') \leq \gamma'$. Let us remark that this holds for $k = \text{Card}(\mathcal{N})$. Let us prove that this true for $k - 1$. Let $([p']_{\gamma'}, L') \in \mathcal{N}$, such that $\text{num}([p']_{\gamma'}, L') = k - 1$, there are two possible cases. If $L' = N$, $\text{refine}([p']_{\gamma'}, L') = \gamma'$ and the property holds. If $L' \neq N$, then for all $([q_i]_{\delta_i}, M_i) \rightsquigarrow ([p']_{\gamma'}, L')$, we have from remark 2, that $\text{num}([q_i]_{\delta_i}, M_i) > \text{num}([p']_{\gamma'}, L')$, hence by the induction assumption, we have that $\text{refine}([q_i]_{\delta_i}, M_i)$ is well defined and $\text{refine}([q_i]_{\delta_i}, M_i) \leq \delta_i$. Hence, $\text{refine}([p']_{\gamma'}, L')$ is well defined. Further, from equation (4), we have that for all $q'_{i,j} \in \text{succ}(q_i) \cap \text{merge}([p']_{\gamma'}, L')$,

$$\gamma' \geq \lambda \delta_i + V(q'_{i,j}, p') \geq \lambda \text{refine}([q_i]_{\delta_i}, M_i) + V(q'_{i,j}, p')$$

Since this holds for all $i \in \{1, \dots, n\}$, for all $j \in \{1, \dots, m_i\}$, it follows that $\text{refine}([p']_{\gamma'}, L') \leq \gamma'$. This completes the induction. ■

We define the set of refined neighborhoods:

$$\hat{\mathcal{N}} = \left\{ ([q]_{\hat{\delta}}, M) \mid \hat{\delta} = \text{refine}([q]_{\delta}, M), ([q]_{\delta}, M) \in \mathcal{N} \right\}.$$

We lift the transition relation \rightsquigarrow from the set \mathcal{N} to the set $\hat{\mathcal{N}}$ by setting $([q]_{\hat{\delta}}, M) \rightsquigarrow ([p']_{\hat{\gamma}'}, L')$ if and only if $([q]_{\delta}, M) \rightsquigarrow ([p']_{\gamma'}, L')$.

Lemma 3. *Let $([q]_{\hat{\delta}}, M) \in \hat{\mathcal{N}}$ and $r \in Q$, such that $r \in [q]_{\hat{\delta}}$, then:*

- $r \notin Q_u$;
- if $M \geq 1$, for all $r \rightarrow r'$, there is $([q]_{\hat{\delta}}, M) \rightsquigarrow ([p']_{\hat{\gamma}'}, L')$ such that $r' \in [p']_{\hat{\gamma}'}$.

Proof. There exists $([q]_{\delta}, M) \in \mathcal{N}$ with $\hat{\delta} = \text{refine}([q]_{\delta}, M)$. From Lemma 2, $\hat{\delta} \leq \delta$, it follows that $[q]_{\hat{\delta}} \subseteq [q]_{\delta}$. Then, from Lemma 1, the first property holds. Let us assume that $M \geq 1$ and let $r \rightarrow r'$, from equation (2), there exists $q \rightarrow q'$, such that $V(q', r') \leq \lambda V(q, r) \leq \lambda \hat{\delta}$. According to remark 2, there exists $([q]_{\hat{\delta}}, M) \rightsquigarrow ([p']_{\hat{\gamma}'}, L')$ such that $q' \in \text{merge}([p']_{\hat{\gamma}'}, L')$. Hence, from equation (5), we have that $\hat{\gamma}' \geq \lambda \hat{\delta} + V(q', p')$. Using the triangular inequality yields

$$V(p', r') \leq V(p', q') + V(q', r') \leq V(p', q') + \lambda \hat{\delta} \leq \hat{\gamma}'.$$

Hence, $r' \in [p']_{\hat{\gamma}'}$ and the second property holds. ■

We can now state the following unbounded safety verification result:

Theorem 2. *Let us assume that*

$$\forall ([p]_{\hat{\gamma}}, 0) \in \hat{\mathcal{N}}, \exists ([q]_{\hat{\delta}}, M) \in \hat{\mathcal{N}} \text{ with } M \geq 1 \text{ such that } [p]_{\hat{\gamma}} \subseteq [q]_{\hat{\delta}}. \quad (6)$$

Then, the MTS T satisfies the unbounded safety property.

Proof. Let $r_0 \dots r_K$ (with $K \in \mathbb{N}$) be an initialized trajectory of T . Since the MTS T has been proved safe for some bound $N \geq 1$ using algorithm 1, there exists $([q_0]_{\delta_0}, N) \in \mathcal{N}$ such that $r_0 \in [q_0]_{\delta_0}$. We have $\hat{\delta}_0 = \text{refine}([q_0]_{\delta_0}, N) = \delta_0$, therefore $([q_0]_{\hat{\delta}_0}, N) \in \hat{\mathcal{N}}$ and $r_0 \in [q_0]_{\hat{\delta}_0}$. Let us prove, by induction, that for all $k \in \{0, \dots, K\}$, there exists $([q_k]_{\hat{\delta}_k}, M_k) \in \mathcal{N}$ such that $r_k \in [q_k]_{\hat{\delta}_k}$ and $M_k \geq 1$. This is clearly true for $k = 0$. Let us assume this is true for some $k \in \{0, \dots, K-1\}$ and show that it is true for $k+1$. We have $r_k \rightarrow r_{k+1}$, then from Lemma 3, it follows that there exists $([q_k]_{\hat{\delta}_k}, M_k) \rightsquigarrow ([p_{k+1}]_{\hat{\gamma}_{k+1}}, L_{k+1})$ such that $r_{k+1} \in [p_{k+1}]_{\hat{\gamma}_{k+1}}$. If $L_{k+1} \geq 1$, then the induction hypothesis holds for $k+1$. If $L_{k+1} = 0$, we have from equation (6) that there exists $([q_{k+1}]_{\hat{\delta}_{k+1}}, M_{k+1}) \in \hat{\mathcal{N}}$ with $M_{k+1} \geq 1$ and such that $[p_{k+1}]_{\hat{\gamma}_{k+1}} \subseteq [q_{k+1}]_{\hat{\delta}_{k+1}}$. It follows that $r_{k+1} \in [q_{k+1}]_{\hat{\delta}_{k+1}}$. This completes the induction. Then, from Lemma 3, we have for all $k = 0 \dots K$, $r_k \notin Q_u$. This completes the proof. ■

The inclusion test $[p]_{\hat{\gamma}} \subseteq [q]_{\hat{\delta}}$ in equation (6) can be replaced conservatively by the easily checkable inequality $V(p, q) + \hat{\gamma} \leq \hat{\delta}$.

Remark 4. From equation (5), we can show that there exists $([p]_{\hat{\gamma}}, 0) \in \hat{\mathcal{N}}$ such that for all $([q]_{\hat{\delta}}, M) \in \hat{\mathcal{N}}$, $\hat{\gamma} \geq \lambda^M \hat{\delta}$. This implies that the unbounded safety verification result given by Theorem 2 cannot be applied if T is not contractive with respect to bisimulation.

4 Experimental Results

In this section, we evaluate the performances of our approach by verifying safety properties of the MTS defined in example 1. The set of unsafe states is parameterized by $\theta \in \mathbb{R}$: $Q_u = \{(\sigma, x) \in Q \mid (0 \ 1)x \geq \theta\}$. It can be verified that by choosing the admissible sequence of discrete states 1, 2, 3, 4, 3, 4, 3, 4, ..., the second component of the continuous state approaches asymptotically 1. Hence, we shall choose $\theta \geq 1$.

4.1 Bounded safety verification

We start with bounded safety verification. Algorithm 1 has been implemented in Matlab. It was applied for several values of the parameters ρ , N and θ . The results are presented in figures 2, 3 and 4.

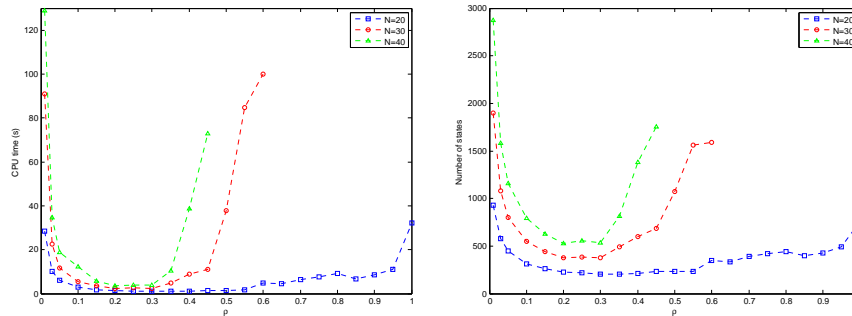


Fig. 2. CPU time and number of elements in \mathcal{N} for several values of ρ and N ($\theta = 1.1$).

We first discuss the importance of the choice of the tuning parameter ρ . Let us recall that for $\rho = 0$, algorithm 1 explores exhaustively all the trajectories of T of length N . For $\rho = 1$, algorithm 1 merges states whenever it is possible and is similar to the algorithm presented in [6]. In figure 2, we can see the performances of the algorithm (CPU time and number of neighborhoods in \mathcal{N}) for several values of ρ and $N \in \{20, 30, 40\}$, $\theta = 1.1$. As expected, the larger the bound N , the more expensive the verification. What is more surprising is the influence of the tuning parameter ρ . It can be seen that the optimal value lies somewhere around $\rho = 0.2$ and not at $\rho = 1$ as one may intuitively think. This means that sometimes, it is better not to merge states, even though it is possible.

We shall try to give an explanation by looking at the distribution of the couples $([q]_\delta, M) \in \mathcal{N}$ as a function of the safety bound M (see figure 3). For larger values of ρ , e.g. 0.6, we can see that the number of couples with a small value of M is much less than the number of couples with an intermediate value for M (bell-shaped distribution). The interpretation is the following. With $\rho = 0.6$, algorithm 1 often uses the opportunity to merge states. This results, for small values of M , in a lot of merging operations and few computed neighborhoods.

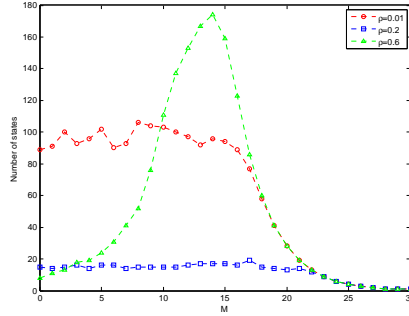


Fig. 3. Distribution of the couples $([q]_\delta, M) \in \mathcal{N}$ as a function of the safety bound M , for several values of the tuning parameter ρ , $N = 30$, $\theta = 1.1$.

However, a lot of merging operations for small values of M result, because of backward computations, in safe neighborhoods of smaller size for larger values of M . As the safe neighborhoods become smaller, algorithm 1 has much less opportunities to merge states and it needs to compute a lot of neighborhoods for intermediate values of M to verify the safety of the system. On the contrary, for smaller values of ρ , e.g. 0.2 and 0.01, the distribution of the couples looks quite uniform. This means that, by merging states less often when it gets the opportunity, algorithm 1 receives this opportunity more often. The optimal balance seems to be obtained for $\rho = 0.2$.

For the sake of comparison, we also implemented the exhaustive exploration without testing for merging, which results in much faster computations than algorithm 1 with $\rho = 0$. It takes 0.4 seconds (4022 explored states) for $N = 20$ and 18.3 seconds (183915 explored states) for $N = 30$, we stopped it before completion for $N = 40$. We can see that even for $N = 30$, our algorithm with $\rho = 0.2$ is already much faster (2.3 seconds) than the exhaustive exploration of the trajectories.

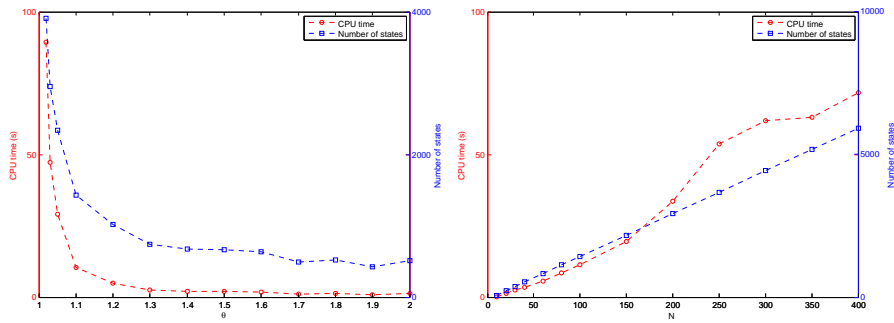


Fig. 4. CPU time and number of elements in \mathcal{N} for several values of θ (left, $N = 100$, $\rho = 0.2$) and N (right, $\theta = 1.1$, $\rho = 0.2$).

We now discuss the influence of the parameters θ and N (ρ is set to 0.2). In figure 4, we represented the performances of the algorithm (CPU time and

number of neighborhoods in \mathcal{N}) for several values of θ and N . We can see that as θ approaches 1, algorithm 1 needs more time to verify the safety of the system: as θ becomes close to 1, some trajectories of the system are very close to the unsafe set resulting in safe neighborhoods of small size and less merging opportunities. This corroborates the fact pointed out in [3] that robust safety properties are easier to verify. The effect of the bound N is quite surprising. When using exhaustive exploration of the trajectories, the time needed for bounded safety verification grows exponentially with parameter N . With our approach and for the optimal value $\rho = 0.2$, it seems that the time needed for bounded safety verification grows only linearly with N . This is a huge improvement. For comparison, we were able to verify, with our approach, bounded safety for $N = 400$ in a minute whereas we interrupted the exhaustive exploration for $N = 40$ since it was taking too much time.

4.2 Unbounded safety verification

We move to unbounded safety verification, the parameter θ is set to 1.1. The bounded safety verification had previously been verified for $N = 30$ with $\rho = 0.2$. Using the result presented in Theorem 2, a proof of unbounded safety could be derived. Most of the computational effort was spent on bounded safety verification so the overall process took less than 3 seconds. On figure 5, we represented the set of safe neighborhoods obtained after bounded safety verification and the set of safe neighborhoods after refinement. The neighborhoods corresponding to safety bound $M = 0$ are represented in dark (red). We can check that, after refinement, these are included in other safe neighborhoods, thus allowing us to conclude that the system satisfies the unbounded safety property.

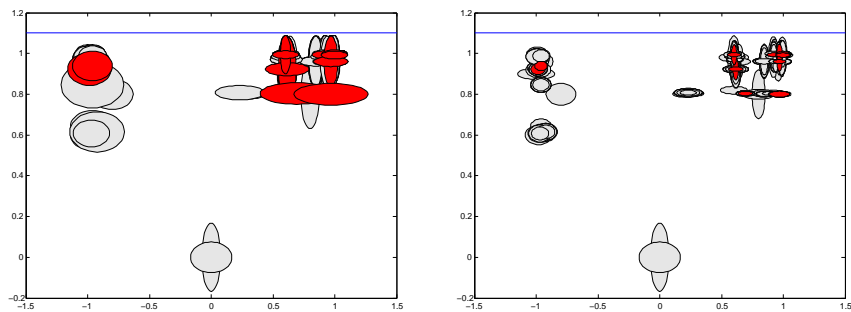


Fig. 5. Set of safe neighborhoods obtained after bounded safety verification (left) and set of refined safe neighborhoods allowing to derive a proof of unbounded safety (right). The darker (red) safe neighborhoods are those corresponding to safety bound $M = 0$. Region above the line is the unsafe set.

5 Conclusion

In this paper, we first presented an algorithm for verifying bounded safety of metric transition systems. In some cases, proofs of unbounded safety can be derived from the results of our algorithm. We provided numerous experiments that show the efficiency of the approach when the tuning parameter ρ of the algorithm is well chosen. Future work will focus on better understanding the influence of this parameter and extending our approach to infinitely branching transition systems.

References

1. Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press (2000)
2. Kapinski, J., Krogh, B., Maler, O., Stursberg, O.: On systematic simulation of open continuous systems. In: HSCC. Volume 2623 of LNCS., Springer (2003) 283–297
3. Girard, A., Pappas, G.: Verification using simulation. In: HSCC. Volume 3927 of LNCS., Springer (2006) 272–286
4. Donzé, A., Maler, O.: Systematic simulation using sensitivity analysis. In: HSCC. Volume 4416 of LNCS., Springer (2007) 174–189
5. Julius, A., Fainekos, G., Anand, M., Lee, I., Pappas, G.: Robust test generation and coverage for hybrid systems. In: HSCC. Volume 4416 of LNCS., Springer (2007) 329–342
6. Lerda, F., Kapinski, J., Clarke, E., Krogh, B.: Verification of supervisory control software using state proximity and merging. In: HSCC. Volume 4981 of LNCS., Springer (2008) 344–357
7. Girard, A., Pappas, G.: Approximation metrics for discrete and continuous systems. IEEE Trans. Automatic Control **52**(5) (2007) 782–798
8. Weiss, G., Alur, R.: Automata based interfaces for control and scheduling. In: HSCC. Volume 4416 of LNCS., Springer (2007) 601–613
9. Milner, R.: Communication and Concurrency. Prentice Hall (1989)