# On Systematic Simulation of Open Continuous Systems

Jim Kapinski[1], Bruce H. Krogh[1], Oded Maler[2], and Olaf Stursberg[3]

[1] Dept. of Electrical and Computer Engineering
Carnegie Mellon University, 5000 Forbes Avenue
Pittsburgh, PA 15213-3890 USA
krogh@ece.cmu.edu jpk3@andrew.cmu.edu
[2] VERIMAG
Centre Equation, 2, av. de Vignate
38610 Gières, France
Oded.Maler@imag.fr
[3] Process Control Lab (CT-AST)
University of Dortmund
44221 Dortmund
Germany
olaf.stursberg@uni-dortmund.de

**Abstract.** In this paper we investigate a new technique to determine whether an open continuous system behaves correctly for all admissible input signals. This technique is based on a discretization of the set of possible input signals, and on storing neighborhoods of points reachable by trajectories induced by those signals. Alternatively, this technique, inspired by automata theory, can be seen as an attempt to make simulation a more systematic activity by finding a small set of input signals such that the behaviors they induce "cover" the whole reachable state space.

## 1 Introduction

Practitioners in control and in other domains use numerical simulations in order to convince themselves that their systems behave correctly. It is a trivial observation that the level of confidence for analysis results from simulations of continuous dynamic systems is much lower than what can be obtained for discrete finite-state systems [M98,M01]. Consider first a closed continuous system over $\mathbb{R}^n$ defined by a differential equation $\dot{\mathbf{x}} = f(\mathbf{x})$ and the following question: does the trajectory $\xi$ of that system starting from a point $\mathbf{x}_0$ ever reach a set $P$? Even if the differential equation admits a closed form solution, this solution is not of much help for answering the reachability problem. For example, in linear systems the problem

$$\exists t \; e^{At}\mathbf{x}_0 \in P$$

is not known to be solvable except for some very special cases [PLY99].

Numerical simulation is based on selecting a discrete set of time points $\bar{T} \subset \mathbb{R}_+$, a discrete but very large subset of the state space $\bar{X} \subset X$ (the floating point numbers), and on successive generation, via numerical integration, of a discretized approximation of $\xi$ of the form $\xi' : \bar{T} \to \bar{X}$ such that for every $t \in \bar{T}$ there is some bound on the distance between $\xi[t]$ and $\xi'[t]$. The user looks at the image of $\xi'$ (with sufficiently fine discretization and graphical resolution it looks continuous to the human eye) and draws conclusions about $\xi$. The fact that the time domain of $\xi'$ is restricted to $\bar{T}$ (and, even there, its value only approximates $\xi$) can be overcome by replacing the condition $\xi'[t] \in P$ with a weaker condition on the distance between $\xi'$ and $P$, or equivalently by over-approximating $P$ by $P'$ to compensate for the approximation error (see Figure 1). This way one can guarantee that if $\xi[t] \in P$ for some $t \in [0, r]$ then there is some $t' \in \bar{T} \cap [0, r]$ such that $\xi'[t'] \in P'$.

The more challenging problem is when the simulated trajectory *does not* reach $P$ and the question is when to stop the simulation. In deterministic finite-state systems every trajectory is ultimately-periodic and the simulation can be stopped once $\xi[t] = \xi[t']$ for some $t' < t$. This is rarely the case in numerical simulation of continuous systems, including those that admit limit cycles.[4] Nevertheless, an intelligent user observing the evolution of $\xi'$, and having a strong intuition regarding the behavior of continuous systems, can become convinced in the non-reachability of $P$ by $\xi$ after performing the simulation for a *finite* amount of time. Given the undecidability of most reachability problems for continuous systems this is the best we can hope for, excluding, of course, techniques of a different nature such as those based on Lyapunov functions. So our starting point is:

**Postulate 1 (Simulation is Fine)** *An intelligent mortal can solve the reachability problem for a well-behaved closed continuous system using a finite amount of numerical simulation.*

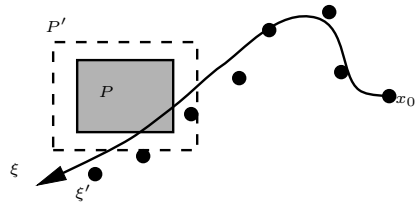


**Fig. 1.** A continuous behavior $\xi$ and its numerical approximation $\xi'$.

Consider now *open systems*, that is, systems exposed to input signals, each of which induces a distinct trajectory. When inputs are interpreted as uncontrolled

[4] And if such an equality occurs it might be the result of rounding errors.

disturbances or parameters, we want to convince ourselves that such a system behaves correctly *for all inputs*. When we interpret input as control, our goal is to find an input signal that makes the system behave as desired or optimizes some performance index. In both cases we want to lift simulation practice from a single trajectory to many and to this end discretize the space of admissible inputs.

## 2 The Basic Idea

Let $\mathcal{T} = \mathbb{R}_+$ be the time domain, $X$ be a bounded subset of $\mathbb{R}^n$ (state space) and $V$ be a bounded subset of $\mathbb{R}^m$ (input space). An open dynamical system is a system whose dynamics is defined by

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{v}).$$

We use $\mathcal{S}(V)$ to denote all $V$-valued signals, i.e. functions from $\mathcal{T}$ to $V$, without putting any restrictions on them. Every input signal $\psi \in \mathcal{S}(V)$ and initial state $\mathbf{x} \in X$ induces a behavior $\xi(\mathbf{x}, \psi) : \mathcal{T} \to X$ and we use the notation

$$\mathbf{x} \xrightarrow{\psi,t} \mathbf{x}'$$

to denote the fact that $\xi(\mathbf{x}, \psi)[t] = \mathbf{x}'$, that is, $\psi$ steers the system from $\mathbf{x}$ to $\mathbf{x}'$ at time $t$. Our goal is to show that for every $\psi \in \mathcal{S}(V)$, the trajectory $\xi(\psi)$ behaves correctly, or, in other words, that the set of states reachable from $\mathbf{x}$,

$$\mathcal{R}(\mathbf{x}) = \{\mathbf{x}' : \exists \psi \in \mathcal{S}(V)\ \exists t \in \mathcal{T}\ \mathbf{x} \xrightarrow{\psi,t} \mathbf{x}'\},$$

does not intersect $P$.

For each individual input signal the problem can be reduced to a simulation of a closed system. If we could conduct a simulation with each and every such signal, the problem could be solved. However, the set of input signals is very uncountable, consisting of $(2^{\aleph_0})^{2^{\aleph_0}}$ elements, and even if we restrict it to more reasonable sub-classes, such as measurable, piecewise-continuous or even smooth functions from $\mathcal{T}$ to $V$, its size excludes the possibility of exhaustive simulation.

Our first step is to discretize the set of input signals by discretizing both time and space. To avoid notational complications we assume $V$ to be a hyper-rectangle that fits a uniform discrete grid.[5]

**Definition 1 ($(\delta, \varepsilon)$-Discretization).** *A $(\delta, \varepsilon)$-discretization of $\mathcal{S}(V)$ consists of a discrete time domain*

$$\mathcal{T}_\delta = \{n\delta : n \in \mathbb{N}\}$$

*and a discrete input space*

$$V_\varepsilon = V \cap \{(n_1\varepsilon, \ldots n_m\varepsilon) : (n_1, \ldots n_m) \in \mathbb{Z}^m\}.$$

[5] Grids that are non-uniform inside a dimension or have different scales at each dimension pose no problem for the techniques described in this paper.

*A discrete input signal is a function $\psi' : \mathcal{T}_\delta \to V_\varepsilon$ which induces naturally a piecewise-constant signal $\bar{\psi} : \mathcal{T} \to V_\varepsilon$ defined for every $r \in \mathbb{R}_+$ as $\bar{\psi}[r \cdot \varepsilon] = \psi'[\lfloor r \rfloor \cdot \varepsilon]$. We denote the set of all such signals by $\mathcal{S}_\delta(V_\varepsilon)$.*

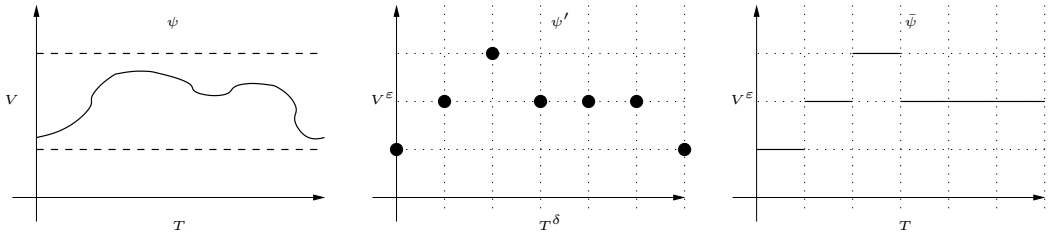


**Fig. 2.** A signal $\psi$, its discretization $\psi'$ and the induced piecewise-constant signal $\bar{\psi}$.

The whole situation is illustrated in Figure 2. Since $\mathcal{S}_\delta(V_\varepsilon)$ is a subset of $\mathcal{S}(V)$, the set of states reachable by piecewise-constant signals,

$$\mathcal{R}_{\delta,\varepsilon}(\mathbf{x}) = \{\mathbf{x}' : \exists \psi \in \mathcal{S}_\delta(V_\varepsilon)\ \exists t \in \mathcal{T}\ \mathbf{x} \xrightarrow{\psi,t} \mathbf{x}'\},$$

is a subset of $\mathcal{R}(\mathbf{x})$. However, for most cases of practical interest, we can find for every signal $\psi \in \mathcal{S}(V)$ a $(\delta,\varepsilon)$-discretization and a piecewise-constant signal $\bar{\psi} \in \mathcal{S}_\delta(V_\varepsilon)$ as close to $\psi$ as we want (in terms of some appropriate metric), by making $\delta$ and $\varepsilon$ small enough. For well-behaved systems, this means that for every reachable point in $\mathcal{R}$ we can find a close point in $\mathcal{R}_{\delta,\varepsilon}$. Moreover, we can restrict further the set of reachable states to those that are reached at discrete time points,

$$\bar{\mathcal{R}}_{\delta,\varepsilon}(\mathbf{x}) = \{\mathbf{x}' : \exists \psi \in \mathcal{S}_\delta(V_\varepsilon)\ \exists t \in \mathcal{T}_\delta\ \mathbf{x} \xrightarrow{\psi,t} \mathbf{x}'\},$$

and this set approaches $\mathcal{R}_{\delta,\varepsilon}$ as $\delta$ goes to zero.The number of discretized signals of (real-time) length $t$ is $(V/\delta)^{(t/\varepsilon)}$ and one can pose some interesting questions concerning the trade-off between time and space discretization. ¿From now on we consider fixed $\delta$ and $\varepsilon$, use $\bar{\mathcal{T}}$ and $\bar{V}$ to denote the discretized domains, and assume:

**Postulate 2 (Discretized Signals)** *Solving the reachability problem with respect to $\mathcal{S}_\delta(V_\varepsilon)$, that is, computing $\mathcal{R}_{\delta,\varepsilon}$ or even $\bar{\mathcal{R}}_{\delta,\varepsilon}$ instead of $\mathcal{R}$, is interesting.*

The set $\mathcal{S}_\delta(V_\varepsilon)$ is "isomorphic" to one of the most fundamental structures of computer science, namely the set $\bar{V}^*$ of all sequences over a finite alphabet, also known as the free monoid generated by $\bar{V}$. From now on we will use the $\bar{V}^*$ notation for signals, with a sequence like $\mathbf{v}_1\mathbf{v}_2\mathbf{v}_1$ corresponding to a signal of length $3\delta$ whose value is $\mathbf{v}_1$ at $[0,\delta)$, $\mathbf{v}_2$ at $[\delta,2\delta)$ and $\mathbf{v}_1$ at $[2\delta,3\delta)$. The

juxtaposition of signals corresponds to concatenation of sequences, e.g. $\mathbf{v}_1\mathbf{v}_2\mathbf{v}_1 \cdot \mathbf{v}_1\mathbf{v}_1 = \mathbf{v}_1\mathbf{v}_2\mathbf{v}_1\mathbf{v}_1\mathbf{v}_1$. We denote by $\bar{V}^k$ the set of sequences of length $k$, with $\bar{V}^0$ being the set consisting of the empty sequence which corresponds to a zero-duration signal, and let $\bar{V}^{\leq k} = \bigcup_{i=1}^{k} \bar{V}^i$. The set $\bar{V}^*$ can be visualized as a tree rooted in the empty word where every node has $|\bar{V}|$ successors as in Figure 3-(a). The trajectories induced by elements of $\bar{V}^*$ on the state space inherit the same structure as can be seen in Figure 3-(b).

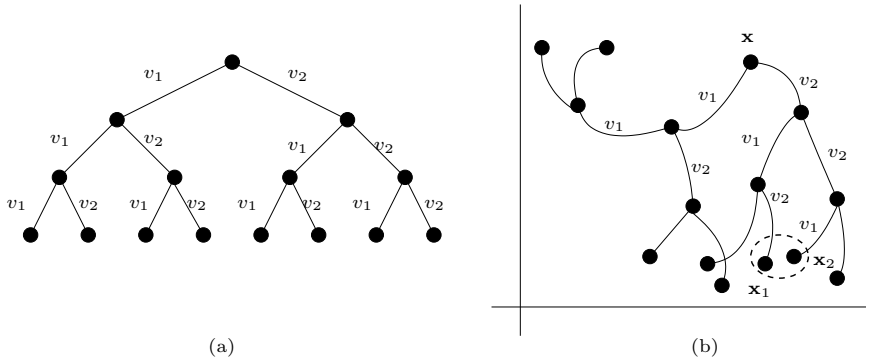


**Fig. 3.** (a) A finite fragment of the $\bar{V}^*$ tree; (b) The trajectories it induces on $X$ starting from point $\mathbf{x}$.

Endowing the input space and its induced trajectory space with a tree structure opens new possibilities for state-space exploration techniques inspired from the enormous amount of work on graph searching procedures. The input and state spaces can be explored in breadth-first, depth-first or best-first search regimes, with or without user intervention. Many heuristics developed for test-case generation or for algorithmic debugging can be applied, as well as probabilistic techniques. Moreover, if we interpret $V$ as control rather than disturbances, similar search techniques can be used to synthesize controllers that steer systems into goal states. In the rest of the paper we demonstrate the potential of this approach on the problem of computing reachable states under all disturbance signals. From now on we work in discrete time, i.e. we consider the problem computing $\bar{\mathcal{R}}_{\delta,\varepsilon}$.

## 3 Simulation-guided Reachability

In this section we develop a new algorithm for computing an approximation of reachable states of an open system. Unlike "traditional" approaches for solving this problem, e.g. [KV97,V98,ABDM00,CK03], in this approach the exploration of the state space is guided by individual input signals and the trajectories they generate.

A useful algebraic notation that we will employ is to write the "action" of a sequence $\psi \in \bar{V}^*$ on a state $\mathbf{x}$ as $\mathbf{x} \cdot \psi = \mathbf{x}'$, meaning that the input signal $\psi$ drives the system from $\mathbf{x}$ to $\mathbf{x}'$. This notation can be lifted naturally to sets of states and sets of inputs. The set of points reachable from $\mathbf{x}$ at time $k$ is $R^k(\mathbf{x}) = \{\mathbf{x} \cdot \psi : \psi \in \bar{V}^k\}$ and those reachable until time $k$ is $R^{\leq k}(\mathbf{x}) = \{\mathbf{x} \cdot \psi : \psi \in \bar{V}^{\leq k}\}$. Clearly, $R^{k+1}(\mathbf{x}) = R^k(\mathbf{x}) \cdot \bar{V}$.

If we are interested only in reachability within a bounded time horizon, we can do with a finite (although exponential) number of simulations. The semigroup property $\mathbf{x} \cdot (\psi \cdot \mathbf{v}) = (\mathbf{x} \cdot \psi) \cdot \mathbf{v}$ allows us to "re-use" partial simulation; that is, instead of running two simulations for $\psi \cdot \mathbf{v}_1$ and $\psi \cdot \mathbf{v}_2$ starting from the initial state, we can simulate with $\psi$ to reach a point $\mathbf{x}'$ and then simulate from $\mathbf{x}'$, once with $\mathbf{v}_1$ and once with $\mathbf{v}_2$. In other words, we can compute $R^{k+1}(\mathbf{x})$ directly from $R^k(\mathbf{x})$ using the following standard breadth-first algorithm, which has the property that at the end of the $k^{th}$ iteration of the main loop the set *Reached* is equal to $R^{\leq k}(\mathbf{x})$ which is exactly the set of points that we will encounter if we run $|\bar{V}|^k$ simulations.

**Algorithm 1 (Reachability for Discretized Input Signals)**

*Reached:=Waiting:=*$\{\mathbf{x}_0\}$*;New:=*$\emptyset$*;*
**Repeat** $k = 0, 1, \ldots$
  **For** *each* $\mathbf{x} \in$ *Waiting*
    **For** *each* $\mathbf{v} \in \bar{V}$
      *Compute* $\mathbf{x}' = \mathbf{x} \cdot \mathbf{v}$*;*
      *Insert* $\mathbf{x}'$ *into New;*
    *Remove* $\mathbf{x}$ *from Waiting;*
  *Waiting:=New; Reached:=Reached*$\cup$*New; New:=*$\emptyset$*;*
**Forever**

For unbounded time (or just for large $k$), even Postulate 1 is not going to help us because the number of trajectories to simulate grows indefinitely (and exponentially). We borrow the following observation from automata theory, which is just another instance of the semigroup property: If $\mathbf{x} \cdot \psi_1 = \mathbf{x} \cdot \psi_2$, then for every $\mathbf{v}$ we have $\mathbf{x} \cdot (\psi_1 \cdot \mathbf{v}) = \mathbf{x} \cdot (\psi_2 \cdot \mathbf{v})$ and the simulations with extensions of $\psi_2$ need not be performed since all of them will be "represented" by extensions of $\psi_1$. Thus, we could modify the algorithm slightly by not inserting to *New* points that are already in *New* or in *Reached*. For finite automata with $n$ states this usually results in a dramatic decrease of the number of input sequences needed to reach all states from $|\bar{V}|^n$ to $O(n)$.

For continuous systems, however, strict equality is rare and should be replaced by a weaker notion. Looking, for example, at the trajectories of Figure 3-(b) we see that inputs $\mathbf{v}_2\mathbf{v}_1\mathbf{v}_2$ and $\mathbf{v}_2\mathbf{v}_2\mathbf{v}_1$ lead to two neighboring points $\mathbf{x}_1$ and $\mathbf{x}_2$. For certain systems and under certain conditions we can guarantee that for every input $\psi$, $\mathbf{x}_1 \cdot \psi$ will remain close to $\mathbf{x}_2 \cdot \psi$, and hence the points reachable from $\mathbf{x}_1$ and their neighborhoods can "represent" those reachable from $\mathbf{x}_2$

without the latter being actually computed. This has the potential of slowing down significantly the explosion in the number of explored trajectories.

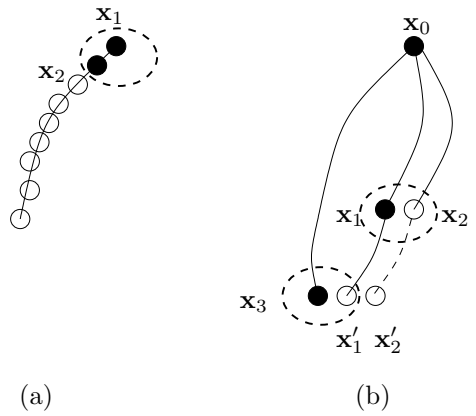


**Fig. 4.** (a) Someone should represent your ancestors; (b) Nearness is not transitive.

A naive application of this idea will discard (i.e. not insert into $New$) points that are close to previously-reached points. This will not work, however, as the two following types of counter-examples show. In the simpler one, shown in Figure 4-(a), we have a slow system such that the successor $\mathbf{x}_2 = \mathbf{x}_1 \cdot \mathbf{v}$ is close to $\mathbf{x}_1$ and the simulation is stopped without any other representative of the trajectory that goes further away from $\mathbf{x}_1$. This problem, which is the same as the problem of deciding when to stop a single simulation, could be easily prevented by not discarding a point just because it is in the neighborhood of its predecessor.

The second counter-example in Figure 4-(b) is more problematic. Suppose that $\mathbf{x}_1$ and $\mathbf{x}_2$ are two close points in $R^k(\mathbf{x}_0)$ for some $k$ and we decide not to explore $\mathbf{x}_2$ further and let it be represented by $\mathbf{x}_1$. At some later time $k'$ a successor $\mathbf{x}'_1$ of $\mathbf{x}_1$ is close to some $\mathbf{x}_3$ and is discarded, but the successor $\mathbf{x}'_2$ of $\mathbf{x}_2$ is not close to $\mathbf{x}_3$ and is not represented by it. In other words, nearness, unlike equality, is not transitive.

To overcome this problem we move from a point-based to a neighborhood-based algorithm, where a neighborhood $N(\mathbf{x})$ is a set consisting of all points close to $\mathbf{x}$ according to some metric (a metric which may change during the system evolution). The problem is then re-formulated as finding $\bar{\mathcal{R}}_{\delta,\varepsilon}(N(\mathbf{x}_0))$, all points reachable from an initial neighborhood. The action of an input $\mathbf{v}$ on a neighborhood $N(\mathbf{x})$, denoted by $N(\mathbf{x}) \cdot \mathbf{v}$ is the result of applying $\mathbf{v}$ to all elements of $N(\mathbf{x})$, yielding a neighborhood $N'(\mathbf{x}')$ of $\mathbf{x}' = \mathbf{x} \cdot \mathbf{v}$. Algorithm 1 when applied to $N(\mathbf{x}_0)$ will produce at each step $k$ a set of neighborhoods whose union is exactly the set of points reachable from $N(\mathbf{x}_0)$ at time $k$.

Now we can record the fact that a point represents another point by increasing its neighborhood, that is, whenever two close points are "merged" their neighborhoods are merged into a neighborhood that contains both. This way an algorithm for over-approximating $\bar{\mathcal{R}}_{\delta,\varepsilon}(N(\mathbf{x}_0))$ is obtained. We will first describe the exploration technique in abstract terms, without specifying the actual form of neighborhoods and operations. The algorithm needs three operations:

1. $Next$ which produces an over-approximation of the action of an input on a neighborhood, that is, a function satisfying
$$Next(N(\mathbf{x}), \mathbf{v}) \supseteq N(\mathbf{x}) \cdot \mathbf{v}.$$
2. $JoinTest$, a function that selects from a set of neighborhoods one which should be merged with a given neighborhood. For example, $JoinTest(Set, N(\mathbf{x}))$ may return the neighborhood $N'(\mathbf{x}') \in Set$ which minimizes some distance between $N(\mathbf{x})$ and $N(\mathbf{x}')$. It may return the empty set in case no elements of $Set$ is close to $N(\mathbf{x})$.
3. $Join$, merging two neighborhoods into a containing neighborhood, a function satisfying
$$Join(N_1(\mathbf{x}_1), N_2(\mathbf{x}_2)) \supseteq N_1(\mathbf{x}_1) \cup N_2(\mathbf{x}_2).$$

**Algorithm 2 (Reachability with Neighborhoods)**

*Reached:=Waiting:=*$\{N(\mathbf{x}_0)\}$*; New:=*$\emptyset$*;*
**Repeat** $k = 0, 1, \ldots$
  **For** *each* $N(\mathbf{x}) \in$ *Waiting*
    **For** *each* $\mathbf{v} \in \bar{V}$
      *Compute* $N'(\mathbf{x}') = Next(N(\mathbf{x}), \mathbf{v})$*;*
      $\hat{N}(\hat{\mathbf{x}}) :=$ *JoinTest(Reached*$\cup$*New,*$N'(\mathbf{x}')$*);*
      **If** $\hat{N}(\hat{\mathbf{x}}) \neq \emptyset$ **Then**
        **If** $N'(\mathbf{x}') \not\subseteq \hat{N}(\hat{\mathbf{x}})$ **Then**
          *Remove* $\hat{N}(\hat{\mathbf{x}})$ *from New and from Reached*
          **If** $\hat{N}(\hat{\mathbf{x}}) \subseteq N'(\mathbf{x}')$ **Then**
            *Insert* $N'(\mathbf{x}')$ *into New*
          **Else**
            *Compute* $N^*(\mathbf{x}^*) = Join(N'(\mathbf{x}'), \hat{N}(\hat{\mathbf{x}}))$*;*
            *Insert* $N^*(\mathbf{x}^*)$ *into New*
      **Else**
        *Insert* $N'(\mathbf{x}')$ *into New;*
    *Remove* $N(\mathbf{x})$ *from Waiting;*
  *Waiting:=New; Reached:=Reached*$\cup$*New; New:=*$\emptyset$*;*
**Forever**

The $JoinTest$ function for merging neighborhoods can be subject to heuristic considerations related to the trade-off between accuracy (the degree of over-approximation) and complexity (the number of points).

**Claim 1 (Coverage)** *For every $k$*

$$R^{\leq k}(N(\mathbf{x}_0)) \subseteq \bigcup_{N(\mathbf{x}) \in Reached} N(\mathbf{x})$$

*holds at the end of the $k^{th}$ iteration of the main loop of Algorithm 2.*

## 4 Linear Systems in Discrete time

In this section we describe a concrete version of the algorithm for linear time-invariant (LTI) systems in discrete time, i.e., systems with state equations of the form

$$\mathbf{x}_{k+1} = \Phi \mathbf{x}_k + \Gamma \mathbf{v}_k,$$

with $\Phi$ nonsingular.[6] As neighborhoods we use ellipsoids, each parameterized by a positive definite symmetric matrix $Q \in \mathbb{R}^{n \times n}$, i.e.

$$N_Q(\mathbf{x}) = \{\hat{\mathbf{x}} : \|\hat{\mathbf{x}} - \mathbf{x}\|_{Q^{-1}} \leq 1\},$$

where $\|\mathbf{x}\|_Q$ is defined as $\mathbf{x}^T Q \mathbf{x}$. The action of an input $\mathbf{v}$ on a neighborhood is defined as

$$Next(N_Q(\mathbf{x}), \mathbf{v}) = N_{Q'}(\mathbf{x}')$$

with $\mathbf{x}' = \mathbf{x} \cdot \mathbf{v}$ and

$$Q' = \Phi Q \Phi^T.$$

The join operation for merging of two ellipsoidal neighborhoods is defined as

$$Join(N_{Q_1}(\mathbf{x}_1), N_{Q_2}(\mathbf{x}_2)) = N_{\tilde{Q}}(\tilde{\mathbf{x}}),$$

where $N_{\tilde{Q}}(\tilde{\mathbf{x}})$ is the minimum volume ellipsoid containing $N_{Q_1}(\mathbf{x}_1)$ and $N_{Q_2}(\mathbf{x}_2)$. Efficient LMI-based numerical routines exist for computing $\tilde{Q}$ and $\tilde{\mathbf{x}}$ [BGFB94]. The function $JoinTest(Set, N(\mathbf{x}))$ returns the ellipsoid in $Set$ whose center minimizes the distance to $\mathbf{x}$, provided that this distance is smaller than a given threshold (a tunable parameter of the algorithm).

It is not hard to see that this concrete version of Algorithm 2 satisfies Claim 1 and computes an over-approximation of the reachable set. It is important to distinguish this technique from other reachability algorithms [CK99,BM99,ABDM00] including those that use ellipsoids for representing reachable sets [KV97,V98,BT00]. Typically such algorithms are inherently "breadth-first" where at every step $k$ there is one object (a polyhedron or an ellipsoid) that represents all reachable points for all inputs at time $k$. This set is computed by running an optimization over the input domain $V$ to find the input that takes the system mostly "outwards". Our approach is more enumerative in nature and may use several ellipsoids at each time step. Our technique can be tuned by modifying the

[6] We note that the matrix $\Phi$ is always nonsingular when the discrete-time system is obtained by sampling a continuous LTI dynamic system.

neighborhood size, the *JoinTest* definition, and the *Next* and *Join* operations. Working with small neighborhoods makes the algorithm closer to exhaustive simulation while larger neighborhoods make it closer to "standard" reachability algorithms. Like those algorithms, ours can be easily adapted to hybrid automata by intersecting the reachable sets with transition guards.

## 5 Experimental Results

We have implemented the algorithm in MATLAB and we illustrate its behavior with the following two examples.

### 5.1 Servo System Example

For the servo system shown in figure 5 we want to show that for all reference inputs, the distance between the actual value and the reference remains within a given bound. The system has first-order plant dynamics and a first-order reference signal filter. In the figure, $x_1$ is the plant output, $v$ is the reference input, $x_2$ is the output of the reference signal filter, and $d = x_2 - x_1$ is the error. The continuous time dynamic equations are given by $\dot{\mathbf{x}} = A\mathbf{x} + Bv$ with

$$A = \begin{bmatrix} -g & g \\ 0 & -\frac{1}{\tau} \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ \frac{1}{\tau} \end{bmatrix}$$

For piecewise constant inputs with a sampling period of $\delta$, the sampled system dynamics are

$$\mathbf{x}_{k+1} = \Phi\mathbf{x}_k + \Gamma v_k$$
$$\Phi = e^{A\delta} \qquad \Gamma = A^{-1}(e^{A\delta} - I)B$$

For $g = 10$, $\tau = 0.1$, and $\delta = 0.1$ this yields

$$\Phi = \begin{bmatrix} 0.368 & 0.368 \\ 0.000 & 0.368 \end{bmatrix} \qquad \Gamma = \begin{bmatrix} 0.264 \\ 0.632 \end{bmatrix}$$

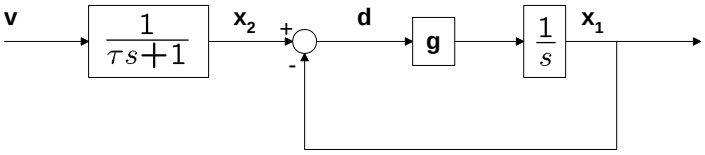


**Fig. 5.** A simple servo system.

We wish to verify that the system does not reach the set

$$P = \{\mathbf{x} : |x_1 - x_2| > 1\}$$

for all sequences in $\bar{V}^*$, $\bar{V} = \{0.0, 0.5, 1.0\}$. Figure 6-(a) shows one sample 8-step trajectory of the system and Figure 6-(b) shows the states reached along the trajectories generated by all elements of $\bar{V}^8$ starting from $\mathbf{x}_0 = [0\ 0]^T$. The boundaries of $P$ are the diagonal lines in the figure. The number of simulation points is $\sum_{k=0}^{8} 3^k = 9841$. Figure 6-(c), generated by exhaustive simulation (i.e., iterative application of the *Next* operator), shows the set of states reachable from all points within $N_{10I}(\mathbf{x}_0)$ (a circle with radius $\sqrt{0.1}$ centered at the origin). Figure 6-(d) shows the result of applying 8 iterations of Algorithm 2 to the example, starting with the same initial set. Only 273 *Next* operations were required and upon completion of the procedure, *Reach* contained only the 21 ellipsoids shown in the figure.

Figure 7 shows a comparison of the computation time for exhaustive simulation versus our algorithm for the servo system. Note that that the computation time of our algorithm grows only linearly with the number of time increments in contrast to the exponential inherent in exhaustive simulation. For small numbers of steps ($k < 8$) exhaustive simulation requires less computation time due to the LMI optimizations in the *Join* operation of our algorithm. We are currently investigating other examples of stable systems hoping to reproduce the phenomenon of sub-exponential growth in the number of computed points.

### 5.2 A Marginally Stable System

We next consider a four-dimensional system with a four-dimensional input space derived from the continuous-time system $\dot{\mathbf{x}} = A\mathbf{x} + \mathbf{v}$ with

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -4 & 0 \end{bmatrix}$$

and

$$\mathbf{v} \in V = [-1, 1] \times [-0.1, 0.1] \times [-1, 1] \times [-0.1, 0.1].$$

This example appeared first in [KV97], p. 279, where it was subject to an ellipsoid-based reachability algorithm and was treated later in [D00], p. 83, using polyhedra. The system is marginally stable, but the inputs can make it diverge.

As a discretized input space for the continuous-time model we use the vertices, i.e.

$$V' = \{-1, 1\} \times \{-0.1, 0.1\} \times \{-1, 1\} \times \{-0.1, 0.1\}.$$

Time-discretization with $\delta = 0.2$ yields

$$\mathbf{x}_{k+1} = \Phi\mathbf{x}_k + \mathbf{v}_k,$$

where $\Phi = e^{0.2A}$, and $\mathbf{v}_k \in \bar{V}$, where

$$\bar{V} = \{(\Phi - I)\mathbf{v} : \mathbf{v} \in V'\}$$

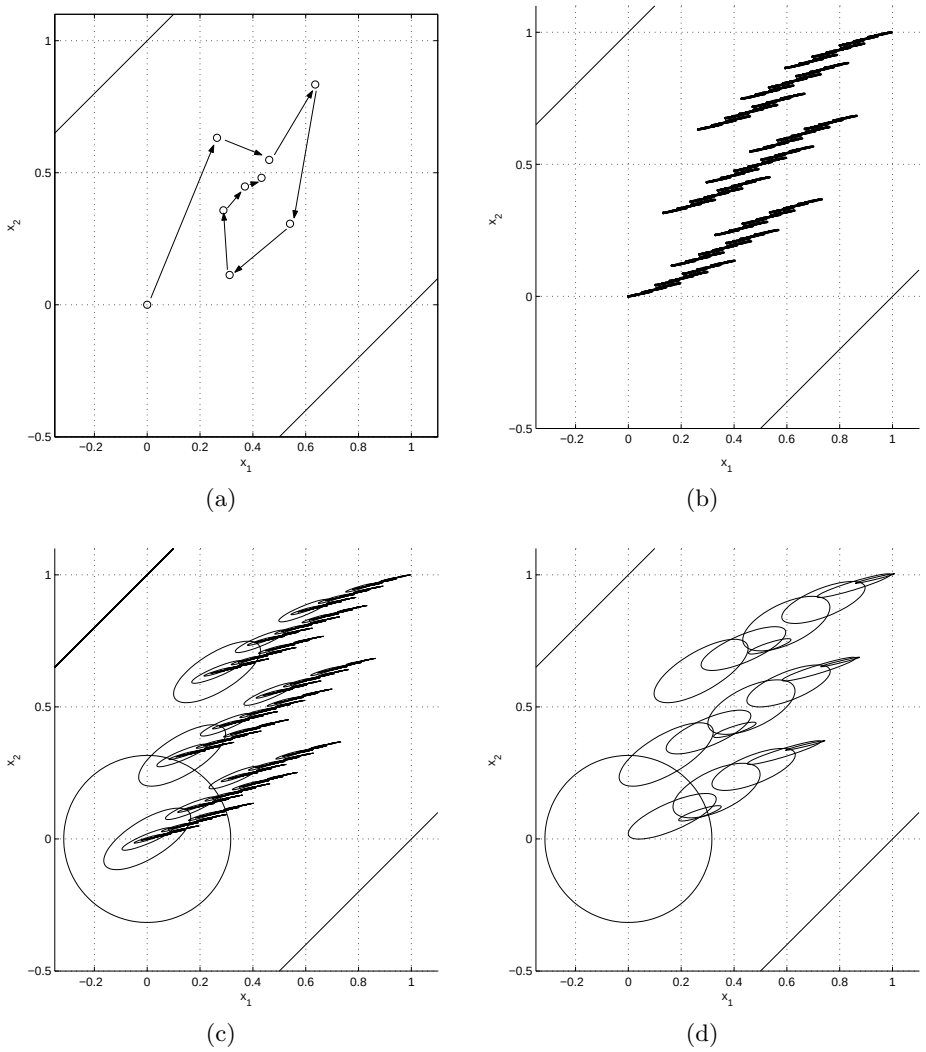


**Fig. 6.** (a) A sample trajectory of the system under the input $\psi = 1.0 \cdot 0.5 \cdot 1.0 \cdot 0.0 \cdot 0.0 \cdot 0.5 \cdot 0.5 \cdot 0.5$; (b) The states reachable from $(0, 0)$ for all 8-step trajectories generated by exhaustive simulation; (c) states reachable from a circle around $(0, 0)$ for all these trajectories as computed by exhaustive simulation; (d) The over-approximation of the reachable states as computed by our algorithm.

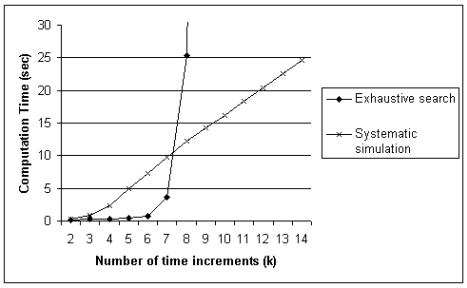


**Fig. 7.** Computation times for exhaustive simulation and our algorithm for the servo example

Figure 8 shows the over approximation of the set of reachable states (projected onto third and fourth dimensions) generated with Algorithm 2, computed for 3 time steps, starting with the top left plot showing the initial ellipsoid, which is a hyper-sphere of radius 0.01. Note that the ellipsoids that appear to be subsets of other ellipsoids are not subsets in the full four-dimensional space. For each time increment, the set of reachable states grows due to the fact that the system is marginally stable.

Figure 9 shows a comparison of the computation times between exhaustive simulation and systematic simulation. For this system, the time required by systematic simulation is exponential in the number of time increments. For the servo system many paths in the search tree terminate because new ellipsoids ($N'(x')$ in Algorithm 2) land completely within existing ellipsoids; only few paths in the marginally stable example terminate.

## 6 Conclusions

We have developed a framework for systematic simulation of open continuous dynamical system. By relating the structure of the discretized input space with the structure of the induced trajectory space, state-space exploration can be done as a variation of graph search algorithms. As a first application of this framework we developed a new algorithm for computing reachable sets. We intend to continue this work by improving the algorithm and comparing its performance with that of other algorithms, by developing other search techniques and by applying these ideas to controller synthesis.

**Acknowledgments** We would like to thank Eugene Asarin and some anonymous referees for many useful comments concerning this paper. This work was partially supported by the EC project IST-2001-33520 CC (Control and Computation) by DARPA (contracts F33615-02-C-0429 and F33615-00-C-1701), ARO (DAAD19-01-1-0485) and NSF (CCR-0121547).

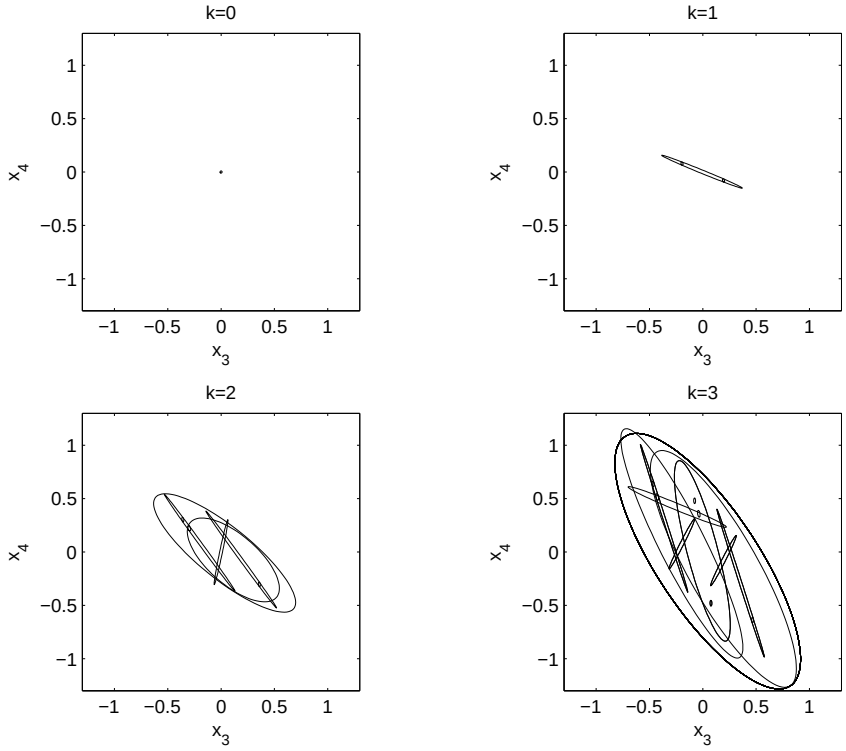


**Fig. 8.** Reachable set for the marginally stable example from 0 to 3 time increments, projected on the first two variables.

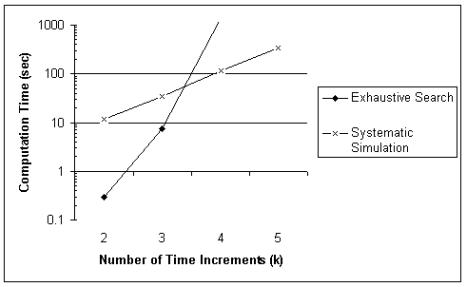


**Fig. 9.** Computation times for exhaustive simulation method and systematic simulation for the marginally stable example

## References


[D00] T. Dang, *Verification and Synthesis of Hybrid Systems*, PhD thesis, INPG, Grenoble, 2000.

[ABDM00] E. Asarin, O. Bournez, T. Dang and O. Maler, Reachability Analysis of Piecewise-Linear Dynamical Systems, in B. Krogh and N. Lynch (Eds.), *Hybrid Systems: Computation and Control*, 20-31, LNCS 1790, Springer, 2000.

[BM99] A. Bemporad and M. Morari, Verification of Hybrid Systems via Mathematical Programming, in F.W. Vaandrager and J.H. van Schuppen (Eds.), *Hybrid Systems: Computation and Control*, 31-45, LNCS 1569, Springer, 1999.

[BT00] O. Botchkarev and S. Tripakis, Verification of Hybrid Systems with Linear Differential Inclusions Using Ellipsoidal Approximations, in B. Krogh and N. Lynch (Eds.), *Hybrid Systems: Computation and Control*, 73-88, LNCS 1790, Springer, 2000.

[BGFB94] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, SIAM, 1994.

[CK99] A. Chutinan and B.H. Krogh, Verification of Polyhedral Invariant Hybrid Automata Using Polygonal Flow Pipe Approximations, in F.W. Vaandrager and J.H. van Schuppen (Eds.), *Hybrid Systems: Computation and Control*, 76-90, LNCS 1569, Springer, 1999.

[CK03] A. Chutinan and B.H. Krogh, Computational Techniques for Hybrid System Verification, *IEEE Transactions On Automatic Control*, 2003.

[KV97] A. Kurzhanski and I. Valyi, *Ellipsoidal Calculus for Estimation and Control*, Birkhauser, 1997.

[M98] O. Maler, A Unified Approach for Studying Discrete and Continuous Dynamical Systems, *Proc. CDC'98*, IEEE, 1998.

[M01] O. Maler, Control from Computer Science, *IFAC Symposium Nonlinear Control (NOLCOS'01)*, Elsevier, 2001.

[PLY99] G. Pappas, G. Lafferriere and S. Yovine, A New Class of Decidable Hybrid Systems, in F.W. Vaandrager and J.H. van Schuppen (Eds.), *Hybrid Systems: Computation and Control*, LNCS 1569, 29-31, Springer, 1999.

[V98] P. Varaiya, Reach Set Computation using Optimal Control, *Proc. KIT Workshop*, Verimag, Grenoble, 1998.