Algorithms for Synthesizing Safe Sets of Operation For Embedded Systems

Abhishek Dubey

Institute for Software Integrated Systems
Department of Electrical Engineering and Computer Science
Vanderbilt University, Nashville, TN 37203, USA

Abstract

A large number of embedded computing systems are modeled as hybrid system with both discrete and continuous dynamics. In this paper, we present algorithms for analyzing nonlinear time-invariant continuous-time systems by employing reachability algorithms. We propose synthesis algorithms for finding sets of initial states for the continuous dynamical systems so that temporal properties, such as safety and liveness properties, are satisfied. The initial sets produced by the algorithms are related to some classical concepts for continuous dynamical systems, such as invariant sets and domains of attraction.

1. Introduction

The information and digital revolution is changing the scope where traditional dynamical system theory was applied. Integration of information processing with physical processes in embedded systems necessitate the use of hybrid system theory [14], which is a study of systems that contain both physical and computation processes, for purposes of analyses. It is a bridge between traditional control systems theory, which is inadequate for capturing computational aspects of embedded system, and classical computer science theory, which is inadequate for capturing the physical processes related to embedded systems.

Formalizing methodologies for development of embedded systems is one of the most formidable challenges faced by industry and academia today. Embedded systems are required to meet multiple design objectives, while satisfying the requirements for system performance and system reliability. Due to limitation of physical resources such as power sources, traditional approach of maintaining exaggerated safety-margins in order to meet the requirements is not acceptable in embedded systems design.

In the last decade, there has been a lot of progress in this field. Algorithmic approach for the analysis of hybrid systems are based on state exploration principle. These methods are used for both verification [1, 5, 10, 4] and controller synthesis [21, 8, 2]. These approaches can be grouped into two classes, reductionist and symbolic.

Reductionist methods[11] reduce a hybrid system to an equivalent finite transition system using bisimulation and explore the reduced finite state space. Since the quotient space is finite, algorithmic analysis method applied to the quotient space is guaranteed to terminate in finite steps.

Symbolic algorithms analyze the hybrid system by directly exploring the infinite continuous state space using approximations. These algorithms use the concept of reachable sets and set operations such as union, intersection, difference and complement for analysis. They borrow heavily from the field of real analysis and algebraic topology. Main concern with symbolic methods is termination. Methods that may or may not terminate are called *semi-algorithms*.

Problems in analyzing Hybrid systems emanate from the difficulty of representation and manipulation of higher dimensional continuous sets, (a) because they require more memory for storage, (b) only certain restricted classes of continuous sets can be exactly represented and manipulated symbolically [1, 13]. For most cases, approximate representation methods have to be used for representing and manipulating continuous state sets. Due to finite precision in representation and computation, only approximate results can be obtained.

A number of approach for approximate representation of continuous sets exist. For example, polygonal projections [10], flow-pipes [7, 19], ellipsoids [6, 12], griddy polyhedra [4], level sets [17].

In this paper, we are concerned with developing generic algorithms that can be used for determining sets of initial states for nonlinear time-invariant continuous-time systems so that some temporal properties, such as safety and liveness properties, are satisfied, by directly exploring the infinite state space using any of the symbolic hybrid system tool. With these algorithms we can use structures like multi-dimension lists to extend these algorithms in future and use in hybrid systems.

The initial sets produced by these algorithms are re-

978-0-7695-3602-6/09 $25.00 © 2009 IEEE
DOI 10.1109/ECBS.2009.43

lated to classical concepts for continuous dynamical systems, such as invariant sets and domains of attraction. Later we show some implementation results by using the symbolic methods in [17, 4]. These methods divide a region of Cartesian space in to grids, which result into finite representation of infinite state space. This framework makes set operations and more advanced constructive solid geometry operations straightforward to apply. Since the representations and operations of state sets cannot be exact, only approximate results can be produced.

2. Continuous Dynamics

In this paper, we consider a continuous dynamical system $\Sigma_c = (X, X_s, f)$ where the state space is $X (= \Re^n)$, the state vector is $x \in X$, the initial set is $X_s \subseteq X$ and the vector field is $f : X \to \Re^n$.

The evolution of the state is specified by an ordinary differential equation (ODE):

$$\dot{x}(t) = f(x(t)),\ x(0) = x_s,\ t \geq 0 \tag{1}$$

with the initial state $x_s \in X_s$.

We assume that the vector field of (1) is *Lipschitz* continuous over the state space X. By a solution of (1) we mean a continuously differentiable function of time $x(t)$ satisfying

$$x(t) = x_s + \int_0^t f(x(\tau))d\tau. \tag{2}$$

Since the vector field is *Lipschitz* continuous over X, there is exactly one solution for (1) of the form of (2) in X (see [18]) for $t \geq 0$.

The state of the system (1) at time t starting from x_s at $t = 0$ is called the *flow* and is denoted by $\phi(t, x_s)$. We assume that the flow is globally well-defined, *i.e.* $\phi(t, \cdot)$ is well-defined for all t.

The flow satisfies the following conditions: $\forall x_s \in X_s$ $\forall t, t' \geq 0$ such that

$$\begin{array}{ll} a. & \textit{initial condition},\ \phi(0, x_s) = x_s \\ b. & \textit{dynamics},\ \dot{\phi}(t, x_s) = f(\phi(t, x_s)) \\ c. & \textit{continuity},\ \phi(t', \phi(t, x_s)) = \phi(t' + t, x_s) \\ d. & \textit{invertible},\ \phi^{-1}(-t, x_s) = \phi(t, x_s) \end{array} \tag{3}$$

Lemma 1 *Lipschitz continuity of $f(\cdot)$ implies that the flow $\phi(t, \cdot)$ will always preserve the topological properties such as connectedness, compactness, openness or closeness of the initial condition. In other words, $\phi(t, \cdot)$ is a homeomorphism for each t.*

3. Reachable Sets and Their Properties

3.1. Forward Reachable Set

The forward reachable set, denoted $Post_{cI}(P)$, is a collection of states in X and each state can be reached by a state of a set $P \subseteq X$ in some time specified by a time set $I \subseteq \Re_+$. Hence, the successor of P can be expressed as:

$$\begin{aligned} Post_{cI}(P) \quad &= \{x' \in X | \exists x\ \exists t : x \in P \wedge t \in I \\ &\qquad \wedge x' = \phi(t, x)\} \end{aligned} \tag{4}$$

where $Post_{cI} : 2^X \to 2^X$, in which 2^X denotes the power set of X.

3.2. Backward Reachable Set

Similarly, one can define the backward reachable set, denoted $Pre_{cI}(P)$, as a collection of states in X and each can reach some states of a set $P \subseteq D$ in some time specified by $I \subseteq \Re_+$. The predecessor of P can be expressed as:

$$\begin{aligned} Pre_{cI}(P) \quad &= \{x' \in X | \exists x\ \exists t : x \in P \wedge t \in I \\ &\qquad \wedge x' = \phi^{-1}(t, x)\} \end{aligned} \tag{5}$$

where $Pre_{cI} : 2^X \to 2^X$.

Both definitions can be rewritten as

$$Post_{cI}(P) = \bigcup_{x \in P} \bigcup_{t \in I} \phi(t, x) \text{ and} \tag{6}$$

$$Pre_{cI}(P) = \bigcup_{x \in P} \bigcup_{t \in I} \phi^{-1}(t, x) \tag{7}$$

3.3. Constrained Forward and Reachable Set

For a continuous dynamical systems, there may exist some state constraints that the state variables have to satisfy. These constraints specify the domain, $D \subseteq X$, in which the system should operate for all the time. This domain is also known as *invariant* or *safe set*. We assume that the domain is connected and compact. Furthermore, once any constraint is violated, the (continuous) evolution of the state variables should be prohibited.

The constrained forward (backward) reachable set, denoted $cPost_{cI}(P)$ ($cPre_{cI}(P)$), is a collection of states in X and each state can be reached by a state of $P \subseteq X$ in some future (past) time in $I \subseteq \Re_+$ while in any of those time instants no constraint is violated.

Definition 1 *(Constrained Continuous Forward and Backward Reachable Sets) Consider a continuous dynamical system $\Sigma_c = (X, X_s, f)$. Given a set $P \subseteq X$, a time interval $I \subseteq \Re_+$ and a domain $D \subseteq X$, the constrained continuous forward reachable set, $cPost_{cI} : 2^X \to 2^X$, and backward reachable set, $cPre_{cI} : 2^X \to 2^X$, are defined as:*

$$\begin{aligned} cPost_{cI}(P) = \quad & \{x' \in X | \exists x\ \exists t : \\ & x \in P\ \wedge\ t \in I\ \wedge\ x' = \phi(t, x)\ \wedge \\ & \forall \tau : 0 \leq \tau \leq t \Rightarrow \phi_i(\tau, x) \in D\} \end{aligned} \tag{8}$$

$$cPre_{cI}(P) = \{x \in X | \exists y \; \exists t : x' \in P \wedge t \in I \wedge x = \phi^{-1}(t, x') \wedge \forall \tau : 0 \leq \tau \leq t \Rightarrow \phi_i^{-1}(\tau, x') \in D\} \quad (9)$$

Notice that $cPost_{cI}(P)$ contains the states of $Post_{cI}(P)$ except those states which have any points in their trajectories ever going outside the domain D. When no domain constraint is violated, $cPost_{cI}(P) = Post_{cI}(P)$. We can make a similar argument for $cPre_{cI}(P)$. Hence, one can show that $cPost_{cI}(P) \subseteq Post_{cI}(P)$ and $cPre_{cI}(P) \subseteq Pre_{cI}(P)$, where $P \subseteq X$, $I \subseteq \Re_+$ and $D \subseteq X$. However, given that P is connected and $t' \geq 0$, the forward constrained set, $cPost_{c\{t'\}}(P)$ at the time instant t' can be disconnected.

4. Computation of Reachable Sets

In the following, we will show an algorithm for computing the bounded-time constrained backward reachable sets. The algorithm is modified from the algorithms for the exact computation of unbounded-time backward reachable sets[4, 15].

The algorithm is symbolic and is not specific to any particular way of computation. If the representations and operations of state sets used in the algorithm are exact, the algorithm can give exact solutions.

Note: We assume that there exists a *maximal* time step $\Delta t (> 0)$ for computing the reachable sets for a specific computation method, which could be used for performing exact or approximate computation. However, for some computation methods, if a required time step is larger than Δt, the solution quality at each step could deteriorate. Moreover, the time required for computing reachable sets increase if the time step is larger. For example refer to the paper on d/dt[9], a tool that computes reachable sets for affine systems. In this tool, at each time step an over(under) approximation of set is computed. The error of approximation depends upon the chosen time step.

Recall by Lemma 1, the reachable set from an initial connected set is also connected. Therefore, we can compute the bounded time reachable sets by using union of reachable sets over small discrete time steps. We denote the bounded-time constrained reachable set computed by using the time step Δt as $cPost_{c\Delta t}(P) (= cPost_{c[0,\Delta t]}(P))$ and $cPre_{c\Delta t}(P) (= cPre_{c[0,\Delta t]}(P))$.

Given a bounded time interval $I = [0, T]$ with $0 < \Delta t \leq T < \infty$, we denote the bounded time constrained reachable sets as $cPost_{cT}(P) (= cPost_{c[0,T]}(P))$ and $cPre_{cT}(P) (= cPre_{c[0,T]}(P))$.

Algorithm 1 shows how to compute $cPre_{cT}(\cdot)$ by using $cPre_{c\Delta t}(\cdot)$.

There are two termination conditions. Firstly, the algorithm terminates if the bounded time limit has been reached or exceeded. Secondly, the termination condition occurs if

Algorithm 1 Algorithm for Computing $cPre_{cT}(\cdot)$

```
Input: $\Sigma_c = (X, X_s, f)$, $D$, $P$, $T$
Output: $cPre_{cT}(P)$
Start
    $R_0 = P$
    Repeat $k = 0, 1, 2, \cdots$
        $R_{k+1} = R_k \cup cPre_{c\Delta t}(R_k)$
    Until $(k\Delta t \geq T) \vee (R_{k+1} \subseteq R_k)$
    $cPre_{cT}(P) = R_{k+1}$
End
```

the reachable set does not grow any more. If the time step can be further reduced without affecting the solution quality, one can have the solution close to the fixed point solution, i.e. $R^* = R_{k+1} = R_k$, by adjusting the final step size. The termination condition of this algorithm is guaranteed since both Δt and T are bounded.

Algorithm for computing constrained forward reachable set is similar to the algorithm for computing backward reachable set.

5. Algorithms for Analyzing Continuous Dynamical Systems

In this section, synthesis algorithms for safety and liveness properties are presented. Connections between the results generated by the algorithms and some concepts for dynamical systems are presented.

5.1. Synthesis Algorithms for Continuous Systems

let us consider a continuous dynamical system, $\Sigma_c = (X, X_s, f)$, which is nonlinear and time-invariant. We present symbolic algorithms for analyzing the continuous dynamical systems Σ_c by directly exploring the infinite state space subject to state constraints specified by the domain $D \subseteq X$. Inspired by the algorithms developed for the controller synthesis problems for timed and hybrid systems[16, 4, 15], we propose synthesis algorithms for finding sets of initial states for the continuous dynamical systems so that safety property or liveness property is satisfied.

Consider two kinds of temporal properties defined over the trajectory space of Σ_c: $\Box F$ and it's dual $\Diamond F$ with $F \subseteq X$ as defined in [20]. Given a set $F \subseteq X$ and a trajectory generated by Σ_c, $\Box F$ and $\Diamond F$ give *True* or *False* whether the trajectory *always* stays inside F or *eventually* reaches F, respectively.

$\Box F$ is referred to *safety* property while $\Diamond F$ is referred to *liveness* property. They are formally defined as:

$$\Box F = \begin{cases} True & if\ \forall t \geq 0\ \ x(t) \in F \\ False & otherwise \end{cases} \quad (10)$$

$$\Diamond F = \begin{cases} True & if\ \exists t \geq 0\ \ x(t) \in F \\ False & otherwise. \end{cases} \quad (11)$$

$\Box F$ can be derived by using $\Box F = \neg \Diamond F^c$ where $F^c = X \setminus F$.

These temporal properties are used in specifying verification and synthesis problem for transition systems, which are generalizations of discrete systems, continuous systems as well as hybrid systems.

In a synthesis problem for Σ_c, we are interested in finding a collection of states, denoted F_s, such that $\forall x_s \in X_s$ a temporal property Ω is $True$. The collection of states that satisfies the temporary property is called the *winning states*[16, 15, 20]. In the following, synthesis problems for Σ_c with respect to $\Box F$ and $\Diamond F$ are presented.

Problem 1 ($\Box F$ Synthesis Problem for Σ_c) *Given a continuous dynamical system $\Sigma_c = (X, X_s, f)$ and a set $F \subseteq X$, we are interested in finding a set of initial conditions, $F_s \subseteq X$, such that if a state starts from any where in F_s, it can always stay inside F.*

Problem 2 ($\Diamond F$ Synthesis Problem for Σ_c) *Given a continuous dynamical system $\Sigma_c = (X, X_s, f)$ and a set $F \subseteq D$, we are interested in finding a set of initial conditions, $F_s \subseteq X$, such that if a state starts from any where in F_s, it can eventually reach F.*

In order to have nontrivial solutions for both problems, we need to further assume that for the $\Box F$ property $F_s \subseteq F$ and for the $\Diamond F$ property $F_s \supseteq F$.

We seek to solve the Problems 1 and 2 in a generic way so that the algorithms can be extended to the problems for hybrid systems. Therefore, we must find a set of concepts related to set representation and operations that is general enough for continuous, timed and hybrid systems. Motivated by the algorithms developed by [16, 15], we are interested in solving the problems by developing symbolic algorithms for Σ_c with respect to $\Box F$ and $\Diamond F$.

In [16, 15], the concept of the "immediate" predecessor for discrete system is used in the algorithms. However, this concept can hardly be generalized for continuous dynamical systems due to the fact that the notion of time is different. For continuous systems, time is dense in nature. Therefore, instead of immediate predecessor we use the whole continuous predecessor set that can lead to the current set i.e. we replace the "immediate" predecessor for discrete system with the constrained bounded-time predecessor for Σ_c.

The reason why we can do this is that in the algorithm we keep taking the *union* of these reachable sets. Even though the notion of "immediate" predecessor is not applicable for continuous systems, the algorithm can still be applied with slight modification.

We first present the a $\Diamond$-algorithm for Σ_c. We also derive a $\Box$-algorithm for Σ_c by utilizing the $\Diamond$-algorithm, since we know that the $\Box$ and $\Diamond$ properties are related by $\Box F = \neg \Diamond F^c$ where $F^c = X \setminus F$. The algorithms for solving Problem 1 and 2 are shown below.

Algorithm 2 $\Diamond$-Algorithm for Σ_c

```
Input: $\Sigma_c = (X, X_s, f)$, $D$, $F$, $T$
Output: $F_s$
Start
    $P = F$
    $R_0 = \emptyset$
    Repeat $k = 0, 1, 2, \cdots$
        $R_{k+1} = P \cup cPre_{cT}(R_k)$
    Until $R_{k+1} \subseteq R_k$
    $F_s = R_{k+1}$
End
```

Algorithm 3 $\Box$-Algorithm for Σ_c

```
Input: $\Sigma_c = (X, X_s, f)$, $D$, $F$, $T$
Output: $F_s$
Start
    $P = X \setminus F$
    $R_0 = \emptyset$
    Repeat $k = 0, 1, 2, \cdots$
        $R_{k+1} = P \cup cPre_{cT}(R_k)$
    Until $R_{k+1} \subseteq R_k$
    $F_s = X \setminus R_{k+1}$
End
```

In each algorithm, there is a termination condition. The termination condition occurs if a fixed point solution $R^* = R_{k+1} = R_k$ is reached i.e. the reachable set does not grow any more. Again, if the time interval T can be further reduced without affecting the solution quality, one can have the solution close to the fixed point solution, i.e. $R^* = R_{k+1} = R_k$, by adjusting the value of T.

In general, these algorithms do not necessary terminate since they are semi-algorithms. However, if we represent the state space using finite representative elements such as grids, the algorithms are guaranteed to terminate. However, the solution using grids is approximate but the termination of algorithm is guaranteed.

5.2. Properties of the Sets Produced by the Synthesis Algorithms for Σ_c

The solutions produced by the algorithms 2 and 3 are related to some classical concepts for continuous dynamical systems, which are *domain of attraction*, which is associated to an equilibrium point, and *invariant set*. Let us first state the definitions for equilibrium point and domain of attraction.

Definition 2 *(Equilibrium Point) Consider a continuous dynamical system $\Sigma_c = (X, X_s, f)$. A point x^* is said to be an* equilibrium point *of (1) if $f(x^*) \equiv 0$ for all $t \geq 0$.*

Each equilibrium is the fixed point of the flow and is also referred as stationary solution.

Definition 3 *(Domain of Attraction) Consider a continuous dynamical system $\Sigma_c = (X, X_s, f)$ with equilibrium point x^*. A set $N \subseteq X$ is said to be the domain of attraction of x^* if the set of initial condition $x_s \in N$ satisfies $\lim_{t\to 0} \phi(t, x_s) = x^*$*

It has been shown in literature that if x^* is an asymptotically stable equilibrium point, then the domain of attraction is an open, invariant set [18]. Moreover, the boundary is invariant as well.

Consider a stable equilibrium point x^* is inside F. By applying the $\diamond$-algorithm, if the algorithm terminates, we should be able to obtain an approximation of a domain of attraction specified by F_s provided that $F \subset F_s$. For the actual F_s which may not be computable, the set should contains all the states which can eventually reach F subject to state constraints specified by the domain D. However, since only the approximation of F_s is computed, the topological properties of the reachable sets are not necessarily preserved. Furthermore, if we need to have some guarantees on the results, we need to specify what type of approximation should be used.

6. Computation Results

We used two contemporary computation kernels, d/dt [3] and the Level Set toolbox [17] for hybrid systems to implement these algorithms.

Fig. 1 shows the first example which uses the Vanderpol Equation [18] to demonstrate how $cPost_{cT}$ is computed with the Level Set toolbox. The constraint $D = X \setminus S$, where S is indicated as the black box in the figure, blocks the evolution of the reachable set. The reachable set is split into two parts but remains connected. The system dynamics are defined by the ODE $\dot{x}_1 = x_2$, and $\dot{x}_2 = x_2(1 - x_1^2) - x_1$.

Fig. 2 shows the second example which uses a linear system with a stable focus [18] to demonstrate the $\diamond$-algorithm by using d/dt. The equilibrium point is inside the set F. Each figure in the series shows the intermediate result of each R_k. The algorithm terminates after the reachable set stops growing within the analysis set. This example demonstrates how the domain of attraction can be estimated. The discovered stable focus in this case is a domain of attraction. The system dynamics are defined by the ODE $\dot{x}_1 = -x_1 - 1.9x_2$, and $\dot{x}_2 = 1.9x_1 - x_2$.

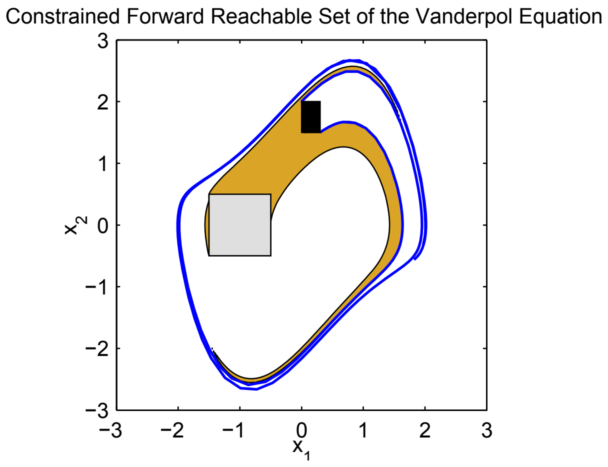

Figure 1. Constrained bounded-time Forward reachable set of the Vanderpol Equation, computed by the Level Set toolbox. The light gray box is the initial set F, the dark region is the forward reachable set, and the black box is the complement of the constraint set D. Computation time taken = 200 min.

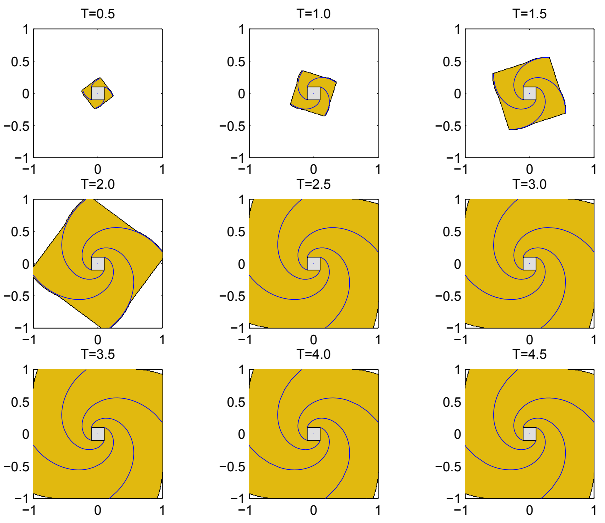

Figure 2. $\diamond$-algorithm applied to a linear system with a stable focus, computed by d/dt. For each figure in the series, the light gray box in the center is F, and the dark region is the set that can be reached from F within time T. The solid curves are the trajectories from the four corners of F. Computation time take = 1 min.

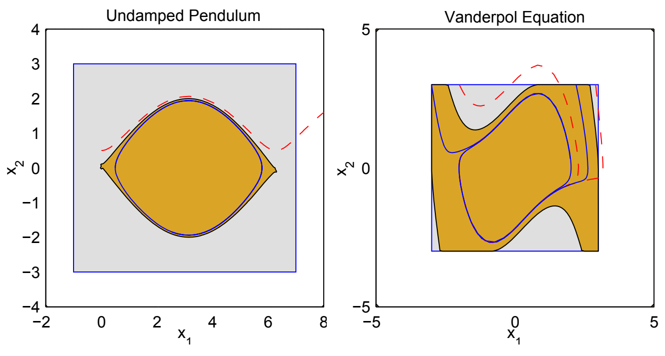

Figure 3. □-algorithm applied to an undamped pendulum (a), and the Vanderpol Equation (b), computed by the Level Set toolbox. The light gray box is F, and the dark region is true for $\Box F$. The solid trajectory of the equation is always staying in F because it is starting from the dark region that is $\Box F$. The dashed trajectory is not always inside because it does not originate in the safe set $\Box F$ (it is in the lighter region). Computation time =70 min for both a and b.

Fig. 3 shows the results of the □-algorithm applied to an undamped pendulum system [18] and the Vanderpol Equation. In each sub-figure, the sets F, F_s are indicated by the light gray region and the dark region, respectively. Trajectories starting from any point inside the dark regions will always stay inside, which shows that the safety property of both systems are satisfied. These examples show how the positively invariant sets can be approximated. The dynamics of the Vanderpol Equation are given in the first example and the dynamics of the undamped pendulum are defined by the ODE $\dot{x}_1 = x_2$, and $\dot{x}_2 = sin(x_1)$.

Previous examples have demonstrated how some useful system properties, such as $\Diamond F$, $\Box F$, can be computed. In addition, we can also use these algorithms to estimate the shape of the limit cycle. A limit cycle [18] is defined as the region of space where at least one other trajectory of the system spirals into it as time approaches infinity. If all trajectories of the behavior of system spiral into the limit cycle, then the limit cycle is called attractive and is also known as domain of attraction. One can similarly define a repulsive limit cycle.

Take the Vanderpol Equation as an example, it can be shown that there exist an attractive limit cycle and an equilibrium point inside the limit cycle. As shown in Fig. 4, by negating the vector field of the Vanderpol Equation, the

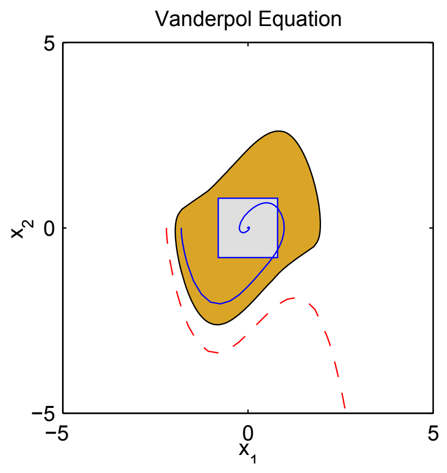

Figure 4. ◇-algorithm applied to the ODE, by negating the vector field of the Vanderpol Equation. The light gray box is F, and the dark region is true for $\Diamond F$, which gives an estimate of the shape of the limit cycle. Computation time taken=60 min.

limit cycle becomes repulsive instead of attractive. Then, by having the set F surrounding the equilibrium point we can use our ◇-algorithm to estimate the shape of the limit cycle, as the dark region shown in the figure. The dynamics in this example is defined by the ODE $\dot{x}_1 = -x_2$, and $\dot{x}_2 = -(x_2(1-x_1^2)-x_1)$.

For the examples in this section, the Level Set toolbox examples are computed on a Pentium IV 2.59GHz Windows machine with 512 MB memory, and the d/dt example is computed on a Pentium IV 2.59GHz Linux machine with 2 GB memory.

7. Conclusion

Embedded systems are modeled as hybrid systems that have both discrete and continuous dynamics. Designing safe and reliable embedded system require the analysis of various rich continuous dynamics observed for a given system. In this paper, we presented two basic synthesis algorithms that can be used to develop further more complicated algorithms to synthesize safe sets that satisfy a given temporal property. The initial sets produced by the algorithms are related to some classical concepts for continuous dynamical systems, such as invariant sets and domains of attraction. To guarantee termination of these algorithms one must use a state representation with finite representative elements. An example is using grids that divide the infinite state space into a finite number of elements.

8. Acknowledgment

The author acknowledges Prof. John Koo for his help in this work.

References

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[2] E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli. Effective synthesis of switching controllers for linear systems. *Proceedings of the IEEE*, 88(7):1011–1025, July 2000.

[3] E. Asarin, T. Dang, and O. Maler. The d/dt tool for verification of hybrid systems. In *Computer Aided Verification ,LNCS 2404*, pages 365–370. Springer-Verlag, July 2002.

[4] E. Asarin, T. Dang, O. Maler, and O. Bournez. Approximate reachability analysis of piecewise-linear dynamical systems. In N. A. Lynch and B. H. Krogh, editors, *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 20–31, Pittsburgh, PA, USA, April 2000. Springer Verlag.

[5] A. Bemporad and M. Morari. Verification of hybrid systems via mathematical programming. In F. W. Vaandrager and J. H. van Schuppen, editors, *Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 31–45, Berg en Dal, The Netherlands, March 1999. Springer Verlag.

[6] O. Botchkarev and S. Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In N. A. Lynch and B. H. Krogh, editors, *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 73–88, Pittsburgh, PA, USA, April 2000. Springer Verlag.

[7] A. Chutinan and B. H. Krogh. Verification of infinite-state dynamic systems using approximate quotient transition systems. *IEEE Transactions on Automatic Control*, 46(9):1401–1410, September 2001.

[8] J. E. R. Cury, B. H. Krogh, and T. Niinomi. Synthesis of supervisory controllers for hybrid systems based on approximating automata. *IEEE Transactions on Automatic Control*, 43(4):564–568, April 1998.

[9] T. Dang. *Verification Et Synthesis Des Systemes Hybrides*. PhD thesis, Institut National Polytechnique De Grenoble, October 2000.

[10] M. R. Greenstreet and I. Mitchell. Reachability analysis using polygonal projections. In F. W. Vaandrager and J. H. van Schuppen, editors, *Hybrid Systems: Computation and Control*, volume 1569 of *Lecture Notes in Computer Science*, pages 103–116, Berg en Dal, The Netherlands, March 1999. Springer Verlag.

[11] T. A. Henzinger and R. Majumdar. A classification of symbolic transition systems. In H. Reichel and S. Tison, editors, *Proceedings of the 17th International Conference on Theoretical Aspects of Computer Science (STACS 2000)*, Lectures Notes in Computer Science, pages 13–34. Springer Verlag, February 2000.

[12] A. B. Kurzhanski and P. Varaiya. Ellipsoidal techniques for reachability analysis. In N. A. Lynch and B. H. Krogh, editors, *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 202–214, Pittsburgh, PA, USA, April 2000. Springer Verlag.

[13] G. Lafferriere, G. J. Pappas, and S. Yovine. Symbolic reachability computation for families of linear vector fields. *Journal of Symbolic Computation*, 32(3):231–253, September 2001.

[14] E. A. Lee and H. Zheng. Operational semantics of hybrid systems. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 25–53, Zurich, Switzerland, March 2005. Springer Verlag.

[15] O. Maler. Control from computer science. *Annual Reviews in Control*, 26(1):175–297, 2002.

[16] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In E. W. Mayr and C. Puech, editors, *STACS*, volume 900 of *Lecture Notes in Computer Science*, pages 229–242. Springer Verlag, March 1995.

[17] I. Mitchell and C. Tomlin. Level set methods for computation in hybrid systems. In *HSCC '00: Proceedings of the Third International Workshop on Hybrid Systems: Computation and Control*, pages 310–323, London, UK, 2000. Springer-Verlag.

[18] S. Sastry. *Nonlinear Systems: Analysis, Stability and Control*. Springer Verlag, New York, USA, 1999.

[19] O. Stursberg and B. H. Krogh. Efficient representation and computation of reachable sets for hybrid systems. In O. Maler and A. Pnueli, editors, *Hybrid Systems: Computation and Control*, volume 2623 of *Lecture Notes in Computer Science*, pages 482–497, Prague, Czech Republic, April 2003. Springer Verlag.

[20] C. Tomlin, J. Lygeros, and S. Sastry. A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, 88(7):949–970, July 2000.

[21] H. Wong-Toi. The synthesis of controllers for linear hybrid automata. In *IEEE Conference of Decision and Control*, volume 5, pages 4607–4612, San Diego, CA, USA, December 1997.