Integrating Projections

Mark R. Greenstreet
Department of Computer Science

University of British Columbia
Vancouver, BC V6T 1Z4
Canada
mrg@cs.ubc.ca

Ian Mitchell
Scientific Computing and Computational Mathematics
Stanford University
Stanford, CA 94305-9025
USA
mitchell@sccm.stanford.edu

Abstract This paper describes three techniques for reachability analysis for systems modeled by ordinary differential equations (ODEs). First, linear models with regions modeled by convex polyhedra are considered, and an exact algorithm is presented. Next, non-convex polyhedra are considered, and techniques are presented for representing a polyhedron by its projection onto two-dimensional subspaces. This approach yields a compact representation, and allows efficient algorithms from computational geometry to be employed. Within this context, an approximation technique for reducing non-linear ODE models to linear nonhomogeneous models is presented. This reduction provides a sound basis for applying methods for linear systems analysis to non-linear systems.

1 Introduction

We are interested in verifying that circuits, as modeled by systems of non-linear ordinary differential equations (ODE's), correctly implement discrete specifications. Challenging verification problems arise when VLSI designers use methods such as precharged logic, single-phase clocking, and sense-amp based techniques that depend on the analog properties of the circuits to obtain better performance. In current practice, design validation relies heavily on simulation tools such as SPICE [Nag75]. However, even the best model is only approximate, and each simulation run can only consider a particular set of functions as inputs to the circuit and a particular set of values for model parameters. To obtain a reasonable level of confidence in a design, a large number simulations must be run. This process can be extremely time consuming; yet, in the end, simulation can not prove the correctness of a design.

Recently, we have been exploring an alternative approach to the problem of circuit-level design verification, based on ideas from dynamical systems theory. Correctness criteria for a circuit can be formulated in a logic which has meaning in both continuous and discrete domains. Rather than considering individual simulation runs, correctness criteria become topological properties in the continuous domain that must hold for an invariant set that contains *all* possible trajectories of the ODE model. To establish these invariants, we construct regions such that all trajectories on the boundaries flow inward [GM97]. For simple models these regions can be constructed manually, but for models arising in real circuits more automated methods are required. In [Gre96], we showed how these invariant sets can be constructed by reachability analysis using numerical integration.

An important advantage of our approach is that our analysis is based on ODE models similar to those that are used for industrial circuit simulation. Thus, our results are comparable with those obtained by traditional simulations—by speaking the same language as circuit designers, we encourage interaction with the eventual users of our techniques. Furthermore, considerable effort continues to be invested in developing accurate models for current fabrication processes. By using ODE models as the basis of our work, we can exploit these advances directly.

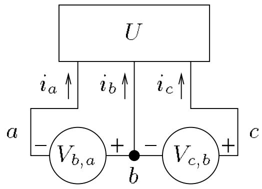
Two contributions are made by this paper. First, we describe an efficient way of representing non-convex high dimensional polyhedra using two-dimensional projections. This representation is by no means universal; however, it has shown promising results for a small number of circuits that we have analysed. Second, we show an integration based approach for computing reachability between regions represented using projections. Although we use floating point arithmetic in our implementation to obtain acceptable performance; in principle, the same techniques could be implemented with rational arithmetic and conservative rounding to create a strictly conservative implementation of the algorithm. The theoretic aspects of these contributions are contained in section 3. Preceding that section is a description of our models.

2 Models

In this section we show how to construct ODE models for our analysis. Section 2.1 describes the construction of models for MOS circuits. These circuits require inputs, and we typically wish to verify a circuit for all legal inputs. Readers familiar with circuit modeling may wish to skip directly to section 2.2, which describes Brockett's annulus construction and shows how it can be used to model inputs to our circuits.

2.1 Circuit Models

We model MOS circuits as a collection of voltage controlled current sources and (linear) capacitors. A voltage controlled current source defines a relationship between the voltages on its terminals and the currents flowing into those terminals. By convention, current is the flow of "positive charges," and a flow of electrons into the device is represented by a negative current. Consider the device depicted below:



The device U is connected to three nodes, a, b, and c. The voltage V_a denotes the voltage at node a, and likewise for V_b and V_c. We write V_{ba} to denote $V_b - V_a$ and likewise for V_{cb}. The current i_a denotes the current flowing into device U through node a. If U is a voltage controlled current source, then i_a is a function of the voltages V_a, V_b and V_c. We write $i_a = U_a(V_a, V_b, V_c)$.

More generally, let $\bar{V}$ denote the vector of node voltages in the circuit, and let $\bar{\imath}_U$ denote the vector of currents flowing into U through each node of the circuit. We write

$$\bar{\imath}_U = I_U(\bar{V}) \tag{1}$$

For example, an n-channel MOSFET can be modeled as a three terminal, voltage controlled current source. The three terminals are the gate, g, the source, s, and the drain, d. A simple model (see [GD85], equations 2.85 - 2.87) is

$$\begin{aligned}
i_g(V_g, V_s, V_d) &= 0 \\
i_d(V_g, V_s, V_d) &= 0, & &\text{if } V_{ds} \geq 0 \ \&\ V_{gs} < V_t \\
&= G(V_{gs} - V_t)^2, & &\text{if } V_{ds} \geq 0 \ \&\ V_{ds} > V_{gs} - V_t \geq 0 \\
&= GV_{ds}(2(V_{gs} - V_t) - V_{ds}), & &\text{if } V_{ds} \geq 0 \text{ and } V_{gs} - V_t \geq V_{ds} \geq 0 \\
&= -i_d(V_g, V_d, V_s), & &\text{if } V_{ds} < 0 \\
i_s(V_g, V_s, V_d) &= -i_d
\end{aligned} \tag{2}$$

where V_t is the "threshold voltage" of the transistor, and G is the transconductance. These two constants are determined by the size and shape of the transistor and by properties of the fabrication process.

A capacitor defines a relationship between the time derivatives of the voltages on the terminals and the currents flowing into these terminals. For a capacitor of fixed capacitance C connected to nodes a and b,

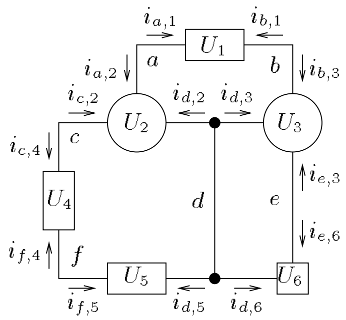
$$i_b = -i_a = C\frac{dV_b}{dt} - C\frac{dV_a}{dt}$$

More generally, a capacitor U defines a matrix valued function C_U such that

$$\bar{\imath}_U = C_U(\bar{V})\frac{d\bar{V}}{dt} \tag{3}$$

For the models arising from MOS circuits, this matrix corresponds to a network of voltage dependent, two-terminal capacitors. Physically, there must be some capacitance between every pair of nodes; in practice, many of these capacitances are small and neglected when constructing a circuit model. Any realistic model will associate at least one capacitor with each node; for such models, $C_U(\bar{V})$ is real-symmetric and positive definite.

Given models for each device in the circuit, we construct an ODE model for the whole system using Kirchoff's current law. As depicted in figure 1, Kirchoff's current law states that the sum of the currents flowing into each node of the circuit must be zero. Likewise, the sum of the currents flowing into each device must be zero. Both of these constraints are direct consequences of charge conservation.



Kirchoff's Current Law:

$$\forall x \in \{a \dots f\}. \sum_{m=1}^{6} i_{x,m} = 0$$

$$\forall m \in \{1 \dots 6\}. \sum_{x=a}^{f} i_{x,m} = 0$$

Fig. 1. Kirchoff's Laws

From Kirchoff's current law, we have

$$\textstyle\sum_{U \in C} C_U(\bar{V}) \frac{d\bar{V}}{dt} + \sum_{U \in I} I_U(\bar{V}) = 0$$

where C denotes the set of capacitor devices, and I denotes the set of current source devices. Solving for $d\bar{V}/dt$ yields

$$\tfrac{d\bar{V}}{dt} = -\left(\textstyle\sum_{U \in C} C_U(\bar{V})\right)^{-1} \left(\textstyle\sum_{U \in I} I_U(\bar{V})\right) \tag{4}$$

which is an ODE model for the circuit.

The device models above are simplistic, allowing a shorter presentation and making the analysis in the remainder of this paper tractable. While these models capture many of the key features of MOS circuit operation, we note that the transistor model of equation 2 neglects the body effect and short channel effects. Similarly, when modeling capacitors we make the simplifying assumption that C_U is a constant; in real MOS designs, C_U depends substantially on $\bar{V}$. Kirchoff's current law is itself an approximation of Maxwell's equations, and so ignores "displacement currents." Typically, designers use more accurate models than those presented for transistors and capacitors—here we have chosen to avoid complexity while retaining the key features of realistic circuit models.

2.2 Input Signals

The problem of verifying an entire chip at the ODE level appears to be hopelessly intractable. Instead, we focus on the problem of verifying small circuits and showing that the outputs of one circuit satisfy the constraints that we assume for inputs to other circuits. Such a method requires a mechanism for specifying the expected inputs and the allowed outputs of each small circuit.

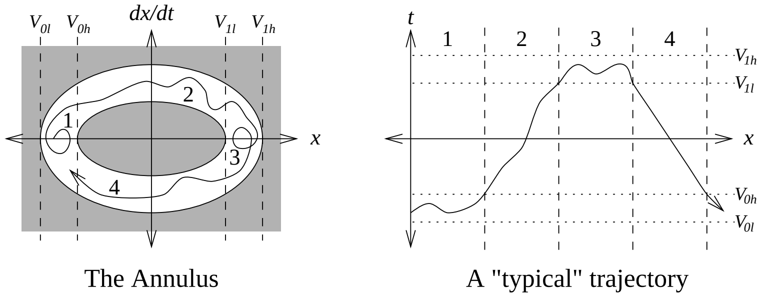


Fig. 2. Brockett's Annulus

Figure 2 depicts the annulus proposed by Brockett [Bro89] that we use to specify the levels and transitions of signals. When a variable is in region 1, its value is constrained but its derivative may be either positive or negative. We will consider this a logically low signal. When the variable leaves region 1, it must enter region 2. Because the derivative of the variable is strictly positive in this region, it makes a monotonic transition rising to region 3. Regions 3 and 4 are analogous to regions 1 and 2, and correspond to logically high and monotonically falling signals respectively. Because transitions through regions 2 and 4 are monotonic, traversals of these regions are distinct events. The properties of the annulus provide a topological basis for discrete behaviours.

Many common signal parameters are represented by the geometry of an annulus. The horizontal radii of the annulus define the maximum and minimum high and low levels of the signal (i.e. V_{0l}, V_{0h}, V_{1l}, and V_{1h} in figure 2). The maximum and minimum rise time for the signal correspond to trajectories along the upper-inner and upper-outer boundaries of the annulus respectively. Likewise, the lower-inner and lower-outer boundaries of the annulus specify the maximum and minimum fall times.

3 Reachability Analysis

In this section we present our theoretic results. After looking at the connection between verification and reachability, we examine three increasingly difficult reachability analyses: linear models with convex polyhedra, linear models with non-convex polyhedra, and finally nonlinear models with non-convex polyhedra.

3.1 Verification as Reachability

Many circuit verification problems can be formulated as reachability analysis problems. For example, consider a circuit that implements a simple state machine. An ODE model provides a mapping between the continuous circuit state (node voltages) and the time derivative of that state. Thus, given a point in the continuous space, the value and derivative of each signal is known. Using a Brockett annulus, each signal can be interpreted discretely as being low, rising, high, or falling. The continuous model implements the discrete specification if every reachable point in the continuous model corresponds to a state or transition of the discrete specification.

First consider the verification of bounded prefixes of trajectories. For a circuit with d nodes, the continuous state space is $\mathbb{R}^d$. We assume that the derivative function for the model is autonomous (i.e. independent of time) and finitely piecewise continuous (therefore locally bounded). Given a bounded region $Q \subset \mathbb{R}^d$, $Q_0 \subseteq Q$, and $t_f \in \mathbb{R}^+$, we want to show that all trajectories that start in Q_0 at time 0 will remain in Q for all times up to t_f. Our approach to this problem is to construct a sequence of time steps $t_0 < t_1 < \ldots < t_k$ such that $t_0 = 0$ and $t_k = t_f$. For $i = 1 \ldots k$, we construct a region Q_i such that any trajectory that starts in Q_{i-1} at time t_{i-1} will be in Q_i at time t_i. We then construct a second set of regions $Q'_0, \ldots Q'_{k-1}$ such that any trajectory that starts in Q_i at time t_i will remain in Q'_i up to and including time t_{i+1}. If $\cup_{i=0}^{k-1} Q'_i \subseteq Q$, then all trajectories that start in Q_0 at time 0 will remain in Q for all times up to t_f as can be readily shown by the construction of Q'_i.

Now consider infinite trajectories. Let Q, Q_i, Q'_i, and t_i be constructed as above, $D = \cup_{i=0}^{k-1} \ldots Q'_i$, and $Q^+ = \cup_{i=0}^{k-1} \ldots Q_i$. If $Q_k \subseteq Q^+$, then any trajectory that starts in Q_0 remains in D forever. To see this, let $x : \mathbb{R}^+ \to \mathbb{R}^d$ be a trajectory with $x(0) \in Q_0$. Let $\tau_{\min} = \min_{i=1}^{k} t_k - t_{k-1}$. There exists a sequence of times, τ_m, such that for all $m \geq 0$, $x(\tau_m) \in Q^+$ and $\tau_m \geq m\tau_{\min}$. The proof is completed by induction on m. For $m = 0$, $\tau_m = 0$. For $m > 0$, let $j \in \{0 \ldots k-1\}$ such that $x(\tau_{m-1}) \in Q_j$. Let

$$\tau_m = \tau_{m-1} + (t_{j+1} - t_j) \geq \tau_{m-1} + \tau_{\min} \geq m * \tau_{\min}$$

Then, $x(\tau_m) \in Q_{j+1} \subseteq Q^+$.

In general, it is not feasible to represent exactly the reachable regions of systems modeled by ODEs. Most non-linear ODEs, including those that arise when modeling VLSI circuits, do not have closed form solutions. Because proof of safety properties is our objective, over estimation of the reachable space is conservative—false negatives are possible, but not false positives. Consequently, we use "containing approximations", within which lie the true reachable state spaces.

As described above, the next few sections examine three different cases of reachability analysis. First, we consider the special case of linear ODE's where the initial region is a convex polyhedron—we show that the Q_i sequence can be computed exactly, and the Q'_i sequence can be computed with arbitrary accuracy. In general, convexity is not preserved by non-linear models, and we develop our treatment of non-linear models in two steps. First, section 3.3 presents a conservative approximation technique for the particular class of non-convex polyhedra that can be represented by their projections onto two-dimensional subspaces; however, linear models are retained. In section 3.4 we show how these projection polyhedra can be used with non-linear models.

3.2 Linear models and convex polyhedra

This section presents the special case where the ODE model is linear, and Q_0 is convex. An ODE model is linear if it can be written in the form

$$\dot{x} = Ax \tag{5}$$

where $x : \mathbb{R}^+ \to \mathbb{R}^d$ is a trajectory and $A \in \mathbb{R}^{d \times d}$ is a matrix (note that this definition of "linear" is more general than the one used in much of the hybrid systems literature). We assume that A has a full-rank set of eigenvectors. If not, a small perturbation of A will produce such a matrix, and the techniques presented in section 3.4 can be applied. With this assumption, the solution of equation 5 is [HS74]

$$x(t) = e^{tA}x(0) \tag{6}$$

For any fixed value of t, e^{tA} is a linear operator that can be represented by a matrix, and e^{tA} is invertible.

A d-dimensional convex polyhedron with m faces can be represented by linear program of the form

$$Mx \leq B \tag{7}$$

where $M \in \mathbb{R}^{m \times d}$ is a matrix and $B \in \mathbb{R}^m$ is a vector (see [PS82]). We write (M, B) to denote the linear program of equation 7, and write $x \in (M, B)$ to denote that x satisfies this linear program.

Polyhedra can be bloated. If (M, B) is a linear program, and u is a real number, then, $bloat((M, B), u)$ is the polyhedron obtained by moving each face of M outward by u. Let $\Delta \in \mathbb{R}^d$ be a vector such that that its j^{th} element is given by $\Delta_i(j) = u||M_j||_2$, where $||M_j||_2$ denotes the L2 norm of row j of M. Then,

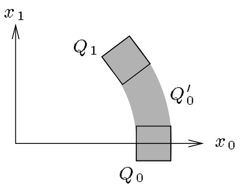
$$bloat((M, B), u) = (M, B - \Delta) \tag{8}$$

Convexity is preserved by linear operators. In particular, let the linear program (M_0, B_0) describe the convex region Q_0, let $t_1 \in \mathbb{R}^+$, and let A be the matrix representation of a linear ODE model. A point x is reachable from Q_0 at time t_1 if and only if $x \in (M_0 e^{-tA}, B)$, which follows directly from equations 6 and 7. Thus, we can construct $Q_1 \ldots Q_k$ such that for $i = 1 \ldots k$, any trajectory that starts in Q_{i-1} at time t_{i-1} will be in Q_i at time t_i. In particular,

$$Q_i = (M_0 e^{-t_i A}, B) \tag{9}$$

These Q_i are exact.

$$\begin{aligned} \dot{x}_0 &= -x_1 \\ \dot{x}_1 &= x_0 \end{aligned}$$



$t_1 = 0.64$

Fig. 3. A simple linear system

Although the Q_i's (the reachable regions at each time step) are convex, the same does not necessarily hold for the Q'_i's (the regions reachable during all times between steps). For example, consider the system depicted in figure 3. Trajectories are counter-clockwise circles centered at the origin. Although Q_0 and Q_1 are both convex, the minimal region for Q'_0 is the region swept out by moving Q_0 through an arc of t_1 radians (the shaded region in figure 3). Region Q'_0 is not convex.

Rather than trying to solve for Q'_0 exactly, we will find an approximation. Note that $\dot{x} = Ax$ is locally bounded; therefore it is bounded in Q. Define the scalar $||\dot{x}||_{\max} = \max_{x \in Q} ||Ax||_2$. A trajectory that starts in region Q_i at time t_i remains within a distance $(t_{i+1} - t_i)||\dot{x}||_{\max}$ of Q_i until time t_{i+1}. Let

$$Q'_i = \mathit{bloat}((M_0 e^{-t_i A}, B), ||\dot{x}||_{\max}) \tag{10}$$

For any trajectory x such that $x(t_i) \in Q_i$ and for any time $t \in [t_i, t_i + 1]$, $x(t) \in Q'_i$ as required.

Although the Q'_i are containing approximations, each one is computed from an exact Q_i—the errors of making a conservative approximation do not accumulate between time steps. To achieve accurate estimates of the reachable space, the time steps should be relatively small so that there is little of Q'_i outside of $Q_i \cup Q_{i+1}$. For example, this approach would compute a large overestimate of Q'_0 for the time step depicted in figure 3.

A straightforward approach to verification is to construct a sequence of Q_i and Q'_i as described above, and verify that each Q'_i is contained in Q. If all containments are established, then the verification is complete. Otherwise, choose i such that Q'_i is not contained in Q. A counterexample to the verification is established if either of the exact solutions Q_i or Q_{i+1} is not contained in Q. If neither of the exact solutions provide a counterexample, divide the step from Q_i to Q_{i+1} into two smaller steps and repeat the verification. This process terminates when containment of all the Q_i''s is verified, a counter-example is found, or the time step is smaller than is meaningful for the chosen model. In the latter case, the property cannot be verified with the given model. Typical variation in MOS circuit parameters can $\pm 20\%$ or more, although closely matched circuits (e.g. sense-amplifiers, see [Bak90]) can be designed that are balanced to within a few parts per thousand.

3.3 Linear models and non-convex polyhedra

Although systems with linear ODE models can be analysed quite accurately using the techniques described in the previous section, such systems do not have a rich enough phase space structure for interesting digital computation. In a linear system, the asymptotic behaviour of trajectories is either convergence towards the origin, divergence to infinity, or an orbit centered at the origin. In order to examine more interesting systems, we need techniques to analyse non-linear models. In general, these models do not preserve the convexity of polyhedra; therefore, we begin by describing the class of non-convex polyhedra that we use in our analysis.

Representation
We represent high dimensional polyhedra by their projections onto two dimensional subspaces, where these projections are not required to be convex. Conversely, a full dimensional polyhedron can be obtained from its projections by back-projecting each into a prism in $\mathbb{R}^d$ and computing the intersection of those prisms (see figure 4). More formally, let $\{u_1, u_2, \ldots u_d\}$ be an orthogonal basis for $\mathbb{R}^d$. If P is a polygon, we write $(u_{X(P)}, u_{Y(P)})$ to denote the basis of P. We write $\mathit{ConvexHull}(P)$ to denote the convex hull (see [PS85]) of P, and it is understood that $X(\mathit{ConvexHull}(P)) = X(P)$ and $Y(\mathit{ConvexHull}(P)) = Y(P)$. We write $\mathit{prism}(P)$ to denote the inverse projection of P back into the full dimensional space:

$$\mathit{prism}(P) = \{(x_1, \ldots x_d) \in \mathbb{R}^d | (x_{X(P)}, x_{Y(P)}) \in P\} \tag{11}$$

Let $\mathcal{P}$ be a collection of polygons. The object represented by $\mathcal{P}$ is $Q(\mathcal{P})$ where

$$Q(\mathcal{P}) = \bigcap_{P = \mathcal{P}} \mathit{prism}(p) \tag{12}$$

We note that faces of $Q(\mathcal{P})$ correspond to edges of the projection polygons. If P is a projection polygon, and e is and edge of P, we write $X(e)$ and $Y(e)$ to denote $X(P)$ and $Y(P)$ respectively. Likewise, we define $\mathit{prism}(e)$ to be

$$\mathit{prism}(e) = \{(x_1, \ldots x_d) \in \mathbb{R}^d | (x_{X(e)}, x_{Y(e)}) \in e\} \tag{13}$$

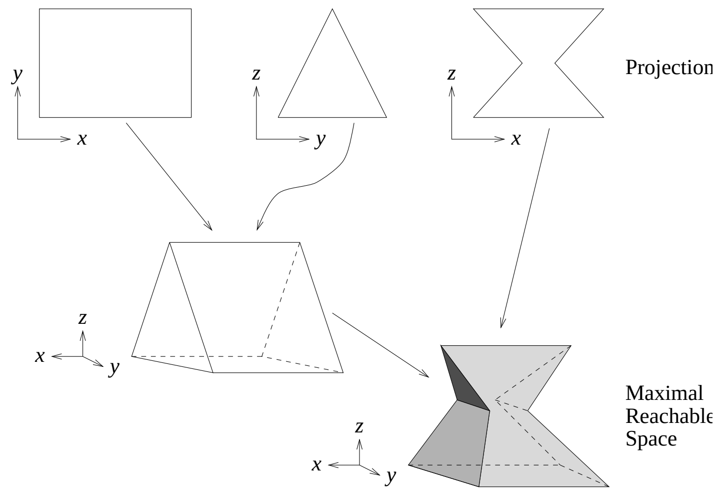


Fig. 4. A three dimensional polyhedron and its projections

If e is an edge of a projection polygon, we write $face(e, \mathcal{P})$ to denote the corresponding edge of e:

$$face(e, \mathcal{P}) = Q(\mathcal{P}) \cap prism(e) \tag{14}$$

We write $face(e)$ when $\mathcal{P}$ is apparent from context.

There are several advantages to this representation. First, it corresponds to an engineer's intuitive notion of how a circuit works. Typically, each signal is "controlled" by a small number of other signals. Pairing each node with each of its controlling nodes naturally captures the causal behaviour of the circuit. Because most circuits have limited fan-in and fan-out, the number of such pairs, and hence the number of polygons, is proportional to the number of nodes in the circuit.

From the perspective of a numerical analyst, the engineer's intuition means that a full dimension polyhedral representation of the reachable region may provide unneeded freedom in its ability to represent constraints between every possible combination of variables. In the same way that many matrices encountered in practice contain interaction between only limited sets of variables, in many ODE systems each variable only directly influences a small number of others. Dense storage and manipulation of sparse matrices is wasteful; similarly, representing the reachable state space as a full dimensional polyhedron may be exponentially extravagant.

Finally, there are algorithmic advantages to using projections. The existence of a sound method for computing the evolution of bounding polyhedra represented in this manner is key to verification. In addition, all geometric operations take place in two dimensions where there are many results and algorithms available from computational geometry [PS85]. Lastly, it is relatively easy to compute the convex hull of a polygon, thus producing a containing approximation of that polygon in the form of a linear program.

Of course, there are many polyhedra that cannot be exactly represented by this approach. First, indentations on the surface of an object can not be represented; likewise, many perforated objects and knot-like objects can only be approximated. We require that the projections are orthogonal; therefore, edges formed by the intersection of projections must be at right-angles. Further experimentation is needed to determine the significance of these limitations when analysing circuits modeled by ODEs.

Reachability

Let $Q(\mathcal{P}_0)$ be a polyhedron, and let $\dot{x} = Ax$ be a linear model for a system. Given a monotonically increasing sequence of times, $t_1 \ldots t_k$, we will construct a sequence of polyhedra $Q(\mathcal{P}_1) \ldots Q(\mathcal{P}_k)$ such that trajectories that start in $Q(\mathcal{P}_0)$ at time $t = 0$ are contained in $Q(\mathcal{P}_i)$ at time $t = t_i$. Our approach is based on three observations, which we justify below. First, it is sufficient to consider trajectories emanating from the faces of $Q(\mathcal{P}_0)$, as these will define the faces of the polyhedron at later times. Second, for each edge e of a projection polygon, it is straightforward to construct a convex containing approximation for $\mathit{face}(e)$. Third, the method described in section 3.2 can be used to determine reachability from this convex approximation.

Because Ax is locally bounded, trajectories are continuous and cannot cross. Therefore, trajectories starting on a face of the polyhedron provide bounds for trajectories starting in the interior.

To construct a convex approximation for $\mathit{face}(e)$ let

$$Z(\mathcal{P}) = \bigcap_{P \in \mathcal{P}} \mathit{prism}(\mathit{ConvexHull}(P)) \tag{15}$$

It is straightforward to show that $\mathit{ConvexHull}(Q(\mathcal{P})) \subseteq Z(\mathcal{P})$, and $\mathit{ConvexHull}(\mathit{face}(e, \mathcal{P})) \subseteq \mathit{ConvexHull}(Q(\mathcal{P})) \cap \mathit{prism}(e)$. Therefore,

$$\mathit{ConvexHull}(\mathit{face}(e, \mathcal{P})) \subseteq Z(\mathcal{P}) \cap \mathit{prism}(e) \tag{16}$$

Given $\mathcal{P}$, a linear program for $Z(\mathcal{P})$ can be constructed by computing the convex hull for each polygon in $\mathcal{P}$ and taking the conjunction of their constraints. Each polygon is two dimensional, allowing efficient (i.e. $O(n \log n)$) algorithms to be used. Once $Z(\mathcal{P})$ is calculated, it is easily extended to produce $Z(\mathcal{P}) \cap \mathit{prism}(e)$ for each edge. This provides our convex approximation of $\mathit{face}(e, \mathcal{P})$.

The method described above allows us to construct a $d - 1$ dimensional convex approximation for each face of $Z(\mathcal{P}_0)$. The reachable space from each face can then be computed by the techniques given in section 3.2. The boundary of the region reachable from $Z(\mathcal{P}_0)$ is contained in the union of the regions reachable from the faces.

In order for the same algorithms to be used for the next time step, we would like to compute a containing approximation of this boundary as a series of projections—describing the new boundary in the same way that $Z(\mathcal{P}_0)$ was described. Given a linear program for a face, the projection of that face onto a plane can be computed by finding an extremal vertex of the projection, and tracing the rest of the vertices with a series of pivots (see [AF92]). Because there may be an exponentially large number of vertices in this projection, such an approach may be slow. To avoid tracing too many vertices, extremal vertices can be computed for a fixed set of directions, and edges associated with these vertices joined to produce a containing approximation of the projection. Regardless of the method chosen to compute the projections, an object that contains everything reachable from $Q(\mathcal{P}_0)$ can be constructed by filling in the projection polygons (another straightforward operation).

An unattractive feature of this approach is that the reachable polyhedron for each face must be projected onto *all* planes used for the original projection polygons. Intuitively, this is because with a linear model, we can calculate the exact image of the convex approximations of the face for arbitrarily large times. During such an extended time interval, the polyhedron can rotate, and any face can become an extremal face for any projection.

3.4 Non-linear models and non-convex polyhedra

We extend the methods of the previous section to non-linear models in three steps. First, we will approximate the non-linear model by a linear model and a correction term. Second, we show how this correction term can be described as an non-determinate function of time, allowing the non-linear ODE to be approximated by a first order linear differential equation with an non-determinate nonhomogeneity. Finally, by bounding the solutions of the nonhomogenous system, we obtain a containing approximation of solutions to the original non-linear system.

Because the method from section 3.3 considers each face separately, we focus on the problem of finding the points reachable in time Δt from a point in $face(e)$ for some edge e, for a model whose derivative function has an L2 norm bounded by $||\dot{x}||_{\max}$. In determining the region reachable from $face(e)$, only points in $bloat(face(e), (\Delta t)||\dot{x}||_{\max})$ need to be considered. The derivation of the linear approximation and correction term is handled by the model—in other words, we leave it to the ingenuity of the programmer. When the model is evaluated, $bloat(face(e), (\Delta t)||\dot{x}||_{\max})$ is available as a linear program, so linear bounds can be readily obtained describing the region in which the approximation and correction must be valid.

As an example, consider the transistor model presented in equation 2 with $V_t = 0.5$. For a particular bloated face, assume $1.2 \leq V_{gs} \leq 1.6$ and $2.4 \leq V_{ts} \leq 3.1$. Then, everywhere in this region $i_{ds} = G(V_{gs} - V_t)^2$. Linearizing about the mid-point of the region and choosing an additive constant to minimize the worst-case absolute value of the error, we get $i_{ds} =$

$G(1.8V_{gs} - 1.69) \pm \epsilon(v_{gs}, v_{ds})$, where $\epsilon(v_{gs}, v_{ds}) \in [-0.02, 0.02]$. Similar techniques apply when the feasible region includes more or other modes of the transistor's operation.

Linear models can also be computed for input signals that are described using annuli (recall figure 2). As for the transistor model, the input signal model queries the linear program for the bloated face to determine upper and lower bounds for the value of the signal. For any given value of the signal, the annulus specifies upper and lower bounds for its time derivatives. From this description, a linear model with an error term can be computed. For such signals, the error term can be quite large; especially when the signal can be in the first (logical low) or third (logical high) regions of the annulus.

The non-linear correction term is a function of the state of a trajectory:

$$\dot{x} = Ax + \epsilon(x) \tag{17}$$

The model provides bounds on $\epsilon(x)$; thus, we write $\epsilon(x) \in E$ for some $E \subseteq \mathbb{R}^d$. For any particular trajectory, the correction term can be understood as a function of time, and we write

$$\dot{x} = Ax + \xi(t) \tag{18}$$

By computing the set of points reachable by trajectories for all functions ξ with $\xi(t) \in E$, we obtain a containing approximation for the original, non-linear system.

Equation 18 is a linear, nonhomogeneous, first-order differential equation. Such equations have a closed form solution [Apo67], namely:

$$x(t) = e^{tA}x(0) + e^{tA}\int_0^{\Delta t} e^{-uA}\xi(u)du \tag{19}$$

The $e^{tA}x(0)$ term is the solution to the linear approximation and the $e^{tA}\int_0^{\Delta t} e^{-uA}\xi(u)du$ term is the perturbation arising due to the non-linear correction in the model. A bound on the contribution of this correction term is computed next.

We assume that A has a full rank set of eigenvectors. If not, A can be perturbed slightly so as to satisfy this condition, and the perturbation can be reflected by slightly enlarging the correction term. Now, A can be diagonalized [HS74]; thus $e^{-tA} = D^{-1}e^{-tA^{\dagger}}D$, where D is the diagonalizing matrix, and $A^{\dagger}$ is diagonal. The elements of $e^{-tA^{\dagger}}$ (also a diagonal matrix) can be readily bounded for all $t \in [0, \Delta t]$. Using standard optimization techniques [PS82], a linear program can be constructed that is a containing approximation for the values of $e^{tA}\int_0^{\Delta t} e^{-uA}\xi(u)du$.

The previous paragraph provides a mathematically rigorous way to bound the contribution to trajectories of the non-linear component of the model. We expect that it would be impractical to implement this method due to its reliance on diagonalizing A—a procedure that is both time-consuming and numerically sensitive. Instead, we plan to sample e^{-uA} for several values of $u \in [0, \Delta t]$ using a numerical approximation such as an integration algorithm. From these samples, approximate bounds on the non-linear contribution can be found. Just as with the mathematically rigorous approach, these bounds can be expressed as a linear program.

Using one of the methods in the previous two paragraphs, a containing approximation in linear program form for $e^{tA} \int_{u=0}^{\Delta t} e^{-uA}\xi(u)$ can be constructed. Section 3.3 built a linear program containing the values of $e^{tA}\mathit{face}(e)$. For reasons that will be explained shortly, we will instead use a linear program that contains the values of $e^{tA}\mathit{face}'(e)$, where

$$\begin{aligned}
&\mathit{face}'(e, (\Delta t)\|\dot{x}\|_{\max}) \\
&\qquad = \mathit{bloat}(Z(\mathcal{P}), \|\dot{x}\|_{\max}) \cap \mathit{prism}(\mathit{extend}(e, (\Delta t)\|\dot{x}\|_{\max})) \\
&\mathit{extend}(e, (\Delta t)\|\dot{x}\|_{\max}) \\
&\qquad = e \text{ with end points extended outward by } (\Delta t)\|\dot{x}\|_{\max}
\end{aligned}$$

Note that $\mathit{face}(e) \subseteq \mathit{face}'(e)$. A containing approximation for the sum of $e^{tA} \int_{u=0}^{\Delta t} e^{-uA}\xi(u)$ and $e^{tA}\mathit{face}'(e)$ can also be described by a linear program, and we can approximate the boundary of the reachable space at time Δt as the union of these linear programs for each face.

The methods described in this section rely on representation of the reachable space by a collection of two dimensional projections. For example, we use an approximation of the convex hull of the reachable space which is derived from the convex hulls of the projections. Furthermore, we need to know the endpoints of each edge when creating the convex approximation of the corresponding face. Finding the endpoints is straightforward when they are defined by segment intersections in a plane. Therefore, each integration step must end by computing projection polygons for the new reachable space object.

The technique described in section 3.3—projecting the convex hull for each transformed face onto each projection plane—could be applied here as well. For the methods described in this section, it is only necessary to project each transformed face back to the projection plane for its original edge. Let e be an edge of polygon P and e' be an adjacent edge of another polygon. Then e and e' are orthogonal. Also note that all points of $\mathit{face}(e')$ lie on the inside of $\mathit{face}'(e, (\Delta t)\|\dot{x}\|_{\max})$. Therefore, all trajectories starting from $\mathit{face}(e')$ remain on the inside of $\mathit{face}'(e)$ at the end of the time step. Thus, the projection of the boundary of the polyhedron into the plane of P is completely determined by the projection of the faces arising from edges in P at the beginning of the time step.

4 Conclusion

Many verification problems can be formulated as questions of reachability. With a circuit modeled by a system of ordinary differential equations, the reachability problem can be formulated as: "given an initial region Q_0 and an ending time t_f (possibly $+\infty$), find a region Q such that all trajectories starting in Q_0 at time $t = 0$ remain in Q at least until time $t = t_f$."

We have addressed this problem for three classes of models and regions. First considering linear models with convex regions, we showed how the region reachable at a future time can be computed exactly. Furthermore, a containing approximation for points reachable through all times up until that future time can be computed with a simple trade-off between effort and accuracy. We note that the HyTech tool [HH95] represents reachable regions as a union of convex polyhedra, and it is possible that the techniques presented there could be applied in this first context.

Because models with non-linearities do not preserve the convexity of regions, it was next necessary to identify an efficient representation for non-convex polyhedra. For our purposes, projection polyhedra—where an object is represented by its projection onto two dimensional subspaces—provide such a representation, allowing us to apply efficient algorithms from computational geometry in two-dimensions to our higher dimensional problems.

Finally, we addressed the analysis of non-linear systems, by approximating the non-linear model by a linear term and a non-linear correction. The correction can be kept small by computing separate such models for each face of the reachable space, and can be approximated by a non-determinant "error" function of bounded magnitude. This construction allowed us to convert a non-linear model into a linear nonhomogenous differential equation, which can be solved analytically, and such solutions allow us to bound the reachable space.

The analysis presented in this paper shows that ideas from computational geometry, dynamical systems, formal methods, linear algebra, and numerical computation can all contribute to the verification of systems with ODE models. The authors are currently implementing a tool to demonstrate these techniques.

Acknowledgements

We appreciate an extended e-mail discussion with Oded Maler and Thao Dang on reachability with continuous models. Jack Snoeyink and Danny Chen have guided us about what is and what is not feasible in computational geometry.

References

[AF92] D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete Computational Geometry*, 8:295–313, 1992.

[Apo67] Thomas M. Apostle. *Calculus*, volume 1. John Wiley and Sons, Inc., New York, second edition, 1967.

[Bak90] H.B. Bakoglu. *Circuits, Interconnections, and Packaging for VLSI.* Addison-Wesley, 1990.

[Bro89] R. W. Brockett. Smooth dynamical systems which realize arithmetical and logical operations. In Hendrik Nijmeijer and Johannes M. Schumacher, editors, *Three Decades of Mathematical Systems Theory: A Collection of Surveys at the Occasion of the 50th Birthday of J. C. Willems*, volume 135 of *Lecture Notes in Control and Information Sciences*, pages 19–30. Springer, 1989.

[GD85] Lance A. Glasser and Daniel W. Dobberpuhl. *The Design and Analysis of VLSI Circuits.* Addison-Wesley, 1985.

[GM97] Mark R. Greenstreet and Ian Mitchell. Reachability with discrete and ODE models. In Michael Lemmon, editor, *Fifth International Hybrid System Workshop*, Notre Dame, September 1997.

[Gre96] Mark R. Greenstreet. Verifying safety properties of differential equations. In *Proceedings of the 1996 Conference on Computer Aided Verification*, pages 277–287, New Brunswick, NJ, July 1996.

[HH95] T.A. Henzinger and P.-H. Ho. HyTech: The Cornell Hybrid Technology Tool. In P. Antsaklis, A. Nerode, W. Kohn, and S. Sastry, editors, *Hybrid Systems II*, Lecture Notes in Computer Science 999, pages 265–293. Springer-Verlag, 1995.

[HS74] Morris W. Hirsch and Stephen Smale. *Differential Equations, Dynamical Systems, and Linear Algebra.* Academic Press, San Diego, CA, 1974.

[Nag75] L.W. Nagel. SPICE2: a computer program to simulate semiconductor circuits. Technical Report ERL-M520, Electronics Research Laboratory, University of California, Berkeley, CA, May 1975.

[PS82] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice Hall, Englewood Cliffs, NJ, 1982.

[PS85] Franco P. Preparata and Michael I. Shamos. *Computational Geometry: An Introduction.* Texts and Monographs in Computer Science. Springer, 1985.