Computing Reachable States for Nonlinear Biological Models

Thao Dang, Colas Le Guernic, and Oded Maler*

CNRS-VERIMAG, 2, av. de Vignate, 38610 Gieres, France

Abstract. In this paper we describe reachability computation for continuous and hybrid systems and its potential contribution to the process of building and debugging biological models. We then develop a novel algorithm for computing reachable states for *nonlinear* systems and report experimental results obtained using a prototype implementation. We believe these results constitute a promising contribution to the analysis of complex models of biological systems.

1 Introduction

The development of modeling formalisms and analysis techniques for the study of biological systems is a central topic in systems biology. The formalisms proposed for representing biological processes are very diverse, differing at the levels of abstraction, time scales and types of dynamics. The formalism chosen depends naturally on the level of detail needed to answer the specific biological question and on the granularity of available experiments. The contribution of this work is at the level of abstraction of ordinary differential equations (ODEs), a widely used modeling formalism. Biological systems, for instance metabolic networks consisting of sets of reactions, can be viewed as continuous dynamical systems with state variables denoting concentrations. The resulting differential equations are derived, for example, from mass action rules and are, typically, polynomial. Such equations can be numerically simulated from a given initial condition provided that the *exact* values of the parameters and the external environmental conditions are known. In certain restricted cases it is possible to determine global properties analytically.

Though widely used, ODEs suffer from several limitations. First, the passage from a finite number of molecules to real-valued concentration is not always justified, especially when the number of molecules is small [22]. Secondly, many biological phenomena, for example gene activation, are more naturally modeled as transitions between discrete states. Pure ODEs cannot easily accommodate this mixture of continuous evolutions and discrete events. Alternatively, purely discrete formalisms, based on transition systems expressed in various syntactic forms, suffer from a similar reciprocal limitation in the sense of not being amenable to quantitative reasoning.

* Part of this work was done while the third author was a Weston visiting professor at Weizmann Institute.

P. Degano and R. Gorrieri (Eds.): CMSB 2009, LNBI 5688, pp. 126–141, 2009.

Second, the lack of quantitative information concerning molecular concentrations, reaction rates and other parameters is the rule, not the exception, in Biology. Consequently the value of predictions obtained using numerical ODEs models, where the values of the parameters are "guessed" or "tuned", is severely limited. Moreover, the validation of models based on ODEs with poorly-known parameters is difficult if not impossible because we are never sure to have covered all the qualitative behaviors compatible with a model by performing only a finite number of simulations, each with a different choice of parameters. This fact limits the applicability of such models for testing biological hypotheses.

To deal with this problem, qualitative approaches, notably based on qualitative versions of differential equations, have been proposed for representing genetic regulatory networks, molecular interaction networks or metabolic pathways [30, 40]. In these models only the *direction* of influence between variables is encoded (e.g. activation vs. inhibition) and much of the quantitative information is absent. As a consequence of such under-constrained descriptions, purely-qualitative approaches often lead to overly-conservative results in the sense of admitting many spurious behaviors. We propose a technique that can be used to analyze in a systematic manner quantitative models admitting this kind of uncertainty whose nature is *set-theoretic* rather than stochastic.

The analysis techniques that we use and extend originate from the study of *hybrid dynamical systems*, a domain situated in the intersection of control theory and computer science and are based on reachability analysis of hybrid automata. As their name suggests, hybrid automata are the result of marrying automata with differential equations. Each discrete state (mode) of the automaton is associated with one set of differential equations according to which the continuous variables evolve while being in that mode. When the variables satisfy certain conditions (transition guards) the automaton may switch to another mode where another set of equations will govern the evolution of the continuous variables. While hybrid automata allow us to express piecewise-continuous processes and can underlie numerical simulation, much of the analytic reasoning available for purely-continuous systems (especially for linear ones) is lost due to switching. In the last couple of years new techniques have been developed for the algorithmic analysis of hybrid systems, which open as well new opportunities for the analysis of purely-continuous systems subject to uncertainties. These techniques combine ideas from control theory, numerical analysis, graph algorithms and computational geometry in order to export algorithmic verification, also known as *model checking*, to the continuous and hybrid domains.

The principles of algorithmic verification can be summarized as follows. The system in question is modeled as an automaton whose transitions are labeled by input events. These inputs represent interactions of the automaton with its external environment (users, other systems). Each sequence of input events induces one behavior of the automaton, a trajectory over its state space. Simulation is the process of stimulating the automaton progressively with one input sequence and observing the behavior that this sequence induces starting from a given initial state. The problem is that the number of such sequences is prohibitively large.

Verification is based, instead, on computing with *sets* of states: starting from an initial set of states P_0, one computes *all* the one-step successors of P_0 (under all possible inputs) to obtain the set P_1, to which the same procedure is applied until all the states reachable from P_0 under any admissible input are computed.[1] Showing, for example, that some "bad" set of states is never reached (a "safety" property) amounts to checking whether the reachable set thus computed intersects the bad set. This computation replaces an infinite (or just huge) number of simulations. More complex properties that specify some temporal patterns of events can be specified and verified as well using similar methods.

The adaptation of this idea to continuous systems works as follows. Consider a differential equation of the form $\dot{x} = f(x, v)$ where x is a vector of state variables and v represents external disturbances and parameter uncertainties which are not known exactly but are always taken from a bounded convex set V. Given a subset P_0 of the state space (in a form of, say, a polytope) and a time step r, one can compute another polytope P_1, which contains all the points reachable from P_0 within the time interval $[0, r]$ under *any admissible value* of v during that interval. Repeating this process we can obtain an over approximation of all the reachable states in any desired time horizon. To give a concrete example, one can compute all the possible evolutions of a reaction under all possible concentrations of a signalling molecule which are typically not precisely known, but which remain in a known interval. The principal contribution of this paper is in developing a new technique for conducting this type of analysis for *nonlinear* systems and in demonstrating its applicability on several biological models.

The rest of the paper is organized as follows. In Section 2 we give a brief introduction to the state-of-the-art in reachability computation for linear systems and explain why it cannot be applied in a straightforward manner to nonlinear systems. We then describe the *hybridization* approach [6] for handling nonlinear systems. Hybridization is based on over approximating a nonlinear system by a *piecewise-affine* system, a restricted type of a *hybrid automaton* without discontinuous jumps. Although, in principle, hybridization provides for the application of linear techniques to nonlinear systems, it suffers from inherent limitations that restrict its applicability to very low-dimensional systems. Section 3 describes our major contribution, a new *dynamic* hybridization scheme in which linearization is *not* based on a fixed partition of the state space and thus avoids much of the associated state explosion. For this algorithm we provide in Section 4 compelling experimental results, analyzing highly-nonlinear systems of 6 and 9 variables taken from systems biology. We conclude with a discussion of future work. Although we have tried to maintain the paper as self contained as possible, some readers might want to consult books like [42, 39, 28, 38] for some notions of geometry, linear algebra and dynamical systems or expository articles such as [34, 35] which discuss similarities and differences between transition systems and continuous dynamical systems.

[1] More precisely, the computation is guaranteed to converge for finite-state systems. In continuous domains we are currently satisfied with a bounded time horizon [34].

2 Reachability: Linear and Nonlinear Systems

Computing the states reachable by *all* trajectories of a dynamical system subject to disturbances and parameter variations emerged as a new research topic from the interaction between computer science and control. Reachability computation can be seen as a peculiar way to conduct *exhaustive* simulation which can be useful for the analysis of control systems, the verification of analog circuits, the debugging of biological models and, in fact, any other activity based on dynamical systems models. After a decade of intensive research, [2,25,11,15,26,4, 33,37,10,5,12,32,24] it is fair to say that a satisfactory solution has been provided for *time-invariant linear systems*. Existing algorithms manage to produce, within seconds, high-quality approximations of the reachable states of linear systems with *hundreds* of state variables, for time horizons of *thousands* of integration steps. Notwithstanding these achievements, the real challenge in almost any application domain, Biology included, is the treatment of *nonlinear* systems, a challenge that we address in the present paper.

Let us recall the rules of the game. Given a dynamical system S defined by a differential equation $\dot{x} = f(x, v)$ with v ranging over some bounded set V, a set P of initial states and some time horizon h, we would like to compute the set of states reachable from points in P by trajectories of S within some $t \in [0, h]$. Fixing some time discretization step r, the reachable set is *approximated* by the union of the sets in a sequence $P_0, P_1, \ldots$ where P_0 contains all states reachable from P within $t \in [0, r]$ and each P_{i+1} includes states reachable from P_i within r time. Actual computations often work first in discrete time where P_{i+1} is reachable from P_i in one time step and then some error terms are added to bloat P_{i+1} and compensate with respect to continuous time.

Reachability computation of linear systems is relatively easy. Consider first a discrete-time autonomous linear system defined by $x' = Ax$ and a set P which admits a finite representation, for example, a polytope represented by its vertices or supporting halfspaces, an ellipsoid represented by its center and deformation matrix or a zonotope represented by its center and generators. Then the linear transformation "commutes" with the representation. For example, if $P = conv(\tilde{P})$, meaning a polytope P being the convex hull of its finite set of vertices $\tilde{P}$, then

$$A \cdot P = A \cdot conv(\tilde{P}) = conv(A \cdot \tilde{P}), \tag{1}$$

that is, the vertices of the polytope obtained by applying A to the whole set P are the result of applying A to the vertices of P.

The extension of this idea to systems with under-specified input, that is, $x' = Ax + v$ where v ranges over a bounded convex set V, is more involved. The set of one-step successors of a set P under such a dynamics is captured by the Minkowski sum $P' = AP \oplus V$, which yields a polytope P' with more vertices than P. This repeated growth in the size of the representation of P_i makes it impractical to iterate for a long time horizon because the number of points on which A has to be evaluated becomes huge. Two approaches are commonly used to alleviate this problem:

1. For ellipsoids or for polytopes represented by their supporting halfspaces one can use techniques based on the maximum principle [41,13] to obtain an over approximation of $AP \oplus V$ whose representation size is not much larger than that of P;
2. The modified recurrence scheme of [27, 24] keeps the number of points to which the linear transformation is applied fixed. Its implementation using zonotopes [23,24], a subclass of polytopes which are closed under Minkowski sum, provides a very efficient solution which is, practically, exact for discrete time.

The technique that we present in this paper is invariant under the choice between these two approaches so we express it in terms of an abstract *successor* operator σ which, given a set P, an affine differential inclusion (see below) of the form $\dot{x} \in Ax \oplus V$ and a time step r, it produces the set $\sigma(P, A, V, r)$ containing all points reachable from points in P by trajectories of duration r of the affine dynamics. The generic linear reachability algorithm can then be written as:

Algorithm 1 (Linear Reachability)
$P_0 := \tilde{R}_{[0,r]}(P)$
repeat $i = 1, 2, \ldots$
  $P_i := \sigma(P_{i-1}, A, V, r)$
until $i = k$

The set $\tilde{R}_{[0,r]}(P)$, the over approximation of the states reachable from P within the time interval $[0, r]$, can be computed, for example, by bloating the convex hull of $P \cup \sigma(P, A, V, r)$ as in [4] or [6].

Moving to *nonlinear* systems of the form $x' = f(x)$ for arbitrary f one observes that "convexity" properties such as (1) do not hold and new ideas are needed. In principle, it is possible to evaluate f on some representative finite sample $\tilde{P} \subset P$ and then use the resulting points to construct a set which over approximates $f(P)$. However, the approximation can be very coarse and will require a costly optimization procedure to be refined, something that cannot be afforded as part of the inner loop of the reachability algorithm. The "hybridization" technique of [6] suggests a good tunable compromise between the quality of the approximation, the difficulty of the computation and the frequency in which it is invoked. Before explaining the idea, let us give some necessary definitions.

We consider a state space X, a bounded subset of $\mathbb{R}^n$ equipped with a metric ρ. Given two bounded closed subsets Y and Y' of X, the *Hausdorff distance* between them (the lifting of ρ to sets) is

$$\rho(Y, Y') = \max\{\max_{y \in Y} \min_{y' \in Y'} \rho(y, y'), \max_{y' \in Y'} \min_{y \in Y} \rho(y, y')\}.$$

The trajectories of a dynamical system are viewed as *signals* over X.

Definition 1 (Signals). *A signal over X is a partial continuous function ξ from $T = [0, \infty)$ to X whose domain of definition is T or a prefix $[0, r]$ of it. In the latter case we say that ξ is finite with duration r. The* concatenation *of a*

finite signal ξ *defined over* $[0, r]$ *and a signal* ξ' *satisfying* $\xi'(0) = \xi(r)$ *is defined in the obvious way and is denoted by* $\xi \cdot \xi'$.

The continuous equivalent of a non-deterministic automaton is the relational vector field, also known as *differential inclusion* [7].

Definition 2 (Relational Vector Fields). *A relational vector field over* X *is a function* $f : X \to 2^X - \{\emptyset\}$ *which is assumed to be* K*-Lipschitz, satisfying*

$$\rho(\{x\}, \{x'\}) < a \Rightarrow \rho(f(x), f(x')) < Ka.$$

When f is a (deterministic) function we write $f(x) = y$ rather than $f(x) = \{y\}$.

Definition 3 (Dynamical Systems, Trajectories and Reachable Sets). *A (continuous)* dynamical system *is a pair* $S = (X, f)$ *where* X *is a state space and* f *is a vector field. A* trajectory *of* S *starting from* x *is a signal* ξ *over* X *with* $\xi(0) = x$ *and for every* t *in the domain of definition of* ξ, $\xi(t) \in X$ *and* $d\xi(t)/dt \in f(\xi(t))$. *The set of all trajectories of* S *starting from any* $x \in P$ *is denoted by* $\mathcal{L}(S, P)$. *The sets of states reachable from* P *within a time interval* $[h, h']$ *is*

$$R_{[h,h']}(P) = \{\xi(t) : \xi \in \mathcal{L}(S, P) \wedge t \in [h, h']\}.$$

Hybridization takes a nonlinear system $S = (X, f)$ and produces another dynamical system (S', f') which over approximates it, that is, $\mathcal{L}(S, P) \subseteq \mathcal{L}(S', P)$ for every P, and then computes the reachable states of S'. A formal definition of S' as a hybrid automaton can be found in [6]. Since our algorithm does not use hybrid automata explicitly we only give an informal explanation.

Consider a partition of X into hyper rectangles (we use the term *box* hereafter). For each box X_q one can compute a linear function A_q and an error polytope V_q such that for every $x \in X_q$, $f(x) \in A_q x \oplus V_q$. In other words, A_q is a local *linearization* of f with error bounded in V_q. Thus the vector field f' is defined as $f'(x) = A_q x \oplus V_q$ iff $x \in X_q$. To perform reachability computation on S' one applies linear reachability using A_q and V_q as long as the reachable states remain within box X_q. Whenever some P_i reaches the boundary between X_q and $X_{q'}$ it is intersected with the switching surface (the transition guard, in the terminology of hybrid automata) and the obtained result is used as an initial set for reachability computation in q' using $A_{q'}$ and $V_{q'}$, as illustrated in Figure 1-(a,b). The main advantage of hybridization is that the costly procedure of finding a good linear approximation is not invoked in every step, only in the passage between boxes. Although this scheme is clean and general, it suffers from some serious difficulties on the way to realization:

– Although the intersection of the actual set of reachable states inside a box with a facet of the box is typically a convex set, its computation can be inefficient and inaccurate. To see why, consider a subsequence of sets $P_j, \ldots, P_k$ computed using some linear technique, all of which intersect the boundary

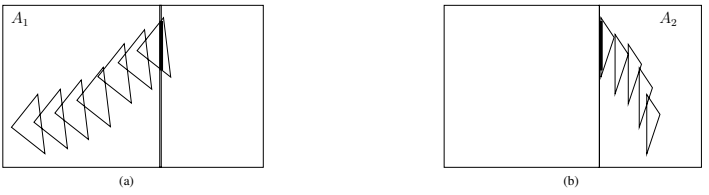


Fig. 1. Computing reachable states of the hybridization: (a) applying linear reachability using A_1 until intersection with the boundary; (b) taking the intersection as an initial set for linear reachability using A_2

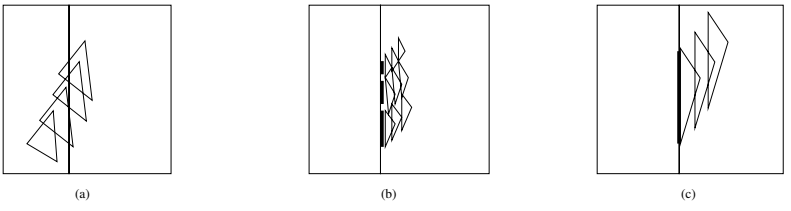


Fig. 2. (a) the intersection with the boundary spans over several iterations; (b) continuing with each intersection separately; (c) continuing with an approximation of the union of intersections

G as illustrated in Figure 2-(a). In this case we have either to spawn several computations with the dynamics of the subsequent box, each starting with some $P_i \cap G$ (Figure 2-(b)) or to over approximate $\bigcup_i P_i \cap G$ by a convex set, an operation that may lead to a large over-approximation error (Figure 2-(c)).

– The size of the partition of the state space is, of course, exponential in the dimension, hence care should be taken in order to avoid state explosion. As suggested in [6], the partition can be generated *on-the-fly* as the reachability computation evolves, rather than being precomputed for the whole state space in advance. However, even on-the-fly generation cannot cope with the fact that in high dimension, a tube of reachable states will typically leave a box via *exponentially* many facets. This situation is illustrated in Figure 3-(a). Since each of these parts of the reachable set goes to a different box, they have to be handled separately (Figure 3-(b)) even though they continue to evolve close to each other.[2] Merging these sets when they converge to the same box is a tedious process and a source of further approximation errors. This problem is particularly severe because making the boxes smaller is the recommended recipe for improving accuracy.

[2] A similar phenomenon has been encountered in the analysis of timed automata [9].

3 Dynamic Hybridization

In this section we describe our novel nonlinear reachability algorithm which, unlike the scheme of [6], is not based on a fixed partitioning of the state space but rather generates overlapping linearization domains around the reachable states. An important ingredient of any hybridization methodology is the linearization procedure that we first define formally.

Definition 4 (Linearization in a Domain). *A linearization operator is a function L which, for a given nonlinear function f and a convex set B (linearization domain), produces a matrix A and a convex polytope V such that for every $x \in B$, $f(x) \in Ax \oplus V$.*

We use the notation $L(f, B) = (A, V)$. In our current implementation the linearization domains are boxes, but other forms are possible. In addition to the linearization operator L and the linear successor operator σ we assume a procedure β which takes as input a set P and produces a linearization domain $B = \beta(P)$ which contains P. The form of B, the relation between its size and the size of P as well as the position of P inside B are implementation details that may vary according to the system in question. We first present in general terms the algorithm for approximates the reachable states, prove its correctness and then discuss our implementation of L and β.

Algorithm 2 (Dynamic Hybridization) Input: *A nonlinear dynamical system $S = (X, f)$ and an initial set P*
Output: *A sequence of sets $P_0, P_1, \ldots P_k$ whose union includes $R_{[0,h]}(P)$*

$B := \beta(P)$
$(A, V) := L(f, B)$
$P_0 := \tilde{R}_{[0,r]}(P)$
$i := 0$
repeat
  $P_{i+1} := \sigma(P_i, A, V, r)$
  if $P_{i+1} \subseteq B$
    $i := i + 1$
  else
    $B := \beta(P_i)$
    $(A, V) := L(f, B)$
until $i = k$

The algorithm performs linear reachability in a linearization domain B as long as the computed sets remain inside B. Once a newly-computed set P_{i+1} is not fully contained in B we backtrack to P_i and construct a new domain B' around P_i along with its corresponding linearization which is used for subsequent computations starting from P_i, as illustrated in Figure 4. The advantage of this approach is obvious: the linearization mesh is constructed *along the reachable set* and thus we avoid artificial splitting of sets due to the structure of the mesh. Needless to say, the intersection operation is altogether avoided.

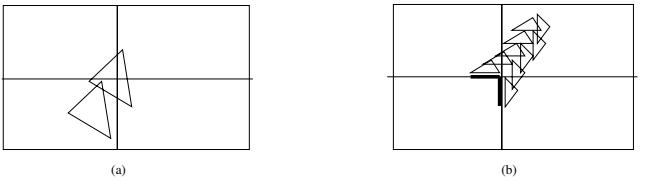


Fig. 3. (a) the reachable set leaves a box through several boundaries; (b) the computation is continued separately for each intersection although the computed sets remain close to each other and even go later to the same box

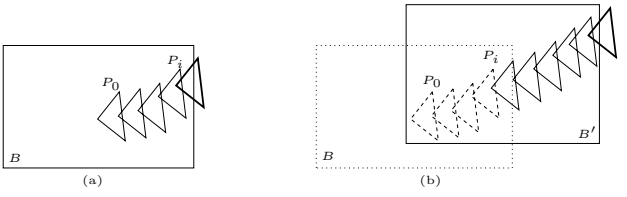


Fig. 4. Dynamic hybridization: (a) Computing in some box until intersection with the boundary; (b) Backtracking one step and computing in a new box

Theorem 1 (Correctness of Algorithm 2). *Let $P_0, P_1, \ldots$ be a sequence of sets produced by Algorithm 2. Then for every $k \leq k'$, we have*

$$R_{[kr,k'r]}(P) \subseteq \bigcup_{i=k}^{k'} P_i.$$

Proof. The proof is by induction on the number of switchings between linearization domains that the algorithm makes. The base case where no switching occurs follows from the correctness of the linear reachability algorithm and the fact that the linearized system over approximates f. For the inductive case, assume the claim holds for s switchings and consider a run of the algorithm with $s+1$ switchings, the last of which occurring after P_j, $k \leq j < k'$. By the inductive hypothesis $R_{[jr,jr]}(P) \subseteq P_j$ and since P_j serves as the initial set for subsequent iterations inside a single linearization domain, the base case applies and $P_{j+1}, \ldots, P_{k'}$ includes $R_{[(j+1)r,k'r]}(P)$ which, together with $P_k, \ldots P_j$, include the states reachable within $[kr, k'r]$. ∎

Algorithm 2 is implemented in C and uses the polytope-based algorithms of **d/dt** [13]. Below we explain the novel technical aspects, namely the dynamic construction of the linearization domain and its respective linearization.

The difference between the function f and its linear approximation A relative to a domain B is $\Delta_B(f, A) = \{f(x) - Ax : x \in B\}$. To guarantee conservative approximation it is sufficient to find some V such that $\Delta_B(f, A) \subseteq V$ and this can be done easily for *any* choice of a domain B and a linearization A. However to

obtain high-quality approximations, we need to choose B and A that minimize, roughly speaking, the diameter of $\Delta_B(f, A)$ which represents the error incurred by the linear over approximation. Clearly the smaller is B, the smaller is the error but then the linearization procedure has to be invoked more frequently. The problem of finding good B and A can be formulated, in principle, as some sort of a constrained *optimization* problem but this computation can be very costly and we use instead the following easy-to-compute heuristic which turns out to work in practice despite being non optimal. The first simplification that we do with respect to an optimized solution is to decouple the choice of the new domain $B = \beta(P)$ from the computation of the linearization $(A, V) = L(B, f)$.

The operator $\beta(P)$ which produces a box containing P is realized as follows. Based on f and X we fix a standard rectangular frame $\mathcal{B}$ of size $d_1 \times, \cdots, \times d_n$. Given a polytope P we define its centroid $c(P)$ to be the average of its vertices and let $\beta(P)$ be a copy of $\mathcal{B}$ whose center coincides with $c(P)$. The only problematic situation occurs when during reachability computation P gets too large and cannot fit (either immediately or after few steps) within the frame $\mathcal{B}$. To prevent Algorithm 2 from getting stuck in the *else* branch, we split P into two or more sets which are then treated separately. In principle, this splitting may lead to state explosion but, in this case, the explosion is due to *intrinsic properties* of the set of reachable states and not due to an arbitrary choice of the coordinate system underlying the mesh. This phenomenon will not occur too often while analyzing stable systems having a contracting dynamics.

To handle the splitting we first compute a tight bounding box $B(P)$ around P. This computation is performed by projecting the vertices on each of the dimensions and taking the minimum and maximum. Let us denote by $e_1 \times, \cdots, \times e_n$ the size of the obtained bounding box. If for every i, $m \cdot e_i < d_i$, where $m > 1$ is a fixed constant, then P is sufficiently small and no splitting takes place. Otherwise we take the direction i which maximizes the ratio e_i/d_i and split P into two parts along this direction by intersecting it with complementary halfspaces orthogonal to direction i (see Figure 5). We repeat the process until the obtained sets are sufficiently small. We thus end up with one or more polytopes around each of which we put a properly-centered copy of $\mathcal{B}$.

Once the linearization domain B is fixed we compute A and V as follows. Let $f = (f_1, \ldots, f_n)$, and let $y = c(B)$ be the *center* of B. The matrix A is obtained by the evaluating (numerically) the Jacobian matrix of f at y, that is,

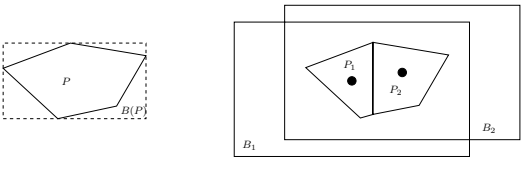


Fig. 5. A set P and its bounding box $B(P)$. The set is too large and is split in the vertical dimension into P_1 and P_2, around which the respective linearization domains B_1 and B_2 are constructed.

$A = \frac{\partial f}{\partial x}(y)$ where $A_{ij} = \frac{\partial f_i}{\partial x_j}$. Then a box $V = V_1 \times V_2 \times \ldots \times V_n$, guaranteed to contain $\Delta_B(f, A)$, is computed as follows. For each dimension i we let V_i be the interval $[l_i, u_i]$ where $l_i = \min\{\pi_i(\Delta_B(f, A))\}$ and $u_i = \max\{\pi_i(\Delta_B(f, A))\}$ with π_i denoting projection on i. These intervals are over approximated based on the Taylor expansion of $f(x) - Ax$.

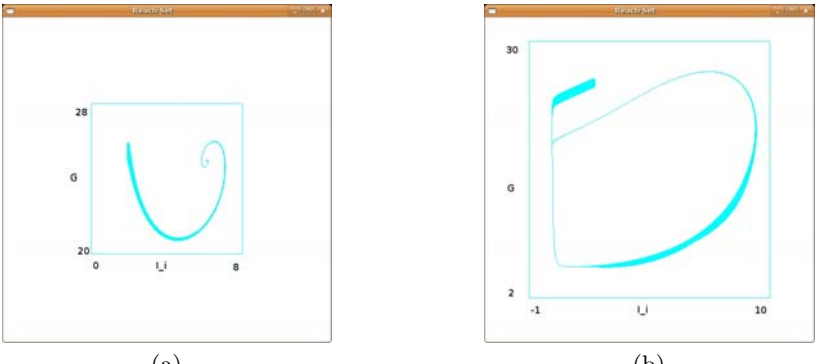
4 Experimental Results

To test the feasibility of our algorithm we applied it to two nonlinear systems whose parameters and qualitative behaviors are documented in the literature.

The *Lac Operon* is a biochemical feedback mechanism through which the bacterium *E. Coli* adapts to the lack of Glucose in its environment by switching to a Lactose diet. We use the model appearing in [31] where the behavior of the system is described by the following system of differential equations:

$$\begin{aligned}
\dot{R}_a &= \tau - \mu * R_a - k_2 R_a O_f + k_{-2}(\chi - O_f) - k_3 R_a I_i^2 + k_8 R_i G^2 \\
\dot{O}_f &= -k_2 r_a O_f + k_{-2}(\chi - O_f) \\
\dot{E} &= \nu k_4 O_f - k_7 E \\
\dot{M} &= \nu k_4 O_f - k_6 M \\
\dot{I}_i &= -2k_3 R_a I_i^2 + 2k_{-3} F_1 + k_5 I_r M - k_{-5} I_i M - k_9 I_i E \\
\dot{G} &= -2k_8 R_i G^2 + 2k_{-8} R_a + k_9 I_i E
\end{aligned}$$

The differential variables denote the concentrations of different reactants, such as R_a (active repressor) O_f (free operator), E (enzyme), M (mRNA), I_i (internal inducer), and G (glucose). We studied the behavior of this 6-dimensional system around a quasi-steady state for the first 4 variables and the obtained results are consistent with the simulation results obtained on a simplified 2-dimensional model shown in [31], page 285. As a set of initial states we take a small box where



(a) (b)

Fig. 6. Lac operon: (a) a stable focus, $k_{-1} = 2.0$; (b) a limit cycle, $k_{-1} = 0.008$

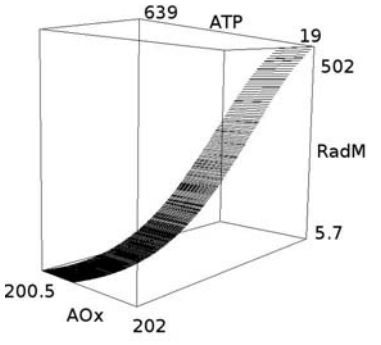


Fig. 7. Results obtained for the aging model

$I_i \in [1.9, 2.0]$ and $G \in [25.9, 26]$. When $k_{-1} = 2.0$ the system exhibits a stable focus and when $k_{-1} = 0.008$ the system exhibits a limit cycle (see Figure 6). Computation times are 3 and 5 minutes, respectively.

We conclude with a model of an *aging process*, based on the mitochondrial theory of aging. The highly-nonlinear differential equations, which include ratios between variables, can be found in [31], page 252. The model admits 9 variables and we show in Figure 7 the reachable set after 300 iterations projected on 3 variables, namely, the concentration of antioxidants (AOx), of radicals (RAdM) which suffer damages, and of ATP (adenosine triphosphate). After 1000 iterations, we observe the convergence towards a steady state. The computation time for 1000 iterations is 23.3 minutes.

5 Discussion

We made progress toward a very ambitious goal: automatic reachability analysis of nonlinear systems as a methodology for investigating under-specified biological models. Let us mention other attempts to solve this problem starting with methods that share with hybridization the idea of approximating the original systems by partitioning the continuous state space and producing a hybrid automaton with a simpler dynamics in each state. In the extreme case where no continuous dynamics is left the finite automaton is the sole responsible for approximating the dynamics. This approach is common in AI and qualitative physics and has been used extensively in Biology [21, 30, 19]. The technique of predicate abstraction applied to hybrid systems [3] is another elaboration of this idea where partition boundaries are based on predicates appearing in specifications. A more refined approach, incorporated into the tools HyTech [20] and PHAVer [17] over approximates the nonlinear system by hybrid automata where in each state the dynamics is defined by a *constant differential inclusion* of the form $A\dot{x} \leq c$. Since in each state the derivative does not depend on the real variables, it is easy to compute the reachable states exactly using linear algebra, however the over approximation with respect to the original system is large (zero-order

compared to first-order approximation in the hybridization of [6]). The translation of continuous systems into timed automata [36] is another example.

Other, more direct, approaches perform reachability on the original nonlinear systems without relying on convexity properties. For example, the face lifting technique [25, 15, 26], which is based on computing the maximal projections of f on all the normals of the facets of a polyhedron, may lead to large over-approximation errors. Other approaches use more complex classes of sets which are not necessarily convex. In [37] the evolution of the reachable states is transformed into a partial differential equation (PDE) where the boundary of the set is represented as the set of zeros of a function defined over the state space. The work of [14] uses Bezier simplices to represent reachable states for systems defined by polynomial differential equations. Finally in [18,1] dynamic linearization and computation of error bounds is performed at every reachability step. None of these methods, to the best of our knowledge, can cope with systems of the size and complexity of the examples presented in this paper.

Let us also mention the whole domain of *interval analysis* [40], a branch of numerical analysis motivated by producing rigorous numerical answers to diverse mathematical questions despite round-off errors. As its name suggests, for the computation of a scalar function, the result is typically an interval guaranteed to contain the correct answer. The generalization to many dimensions leads naturally to bounding boxes. Although the motivation is different from ours as the uncertainty is due to the computation itself rather than the imperfection of the model, there are similarities between some of the techniques and we foresee more future cross fertilization between the domains.

Parameter uncertainty in biological models is a well-known problem that has been subject to extensive work using various techniques. We mention two recent attacks on the problem of *parameter synthesis*, namely, finding or approximating the range of model parameters for which some qualitative behavior is exhibited. The work of [8] takes a hybrid model (piecewise multi-affine dynamics) with parameter uncertainty and abstracts it into a finite automaton. When the property in question is violated by the automaton, the domain of parameter values is refined, a new abstraction is created and so on. A more direct and efficient way to explore the space of parameter values is described in [16] based on adaptive sampling of the parameter space and using ordinary numerical simulation. This technique uses numerical sensitivity information to guide the refinement of the parameter space.

To go further we need to combine dynamic hybridization with the algorithm of [24,27] which can treat linear systems an order of magnitude larger than those treated in the present paper. To this end we need to develop a good splitting procedure for the sets computed by that algorithm. As the reader might have noticed, we have focused here on systems where the uncertainty is restricted to the initial set and we need to extend our linearization operator to nonlinear functions with input, something that can be done using similar principles.

To conclude, we have demonstrated the feasibility of our approach by computing reachable states for nonlinear systems of unpreceded size and complexity.

We intend to pursue this direction further and make reachability computation a useful tool for analyzing complex biological systems. A parallel effort should be invested in making modelers of biological systems aware of the potential of this analysis technology.

References

1. Althoff, M., Stursberg, O., Buss, M.: Reachability Analysis of Nonlinear Systems with Uncertain Parameters using Conservative Linearization. In: CDC 2008 (2008)
2. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The Algorithmic Analysis of Hybrid Systems. Theoretical Computer Science 138, 3–34 (1995)
3. Alur, R., Dang, T., Ivancic, F.: Counterexample-guided Predicate Abstraction of Hybrid Systems. Theoretical Computer Science 354, 250–271 (2006)
4. Asarin, E., Bournez, O., Dang, T., Maler, O.: Approximate Reachability Analysis of Piecewise Linear Dynamical Systems. In: Lynch, N.A., Krogh, B.H. (eds.) HSCC 2000. LNCS, vol. 1790, pp. 21–31. Springer, Heidelberg (2000)
5. Asarin, E., Dang, T.: Abstraction by Projection and Application to Multi-affine Systems. In: Alur, R., Pappas, G.J. (eds.) HSCC 2004. LNCS, vol. 2993, pp. 32–47. Springer, Heidelberg (2004)
6. Asarin, E., Dang, T., Girard, A.: Hybridization Methods for the Analysis of Nonlinear Systems. Acta Informatica 43, 451–476 (2007)
7. Aubin, J.-P., Cellina, A.: Differential Inclusions. Springer, Heidelberg (1984)
8. Batt, G., Belta, C., Weiss, R.: Model Checking Genetic Regulatory Networks with Parameter Uncertainty. In: Bemporad, A., Bicchi, A., Buttazzo, G. (eds.) HSCC 2007. LNCS, vol. 4416, pp. 61–75. Springer, Heidelberg (2007)
9. Ben Salah, R., Bozga, M., Maler, O.: On Interleaving in Timed Automata. In: Baier, C., Hermanns, H. (eds.) CONCUR 2006. LNCS, vol. 4137, pp. 465–476. Springer, Heidelberg (2006)
10. Botchkarev, O., Tripakis, S.: Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In: Lynch, N.A., Krogh, B.H. (eds.) HSCC 2000. LNCS, vol. 1790, pp. 73–88. Springer, Heidelberg (2000)
11. Chutinan, A., Krogh, B.H.: Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations. In: Vaandrager, F.W., van Schuppen, J.H. (eds.) HSCC 1999. LNCS, vol. 1569, pp. 76–90. Springer, Heidelberg (1999)
12. Chutinan, A., Krogh, B.H.: Computational Techniques for Hybrid System Verification. IEEE Trans. on Automatic Control 48, 64–75 (2003)
13. Dang, T.: Verification and Synthesis of Hybrid Systems, PhD thesis, Institut National Polytechnique de Grenoble, Laboratoire Verimag (2000)
14. Dang, T.: Approximate Reachability Computation for Polynomial Systems. In: Hespanha, J.P., Tiwari, A. (eds.) HSCC 2006. LNCS, vol. 3927, pp. 138–152. Springer, Heidelberg (2006)
15. Dang, T., Maler, O.: Reachability Analysis via Face Lifting. In: Henzinger, T.A., Sastry, S.S. (eds.) HSCC 1998. LNCS, vol. 1386, pp. 96–109. Springer, Heidelberg (1998)
16. Donze, A., Clermont, G., Legay, A., Langmead, C.J.: Parameter Synthesis in Nonlinear Dynamical Systems: Application to Systems Biology. In: RECOMB 2009, pp. 155–169 (2009)

17. Frehse, G.: PHAVer: Algorithmic Verification of Hybrid Systems Past HyTech. In: Morari, M., Thiele, L. (eds.) HSCC 2005. LNCS, vol. 3414, pp. 258–273. Springer, Heidelberg (2005)
18. Han, Z., Krogh, B.H.: Reachability Analysis of Nonlinear Systems using Trajectory Piecewise Linearized Models. In: American Control Conference, pp. 1505–1510 (2006)
19. Halasz, A., Kumar, V., Imielinski, M., Belta, C., Sokolsky, O., Pathak, S.: Analysis of Lactose Metabolism in E.coli using Reachability Analysis of Hybrid Systems. IEE Proceedings - Systems Biology 21, 130–148 (2007)
20. Henzinger, T.A., Ho, P.-H., Wong-Toi, H.: Algorithmic Analysis of Nonlinear Hybrid Systems. IEEE Trans. on Automatic Control 43, 540–554 (1998)
21. de Jong, H., Page, M., Hernandez, C., Geiselmann, J.: Qualitative Simulation of Genetic Regulatory Networks: Method and Application. In: IJCAI 2001, pp. 67–73 (2001)
22. Gillespie, D.T.: Stochastic Simulation of Chemical Kinetics. Annual Review of Physical Chemistry 58, 35–55 (2007)
23. Girard, A.: Reachability of Uncertain Linear Systems using Zonotopes. In: Morari, M., Thiele, L. (eds.) HSCC 2005. LNCS, vol. 3414, pp. 291–305. Springer, Heidelberg (2005)
24. Girard, A., Le Guernic, C., Maler, O.: Efficient Computation of Reachable Sets of Linear Time-invariant Systems with Inputs. In: Hespanha, J.P., Tiwari, A. (eds.) HSCC 2006. LNCS, vol. 3927, pp. 257–271. Springer, Heidelberg (2006)
25. Greenstreet, M.R.: Verifying Safety Properties of Differential Equations. In: Alur, R., Henzinger, T.A. (eds.) CAV 1996. LNCS, vol. 1102, pp. 277–287. Springer, Heidelberg (1996)
26. Greenstreet, M.R., Mitchell, I.: Reachability Analysis Using Polygonal Projections. In: Vaandrager, F.W., van Schuppen, J.H. (eds.) HSCC 1999. LNCS, vol. 1569, pp. 103–116. Springer, Heidelberg (1999)
27. Le Guernic, C.: Calcul efficace de l'ensemble atteignable des systémes linaires avec incertitudes, Master's thesis, Université Paris 7 (2005)
28. Hirsch, M., Smale, S.: Differential Equations, Dynamical Systems and Linear Algebra. Academic Press, London (1974)
29. 29 Jaulin, L., Kieffer, M., Didrit, O., Walter, E.: Applied Interval Analysis. Springer, Heidelberg (2001)
30. de Jong, H., Page, M., Hernandez, C., Geiselmann, J.: Qualitative Simulation of Genetic Regulatory Networks: Method and Application. In: IJCAI 2001, pp. 67–73 (2001)
31. Klipp, E., Herwig, R., Kowald, A., Wierling, C., Lehrach, H.: Systems Biology in Practice: Concepts, Implementation and Application. Wiley, Chichester (2005)
32. Kurzhanskiy, A., Varaiya, P.: Ellipsoidal Techniques for Reachability Analysis of Discrete-time Linear Systems. IEEE Trans. Automatic Control 52, 26–38 (2007)
33. Kurzhanski, A., Varaiya, P.: Ellipsoidal tehcniques for reachability analysis. In: Lynch, N.A., Krogh, B.H. (eds.) HSCC 2000. LNCS, vol. 1790, p. 202. Springer, Heidelberg (2000)
34. Maler, O.: A Unified Approach for Studying Discrete and Continuous Dynamical Systems. In: CDC 1998, pp. 2083–2088 (1998)
35. Maler, O.: Control from Computer Science. Ann. Rev. in Control 26, 175–187 (2002)
36. Maler, O., Batt, G.: Approximating Continuous Systems by Timed Automata. In: Fisher, J. (ed.) FMSB 2008. LNCS (LNBI), vol. 5054, pp. 77–89. Springer, Heidelberg (2008)

37. Mitchell, I., Tomlin, C.J.: Level Set Methods for Computation in Hybrid Systems. In: Lynch, N.A., Krogh, B.H. (eds.) HSCC 2000. LNCS, vol. 1790, pp. 310–323. Springer, Heidelberg (2000)
38. Sastry, S.: Nonlinear systems. Analysis, Stability and Control. Springer, Heidelberg (1999)
39. Schrijver, A.: Theory of Linear and Integer Programming. Wiley, Chichester (1986)
40. Thomas, R., D'Ari, R.: Biological Feedback. CRC Press, Boca Raton (1990)
41. Varaiya, P.: Reach Set computation using Optimal Control. In: KIT Workshop, pp. 377–383. Verimag, Grenoble (1998)
42. Zeigler, G.M.: Lectures on Polytpoes. Springer, Heidelberg (1995)