Algorithmic Verification of Continuous and Hybrid Systems

Oded Maler
CNRS-VERIMAG,
University of Grenoble
France

We provide a tutorial introduction to *reachability computation*, a class of computational techniques that exports verification technology toward continuous and hybrid systems. For open under-determined systems, this technique can sometimes replace an infinite number of simulations.

1 Introduction

The goal of this article is to introduce the fundamentals of algorithmic verification of *continuous dynamical systems* defined by *differential equations* and of hybrid dynamical systems defined by *hybrid automata* which are dynamical systems that can switch between several modes of continuous dynamics. There are two types of audience that I have in mind. The first is verification-aware computer scientists who know finite-state automata and their algorithmic analysis. For this audience, the conceptual scheme underlying the presented algorithms will be easy to grasp as it is mainly inspired by symbolic model-checking of non-deterministic automata. Putting aside dense time and differential equations, dynamical systems can be viewed by this audience as a kind of infinite-state reactive program defined over the so-called real numbers. For this reason, I will switch from continuous to discrete-time discourse quite early in the presentation.

The other type of audience is people coming from control or other types of engineering and applied mathematics. These are relatively well versed in the concrete mathematics of continuous systems and should first be persuaded that the verification question is interesting, despite the fact that airplanes can fly (and theoretical papers can be written) without it. In my attempts to accommodate these two types of audience, I have written some explanation that will look trivial to some but this cannot be avoided while trying to do genuine inter-disciplinary research[1] (see also [65] for an attempt to unify discrete and continuous systems and [69] for some historical reflections). My intention is to provide a synthetic introduction to the topic rather than an exhaustive survey, hence the paper is strongly biased toward techniques closer to my own research. In particular, I will not deal with decidability results for hybrid systems with simplified dynamics and not with deductive verification methods that use invariants, barriers and Lyapunov functions. I sincerely apologize to those who will not find citations of their relevant work.

The rest of the paper is organized as follows. Section 2 describes the problem and situates it in context. Section 3 gives the basic definitions of reachability notions used throughout the paper. Section 4 presents the principles of set-based computation as well as basic issues related to the computational treatment of sets in general and convex polytopes in particular. Section 5 is devoted to reachability techniques for *linear* and affine dynamical systems in both discrete and continuous time, the domain

[1] The need to impress the members of one's own community is perhaps the main reason for the sterility of many attempts to do inter-disciplinary research.

L. Clemente and L. Holik (Eds.): 15th International Workshop on
Verification of Infinite-State Systems (INFINITY'13)
EPTCS 140, 2014, pp. 48–69, doi:10.4204/EPTCS.140.4

where a lot of progress has been made in recent years. The extension of these techniques to hybrid and non-linear systems, an active area of research, is discussed in Section 6. I conclude with some discussion of related work and new research directions.

2 The Problem

This paper is concerned with the following problem that we define first in a quasi-formal manner:

Consider a continuous dynamical system with input defined over some bounded state space X and governed by a differential equation of the form[2]

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

where $v[t]$ ranges, for every t, over some pre-specified bounded set V of admissible input values. Given a set $X_0 \subset X$, compute all the states visited by trajectories of the system starting from any $x_0 \in X_0$.

The significance of this question to control is the following: consider a controller that has been designed and connected to its plant and which is subject to external disturbances modeled by v. Computing the reachable set allows one to verify that all the behaviors of the closed-loop system stay within a desired range of operation and do not reach a forbidden region of the state space. Proving such properties for systems subject to uncontrolled interaction with the external environment is the main issue in verification of programs and digital hardware from which this question originates. In verification you have a large automaton with inputs that represent non-deterministic (non-controllable) effects such as behaviors of users and interactions with other systems and you would like to know whether there is an input sequence that drives the automaton into a forbidden state.

Before going further, let me try to situate this problem in the larger control context. After all, control theory and practice have already existed for many years without asking this question nor trying to answer it. This question distinguishes itself from traditional control questions in the following respects:

1. It is essentially a *verification* rather than a *synthesis* question, that is, the controller is assumed to exist already. However, it has been demonstrated that variants of reachability computation can be used for synthesizing switching controllers for timed [14, 8, 21] and hybrid [9] systems.
2. External disturbances are modeled *explicitly* as a *set* of admissible inputs, which is not the case for certain control formulations.[3] These disturbances are modeled in a *set-theoretic* rather than *stochastic* manner, that is, only the set of *possible* disturbances is specified without any probability induced over it. This makes the system in question look like a system defined by *differential inclusions* [16] which are the continuous analog of non-deterministic automata: if you project away the input you move from $\dot{x} = f(x, v)$ to $\dot{x} \in F(x)$. Adding probabilities the the inputs will yield a kind of a stochastic differential equation [15].
3. The information obtained from reachability computation covers also the *transient behavior* of the system in question, and not only its *steady-state* behavior. This property makes the approach particularly attractive for the analysis of *hybrid* (discrete-continuous, numerical-logical) systems where the applicability of analytic methods is rather limited. Such hybrid models can express, for example, deviation from idealized linear models due to constraints and saturation as well as other switching phenomena such as thermostat-controlled heating or gear shifting, see [66] for

[2] We use the physicists' notation where $\dot{x}$ indicates dx/dt. It is amusing to note that the idea of having a dedicated notation for the special variable called Time, is not unique to Temporal Logic.

[3] See [68] for a short discussion of this intriguing fact.

a lightweight introduction to hybrid systems and more elaborate accounts in books, surveys and lecture notes such as [74, 17, 64, 48, 61, 80, 76, 30, 20, 4].

4. The notion of *to compute* has a more effective flavor, that is, to develop algorithms that produce a representation of the set of reachable states (or an approximation of it) which is computationally usable, for example, it can be checked for intersection with a bad set of states.

Perhaps the most intuitive explanation of what is going on in reachability computation (and verification in general) can be given in terms of *numerical simulation*, which is by far the most commonly-used approach for validating complex systems. Each individual simulation consists of picking *one* initial condition and *one* input stimulus (random, periodic, step, etc.), producing the corresponding trajectory using numerical integration and observing whether this trajectory behaves properly. Ideally, to be on the safe side, one would like to repeat this procedure with *all* possible disturbances which are uncountably many. Reachability computation achieves the same effect as *exhaustive* simulation by exploring the state space in a "breadth-first" manner: rather than running each individual simulation to completion and then starting a new one, we compute at each time step all the states reachable by *all* possible one-step inputs from states reachable in the previous step (see [67] for a more elaborate development of this observation and [70] for a more general discussion of *under-determined* systems and their simulation). This set-based simulation is, of course, much more costly than the simulation of an individual trajectory but it provides more confidence in the correctness of the system than a small number of individual simulations would. The paper is focused on one popular approach to reachability computation based on discretizing time and performing a kind of set-based numerical integration. Alternative approaches are mentioned briefly at the end.

3 Preliminaries

We assume a time domain $T = \mathbb{R}_+$ and a state space $X \subseteq \mathbb{R}^n$. A trajectory is a measurable partial function $\xi : T \to X$ defined over all T (infinite trajectory) or over an interval $[0,t] \subset T$ (a finite trajectory). We use the notation $\mathscr{T}(X)$ for all such trajectories and $|\xi| = t$ to denote the length (duration) of finite signals. We consider an input space $V \subseteq \mathbb{R}^m$ and likewise use $\mathscr{T}(V)$ to denote input signals $\zeta : T \to V$. A continuous dynamical system $S = (X, V, f)$ is a system defined by the differential equation

$$\dot{x} = f(x, v). \tag{1}$$

We say that ξ is the *response* of f to ζ from x if ξ is the solution of (1) for initial condition x and $v(.) = \zeta$. We denote this fact by $\xi = f_x(\zeta)$ and also as

$$x \xrightarrow{\zeta/\xi} x'$$

when $|\zeta| = t$ and $\xi[t] = x'$. In this case we say that x' is reachable from x by ζ within t time and write this as

$$R(x, \zeta, t) = \{x'\}.$$

This notion speaks of *one* initial state, *one* input signal and *one* time instant and its generalization for a *set* X_0 of initial states, for *all* time instants in an interval $I = [0,t]$ and for *all* admissible input signals in $\mathscr{T}(V)$ yields the definition of the reachable set:

$$R_I(X_0) = \bigcup_{x \in X_0} \bigcup_{t \in I} \bigcup_{\zeta \in \mathscr{T}(V)} R(x, \zeta, t).$$

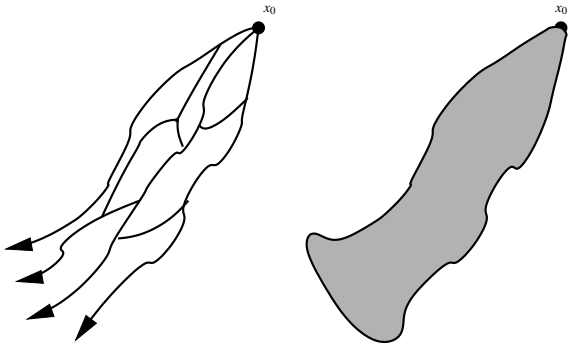

Figure 1: Trajectories induced by input signals from x_0 and the set of reachable states.

Figure 1 illustrates the induced trajectories and the reachable states for the case where $X_0 = \{x_0\}$. We will use the same R_I notation also when I is not an interval but an arbitrary time set. For example $R_{[1..r]}(X_0)$ can denote either the states reachable from X_0 by a continuous-time systems at discrete time instants, or states reachable by a discrete-time system during the first r steps.

Note that our introductory remark equating the relation between simulation and reachability computation to the relation between breadth-first and depth-first exploration of the space of trajectories corresponds to the commutativity of union:

$$\bigcup_{t\in I}\bigcup_{\zeta\in\mathscr{T}(V)} R(x,\zeta,t) = \bigcup_{\zeta\in\mathscr{T}(V)}\bigcup_{t\in I} R(x,\zeta,t).$$

4 Principles

In what follows we lay down the principles of one of the most popular approaches for computing reachable sets which is essentially a set-based extension of numerical integration.

4.1 The Abstract Algorithm

The *semigroup property* of dynamical systems, discrete and continuous alike, allows one to compute trajectories incrementally. The reachability operator also admits this property which is expressed as:

$$R_{[0,t_1+t_2]}(X_0) = R_{[0,t_2]}(R_{[0,t_1]}(X_0)).$$

Hence, the computation of $R_I(X_0)$ for an interval $I = [0,L]$ can be carried out by picking a time step r and executing the following algorithm:

Algorithm 1 (Abstract Incremental Reachability)

Input: *A set* $X_0 \subset X$
Output: $Q = R_{[0,L]}(X_0)$

$P := Q := X_0$
repeat $i = 1, 2 \ldots$
  $P := R_{[0,r]}(P)$
  $Q := Q \cup P$
until $i = L/r$

Remark: When interested in reachability for *unbounded* horizon, the termination condition $i = L/r$ should be replaced by $P \subseteq Q$, that is, the newly-computed reachable states are included in the set of states already computed. With this condition the algorithm is not guaranteed to terminate. Throughout most of this article we focus on reachability problems for a bounded time horizon.

4.2 Representation of Sets

The most urgent thing needed in order to convert the above scheme into a working algorithm is to choose a class of subsets of X that can be *represented* in the computer and be subject to the *operations* appearing in the algorithm. This is a very important issue, studied extensively (but often in low dimension) in computer graphics and computational geometry, but less so in the context of dynamical systems and control, hence we elaborate on it a bit bringing in, at least informally, some notions related to *effective computation*.

Mathematically speaking, subsets of $\mathbb{R}^n$ are defined as those points that satisfy some predicate. Such predicates are *syntactic* descriptions of the set and the points that satisfy them are the *semantic* objects we are interested in. The syntax of mathematics allows one to define weird types of sets which are not subject to any useful computation, for example, the set of irrational numbers. In order to compute we need to restrict ourselves to (syntactically characterized) classes of sets that satisfy the following properties:

1. Every set P in the class $\mathscr{C}$ admits a finite representation.
2. Given a representation of a set $P \in \mathscr{C}$ and a point x, it is possible to check in a finite number of steps whether $x \in P$.
3. For every operation $\circ$ on sets that we would like to perform and every $P_1, P_2 \in \mathscr{C}$ we have $P_1 \circ P_2 \in \mathscr{C}$. Moreover, given representations of P_1 and P_2 it should be possible to compute a representation of $P_1 \circ P_2$.

The latter requirement is often referred to as $\mathscr{C}$ being effectively *closed* under $\circ$. This requirement will later be relaxed into requiring that $\mathscr{C}$ contains a reasonable *approximation* of $P_1 \circ P_2$. To illustrate these notions, let us consider first a negative example of a class of sets admitting a finite representation but not satisfying requirements 2 and 3 above. The reachable set of a linear system $\dot{x} = Ax$ can be "computed" and represented by a finite formula of the form

$$R_I(X_0) = \{x : \exists x_0 \in X_0 \; \exists t \in I \; x = x_0 e^{At}\},$$

however this representation is not very useful because, in the general case, checking the membership of a point x in this set amounts to solving the reachability problem itself! The same holds for checking

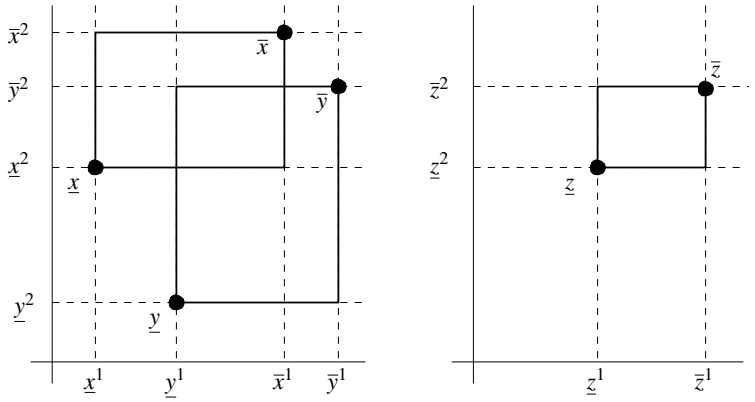

Figure 2: Intersecting two rectangles represented as $\langle \underline{x}, \overline{x} \rangle$ and $\langle \underline{y}, \overline{y} \rangle$ to obtain a rectangle represented as $\langle \underline{z}, \overline{z} \rangle$. The computation is done by letting $\underline{z}^1 = \max(\underline{x}^1, \underline{y}^1)$, $\underline{z}^2 = \max(\underline{x}^2, \underline{y}^2)$, $\overline{z}^1 = \min(\overline{x}^1, \overline{y}^1)$ and $\overline{z}^2 = \min(\overline{x}^2, \overline{y}^2)$.

whether this set intersects another set. On the other hand, a set defined by a quantifier-free formula of the form

$$\{x : g(x) \geq 0\},$$

where g is some computable function, admits in principle a membership check for every x: just evaluate $g(x)$ and compare with 0.

As a further illustration consider one of the simplest classes of sets, hyper-rectangles with axes-parallel edges and rational endpoints. Such a hyper-rectangle can be represented by its leftmost and rightmost corners $\underline{x} = (\underline{x}^1, \ldots, \underline{x}^n)$ and $\overline{x} = (\overline{x}^1, \ldots, \overline{x}^n)$. The set is defined as all points $x = (x^1, \ldots, x^n)$ satisfying

$$\bigwedge_{i=1}^{n} \underline{x}^i \leq x^i \leq \overline{x}^i,$$

a condition which is easy to check. As for operations, this class is effectively closed under translation (just add the displacement vector to the endpoints), dilation (multiply the endpoints by a constant) but not under rotation. As for Boolean set-theoretic operations, it is not hard to see that rectangles are effectively closed under intersection by component-wise max of their leftmost corners and component-wise min of their rightmost corners, see Figure 2. However they are not closed under union and complementation. This is, in fact, a general phenomenon that we encounter in reachability computations, where the basic sets that we work with are convex, but their union is not and hence the reachable sets computed by concrete realizations of Algorithm 1 will be stored as unions (lists) of convex sets (the recent algorithm of [38] is an exception).

As mentioned earlier, sets can be defined using combinations of inequalities and, not surprisingly, linear inequalities play a prominent role in the representation of some of the most popular classes of sets. We will mostly use *convex polytopes*, bounded polyhedra definable as conjunctions of linear inequalities. Let us mention, though, that Boolean combinations of *polynomial* inequalities define the *semi-algebraic* sets, which admit some interesting mathematical and computational results. Their algorithmics is, however, much more complex than that of polyhedral sets. The only class of sets definable by nonlinear inequalities for which relatively-efficient algorithms have been developed is the class of *ellipsoids*, convex sets defined as deformations of a unit circle by a (symmetric and positive definite)

linear transformation [54]. Ellipsoids can be finitely represented by their center and the transformation matrix and like polytopes, they are closed under linear transformations, a fact that facilitates their use in reachability computation for linear systems. Ellipsoids differ from polytopes by not being closed under intersection but such intersections can be approximated to some extent.

4.3 Convex Polytopes

In the following we list some facts concerning convex polytopes. These objects, which underlie other domains such as linear programming, admit a very rich theory of which we only scratch the surface. Readers interested in more details and precision may consult textbooks such as [75, 82].

A *linear inequality* is an inequality of the form $a \cdot x \leq b$ with a being an n-dimensional vector. The set of all points satisfying a linear inequality is called a *halfspace*. Note that the relationship between halfspaces and linear inequalities is not one-to-one because any inequality of the form $ca \cdot x \leq cb$, with c positive, will represent the same set. However using some conventions one can establish a unique representation for each halfspace. A convex polyhedron is an intersection of finitely many halfspaces. A convex polytope is a bounded convex polyhedron. A *convex combination* of a set $\{x_1, \ldots, x_l\}$ of points is any $x = \lambda_1 x_1 + \cdots + \lambda_l x_l$ such that

$$\bigwedge_{i=1}^{l} \lambda_i \geq 0 \wedge \sum_{i=1}^{l} \lambda_i = 1.$$

The *convex hull* of a set $\tilde{P}$ of points, denoted by $P = conv(\tilde{P})$, is the set of all convex combinations of its elements. Convex polytopes admit two types of canonical representations:

1. Vertices: each convex polytope P admits a finite minimal set $\tilde{P}$ such that $P = conv(\tilde{P})$. The elements of $\tilde{P}$ are called the *vertices* of P.
2. Inequalities: a convex polytope P admits a minimal set $H = \{H^1, \ldots, H^k\}$ of halfspaces such that $P = \bigcap_{i=1}^{k} H^i$. This set is represented syntactically as a conjunction of inequalities

$$\bigwedge_{i=1}^{k} a^i \cdot x \leq b^i.$$

Some operations are easier to perform on one representation and some on the other. Testing membership $x \in P$ is easier using inequalities (just evaluation) while using vertices representation, one needs to solve a system of linear equations to find the λ's. To check whether $P_1 \cap P_2 \neq \emptyset$ one can first combine syntactically the inequalities of P_1 and P_2 but in order to check emptiness, these inequalities should be brought into a canonical form. On the other hand, $conv(\tilde{P})$ is always non-empty for any non-empty $\tilde{P}$. Various (worst-case exponential) algorithms convert polytopes from one representation to the other.

The most interesting property of convex polytopes, which is also shared by ellipsoids, is the fact that they are closed under linear operators, that is, for a matrix A, if P is a convex polytope (resp. ellipsoid) so is the set

$$AP = \{Ax : x \in P\}$$

and this property is evidently useful for set-based simulation. The operation can be carried out using both representations of polytopes: if $P = conv(\{x_1, \ldots, x_l\})$ then $AP = conv(\{Ax_1, \ldots, Ax_l\})$ and we leave the computation based on inequalities as an exercise to the reader.

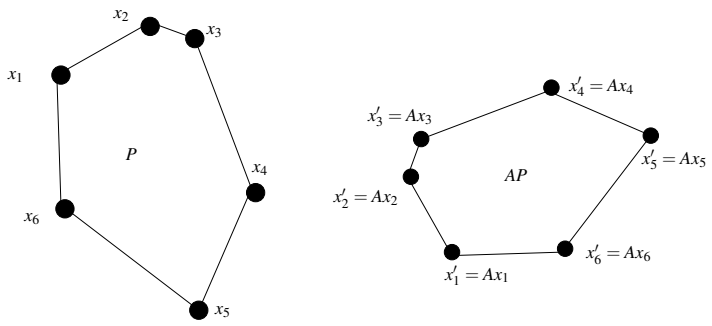

Figure 3: Computing AP from P by applying A to the vertices.

5 Linear Systems

Naturally, the most successful results on reachability computation have been obtained for systems with linear and affine dynamics, that is, systems defined by *linear differential equations*, not to be confused with the simpler "linear" hybrid automata (LHA) where the derivative of x in any state is independent of x. We will start by explaining the treatment of autonomous systems in discrete time, then move to continuous time and then to systems with bounded inputs. In particular we describe some relatively-recent complexity improvements based on a "lazy" representation of the reachable sets. This algorithm underlies the major verification procedure in the **SpaceEx** tool [39] and can handle quite large systems.

5.1 Discrete-Time Autonomous Systems

Consider a system defined by the recurrence equation

$$x_{i+1} = Ax_i.$$

In this case

$$R_{[0..L]}(X_0) = \bigcup_{i=0}^{L} A^i X_0$$

and the abstract algorithm can be realized as follows:

Algorithm 2 (Discrete-Time Linear Reachability)

Input: *A set* $X_0 \subset X$ *represented as* $conv(\tilde{P}_0)$
Output: $Q = R_{[0..L]}(X_0)$ *represented as a list* $\{conv(\tilde{P}_0), \ldots, conv(\tilde{P}_L)\}$

$P := Q := \tilde{P}_0$
repeat $i = 1, 2 \ldots$
  $P := AP$
  $Q := Q \cup P$
until $i = L$

The complexity of the algorithm, assuming $|\tilde{P}_0| = m_0$ is $O(m_0 LM(n))$ where $M(n)$ is the complexity of matrix-vector multiplication in n dimensions which is $O(n^3)$ for simple algorithms and slightly less for

fancier ones. As noted, this algorithm can be applied to other representations of polytopes, to ellipsoids and any other class of sets closed under linear transformations. If the purpose of reachability is to detect intersection with a set B of bad states we can weaken the loop termination condition into $(i = L) \vee (P \cap B \neq \emptyset)$ where the intersection test is done by transforming P into an inequalities representation. If we consider unbounded horizon and want to detect termination we need to check whether the newly-computed P is included in Q which can be done by "sifting" P through all the polytopes in Q and checking whether it goes out empty. This is not a simple operation but can be done. In any case, there is no guarantee that this condition will ever become true.

5.2 Continuous-Time Autonomous Systems

The approach just described can be adapted to *continuous-time* systems of the form

$$\dot{x} = Ax$$

as follows. First it is well known that by choosing a time step r and computing the corresponding matrix exponential $A' = e^{Ar}$ we obtain a discrete-time system

$$x_{i+1} = A'x_i$$

which approximates the original system in the sense that for every i, x_i of the discrete-time system is close to $x[ir]$ of the continuous-time system. The quality of the approximation can be indefinitely improved by taking smaller r. We then use the discrete time reachability operator to compute $P' = R_{\{r\}}(P) = A'P$, that is, the successors of P at time r and can use one out of several techniques to compute an approximation of $R_{[0,r]}(P)$ from P and P':

5.2.1 Make r Small

This approach, used implicitly by [53], just makes r small enough so that subsequent sets overlap each other and the difference between their unions and the continuous-time reachable set vanishes.

5.2.2 Bloating

Let P and P' be represented by the sets of vertices $\tilde{P}$ and $\tilde{P}'$ respectively. The set $\overline{P} = conv(\tilde{P} \cup \tilde{P}')$ is a good approximation of $R_{[0,r]}(P)$ but since in general, we would like to obtain an *over-approximation* (so that if the computed reachable sets does not intersect with the bad set, we are sure that the actual set does not either) we can bloat this set to ensure that it is an *outer* approximation of $R_{[0,r]}(P)$.

This can be done, for example, by pushing the facets of $\overline{P}$ outward by a constant derived from the Taylor approximation of the curve [12]. To this end we need first to transform $\overline{P}$ into an inequalities representation. An alternative approach [24] is to find this over-approximation via an optimization problem. Note that for autonomous systems we can modify Algorithm 1 by replacing the initialization by $P := Q := R_{[0,r]}(X_0)$ and the iteration by $P := R_{[r,r](P)}$, that is, the successors after exactly r time units. This way the over-approximation is done only *once* for $conv(\tilde{P}_0 \cup \tilde{P}_1)$ and then A is applied successively to this set [58].

5.2.3 Adding an Error Term

The last approach that we mention is particularly interesting because it can be used, as we shall see later, also for non-autonomous systems as well as nonlinear ones. Let Y and Y' be two subsets of $\mathbb{R}^n$. Their *Minkowski sum* is defined as

$$Y \oplus Y' = \{y + y' : y \in Y \wedge y' \in Y'\}.$$

The maximal distance between the sets $R_r(P)$ and $R_{[0,r]}(P)$ can be estimated globally. Then, one can fix an "error ball" E (could be a polytope for that matter) of that radius and over-approximate $R_{[0,r]}(P)$ as $AP \oplus E$. Since this computation is equivalent to computing the reachable set of the discrete time system $x_{i+1} = A'x_i + e$ with $e \in E$, we can use the techniques for systems with input described in the next section.

5.3 Discrete-Time Systems with Input

We can now move, at last, to open systems of the form

$$x_{i+1} = Ax_i + Bv_i$$

where v ranges over a bounded convex set V. The one-step successor of a set P is defined as

$$P' = \{Ax + Bv : x \in P, v \in V\} = AP \oplus BV.$$

Unlike linear operations that preserve the number of vertices of a convex polytope, the Minkowski sum increases their number and its successive application may prohibitively increase the representation size. Consequently, methods for reachability under disturbances need some compromise between exact computation that leads to explosion, and approximations which keep the representation size small but may accumulate errors to the point of becoming useless, a phenomenon also known in numerical analysis as the "wrapping effect" [50, 51]. We illustrate this tradeoff using three approaches.

5.3.1 Using Vertices

Assume both P and V are convex polytopes represented by their vertices, $P = conv(\tilde{P})$ and $V = conv(\tilde{V})$. Then it is not hard to see that

$$AP \oplus BV = conv(\{Ax + Bv : x \in \tilde{P}, v \in \tilde{V}\}).$$

Hence, applying the affine transformation to all combinations of vertices in $\tilde{P} \times \tilde{V}$ we obtain all the candidates for vertices of P' (see Figure 4). Of course, not all of these are actual vertices of P' but there is no known efficient procedure to detect which are and which are not. Moreover, it may turn out that the number of actual vertices indeed grows in a super-linear way. Neglecting the elimination of fictitious vertices and keeping all these points as a representation of P' will lead to $|\tilde{P}| \cdot |\tilde{V}|^k$ vertices after k steps, a completely unacceptable situation.

5.3.2 Pushing Facets

This approach over-approximates the reachable set while keeping its complexity more or less fixed. Assume P to be represented in (or converted into) inequality representation. For each supporting halfspace H^i defined by $a^i \cdot x \leq b^i$, let $v^i \in V$ be the disturbance vector which pushes H^i in the "outermost" way, that is, the one which maximizes the product $v \cdot a^i$ with the normal to H^i. In the discrete time setting described

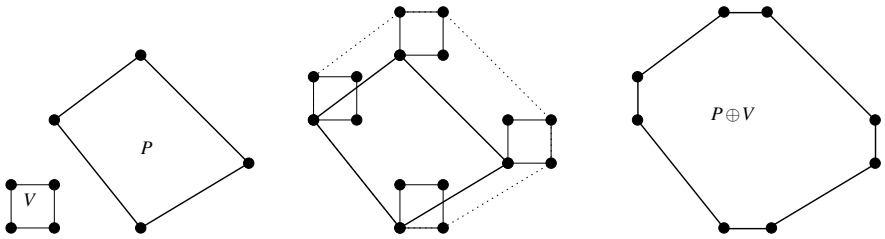

Figure 4: Adding a disturbance polytope V to a polytope P leads to a polytope $P \oplus V$ with more vertices. The phenomenon is more severe in higher dimensions.

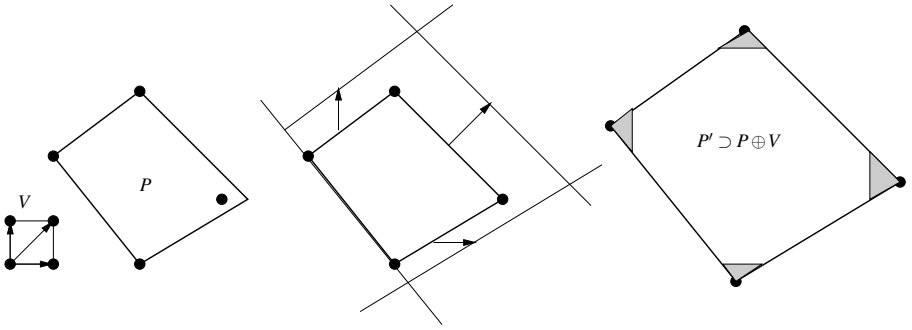

Figure 5: Pushing each face of P by the element of V which pushes it to the maximum. The result will typically not have more facets or vertices but may be a proper superset of $P \oplus V$ (shaded triangles represent the over-approximation error).

here, $v^i \in \tilde{V}$ for every i. We then apply to each H^i the transformation $Ax + Bv^i$ and the intersection of the hyperplanes thus obtained is an over-approximation of the successors (see Figure 5).

This approach has been developed first in the context of continuous time, starting with [81] who applied it to supporting planes of ellipsoids and then adapted in [28] for polytopes. It is also similar in spirit to the face lifting technique of [32]. In continuous time, the procedure of finding each v^i is a linear program derived from the maximum principle of *optimal control*. Recently is has been demonstrated that by using redundant constraints [13] the error can be reduced dramatically for quite large systems.

5.3.3 Lazy Representation

The previous sections showed that until recently, the treatment of linear systems with input provided two alternatives: either apply the linear transformation to an ever-increasing number of objects (vertices, inequalities, zonotope generators) and thus slowing down the computation in each step, or accumulate large over-approximation errors. The following observation due to Colas Le Guernic [57, 41] allows us to benefit from both worlds: it can restrict the application of the linear transformation A to a *fixed* number of points at each step while at the same time *not accumulating* approximation errors. Let us look at two consecutive sets P_k and P_{k+1} in the computation:

$$P_k = A^k P_0 \oplus A^{k-1} BV \oplus A^{k-2} BV \oplus \ldots \oplus BV$$

and

$$P_{k+1} = A^{k+1} P_0 \oplus A^k BV \oplus A^{k-1} BV \oplus \ldots \oplus BV.$$

As one can see, these two sets "share" a lot of common terms that need not be recomputed (and this holds also for continuous-time and variable time-steps). And indeed, the algorithm described in [41] computes the sequence $P_0,\ldots,P_k$ in $O(k(m_0+m)M(n))$ time. From this symbolic/lazy representation of P_i, one can produce an approximating polytope with any desired precision, but this object is *not* used to compute P_{i+1} and hence the wrapping effect is avoided.

A first prototype implementation of that algorithm could compute the reachable set after 1000 steps for linear systems with 200 state variables within 2 minutes. This algorithm was first discovered for zonotopes (a class of centrally symmetric polytopes closed under Minkowski sum proposed for reachability in [40]) but was later adapted to arbitrary convex sets represented by *support functions* [60, 58]. A re-engineered version of the algorithm has been implemented into the **SpaceEx** tool [39]. Although one might get the impression that the reachability problem for linear systems can be considered as solved, the development of **SpaceEx** under the direction of Goran Frehse has confirmed once more that a lot of work is needed in order to transform bright ideas into a working tool that can robustly handle large non-trivial systems occurring in practice.[4]

6 Hybrid and Nonlinear Systems

Being able to handle quite large linear systems, a major challenge is to extend reachability to richer classes of systems admitting hybrid or nonlinear dynamics.

6.1 Hybrid systems

The analysis of hybrid systems was the major motivation for developing reachability algorithms because unlike analytical methods, these algorithms can be easily adapted to handle discrete transitions and mode switching. Figure 6 shows a very simple hybrid automaton with two states, each with its own linear dynamics (using another terminology we have here a piecewise-linear or piecewise-affine dynamical system). An (extended) state of a hybrid system is a pair $(s,x)\in S\times X$ where s is the discrete state (mode). A transitions from state s_i to state s_j may occur when the condition $G_{ij}(x)$ (the transition guard) is satisfied by the current value of x. Such conditions are typically comparisons of state variables with thresholds or more generally linear inequalities. Moreover, while staying at discrete state s, the value of x should satisfy additional constraints, known as *state invariants*.[5] Like timed automata, hybrid automata can exhibit dense non-determinism in parts of the state-space where both a transition guard and a state invariant hold. The runs/trajectories of such an automaton are of the form

$$(s_1,x[0])\xrightarrow{t_1}(s_1,x[t_1])\longrightarrow(s_2,x[t_1])\xrightarrow{t_2}(s_2,x[t_1+t_2])\longrightarrow\cdots,$$

that is, an interleaving of *continuous trajectories* and *discrete transitions* taken at extended states where guards are satisfied. The adaptation of linear reachability computation so as to compute the reachable subset of $S\times X$ follows the procedure proposed already in [5] for simpler dynamics and implemented in the HyTech tool [45]. It goes like this: first, continuous reachability is applied using the dynamics A_1 of s_1, while respecting the state invariant I_1. Then the set of reachable states is intersected with the (semantics of the) transition guard G_{12}. The outcome serves as an initial set of states in s_2 where continuous linear reachability with A_2 and I_2 is applied and so on.

[4] Readers are encouraged to download **SpaceEx** at `http://spaceex.imag.fr/` to obtain a first hand experience in reachability computation.

[5] The full hybrid automaton model may also associate transitions with *reset maps* which are transformations (jumps) applied to x upon a transition.

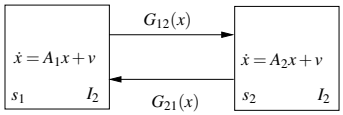

Figure 6: A simple hybrid system with two modes.

The real story is, of course, not that simple for the following reasons.

1. The intersection of the reachable states with the state invariant breaks the symbolic lazy representation as soon as some part of the reachable set leaves the invariant. Likewise, the change of dynamics after the transition invalidates the update scheme of the lazy representation and the set has to be over-approximated before doing intersection and reachability in the next state.
2. The dynamics might be "grazing" the transition guard, intersecting different parts of it at different time steps, thus spawning many subsequent computations. In fact, even a dynamics which proceeds orthogonally toward a single transition guard may spawn many such computations because when small time steps are used, many consecutive sets may have a non-empty intersection with the guard. Consequently, techniques for hybrid reachability such as [59] tend to cluster these sets together before conducting reachability in the next state thus increasing the over-approximation error. This error can now be controlled using the techniques of [38].
3. Even in the absence of these phenomena, when there are several transitions outgoing from a state we may end up with an exponentially growing number of runs to explore.

All these are problems, some tedious and some glorious, that need to be resolved if we want to provide a robustly working tool.

6.2 Nonlinear Systems

Many challenging problems in numerous engineering and scientific domains boil down to exploring the behaviors of nonlinear systems. The techniques described so far take advantage of the intimate relationships between linearity and convexity, in particular the identity $A \cdot conv(\tilde{P}) = conv(A\tilde{P})$. For nonlinear functions such properties do not hold and new ideas are needed. I sketch briefly two approches for adapting reachability for such systems: one which is general and is based on linearizing the system at various parts of the state-space thus obtaining a piecewise-linear system (a hybrid automaton) to which linear reachability techniques are applied. Other techniques look for more sophisticated data-structures and syntactical objects that can represent sets reached by specific classes of nonlinear systems such as those defined by polynomial dynamics. Unlike linear systems, linear reachability is still in an exploratory phase and it is too early to predict which of the techniques described below will survive.

6.2.1 Linearization/Hybridization

Consider a nonlinear system $\dot{x} = f(x)$ and a partition of its state space, for example into cubes (see Figure 7). For each cube s one can compute a linear function A_s and an error polytope V_s such that for every $x \in s$, $f(x) - A_s x \in V_s$ and hence we have a conservative approximation $f(x) \in A_s x \oplus V_s$. We can now build a hybrid automaton (a piecewise-affine dynamical system) whose states correspond to the cubes, and which makes transitions (mode switching) from s to s' whenever x crosses the boundary between them (see Figure 7). The automaton provides an over approximation of the nonlinear system in the sense

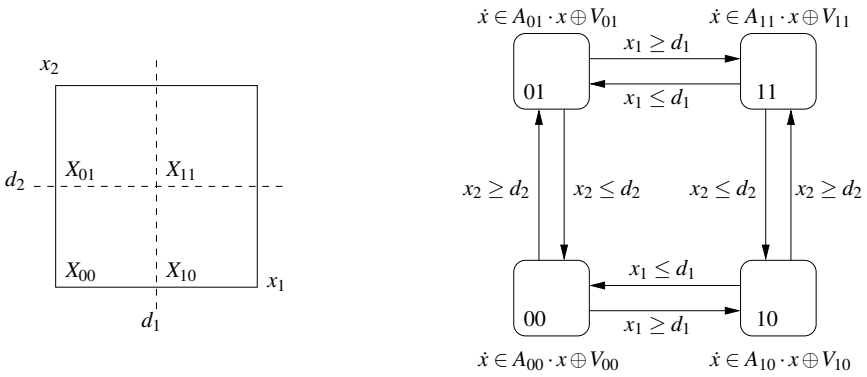

Figure 7: Hybridization: a nonlinear system is over-approximated by a hybrid automaton with an affine dynamics in each state. The transition guards indicate the conditions for switching between neighboring linearizations.

that any run of the latter corresponds to (a projection on X of) a run of the hybrid automaton. This technique, initially developed for doing simulations, has been coined *hybridization* in [11]. An earlier work [46] partitions the state-space similarly but approximates f in each cube by a simpler dynamics of the form $\dot{x} \in C$ for a constant polytope C.

To perform reachability computation on the automaton one can apply the linear techniques described in the preceding section using A_s and V_s, as long as the reachable states remain within cube s. Whenever some P_i reaches the boundary between s and s' we need to intersect it with the switching surface (the transition guard) and use the obtained result as an initial set for reachability computation in s' using $A_{s'}$ and $V_{s'}$. However, the difficulties previously mentioned concerning reachability for hybrid systems, and in particular the fact that the reachable set may leave a cube and penetrate into exponentially many neighboring cubes, renders hybridization impractical beyond 3 dimensions. A dynamic version of hybridization, not based on a fixed grid, has been introduced in [31] and is illustrated in Figure 8. The idea is quite simple: first a linearization domain around the initial set is constructed with the corresponding affine dynamics. Linear reachability computation is performed until the first time the computed polytope leaves the domain. Then the last step is undone, and a new linearization domain is constructed around the current set and linear reachability is resumed. Unlike static hybridization, linearization domains overlap and do not form a partition, but the inclusion of trajectories still hold by construction (see more details in [31]). The intersection operation and the artificial splitting of sets due to the fixed grid are altogether avoided.

The computation of the linear approximation of f and its error bound are costly procedures and should not be done too often. Hence it is important to choose the size and shape of the linearization domains such that the error is small and the computation stays in each domain as long as possible. This topic has been studied in [34] where the curvature of f has been used to construct and orient simplical linearization domains.

6.2.2 Specialized Methods

We mention a few other approaches for reachability computation for nonlinear systems. For polynomial systems, the Bernstein form of polynomials is used to either enclose the image of a set by the polynomial, or to approximate the polynomial by affine bound functions. The different representations using the

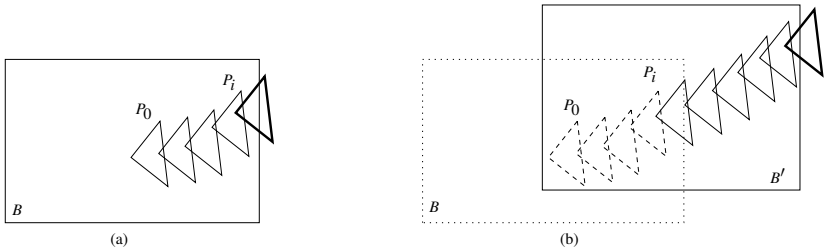

Figure 8: Dynamic hybridization: (a) Computing in some box until intersection with the boundary; (b) Backtracking one step and computing in a new box.

Bernstein form include the Bézier simplices [29] and the Bernstein expansion [35, 77]. A nonlinear function can be over-approximated, based on its Taylor expansion, by a polynomial and an interval which includes all the remainder values. Then, the integration of an ODE system can be done using the Picard operator with reachable sets are represented by boxes [22]. Similarly, in [2] non-linear functions are approximated by linear differential inclusions with dynamical error estimated as the remainder of the Taylor expansion around the current set. As a set representation, "polynomial zonotopes" based on a polynomial rather than linear combination of generators are used.

We may conclude that the extension of reachability computation to nonlinear and hybrid systems is a challenging problem which is still waiting for several conceptual and algorithmic breakthroughs. We believe that the ability to perform reachability computation for nonlinear systems of non-trivial size can be very useful not only for control systems but also for other application domains such as *analog circuits* and *systems biology*. In biological models, uncertainty in parameters and environmental condition is a rule, not an exception, and set-based simulation, combined with other approaches for exploring the uncertainty space of under-determined dynamical systems can be very beneficial.

7 Related and Future Work

The idea of set-based numerical integration has several origins. In some sense it can be seen as related to those parts of numerical analysis, for example *interval analysis* [72, 47], that provide "robust" set-based results for numerical computation to compensate for numerical errors. This motivation is slightly different from verification and control where the uncertainty is attributed to an external environment not to the internal computation. Set-based computations also underlie the *abstract interpretation* framework [27] for static analysis of software.

The idea of applying set-based computation to hybrid systems was among the first contributions of the verification community to hybrid systems research [5] and it has been implemented in the pioneering HyTech tool [45]. However this idea was restricted to hybrid automata with very simple continuous dynamics (a constant derivative in each state) where trajectories can be computed without numerical integration. To the best of our knowledge, the first explicit mention of combining numerical integration with approximate representation by polyhedra in the context of verification appeared in [43]. It was recently brought to my attention[6] that an independent thread of reachability computation, quite similar in motivation and techniques, has developed in the USSR starting from the early 70s [62]. More citations from this school can be found in [63].

[6] G. Frehse, personal communication.

The polytope-based techniques described here were developed independently in [24, 23, 25] and in [12, 28]. Among other similar techniques that we have not described in detail, let us mention again the extensive work on ellipsoids [54, 53, 19, 55] and another family of methods [71, 79] which uses techniques such as *level sets*, inspired by numerical solution of partial differential equations, to compute and represent the reachable states. Among the symbolic (non numerical) approaches to the problem let us mention [56] which computes an effective representation of the reachable states for a restricted class of linear systems. Attempts to scale-up reachability techniques to higher dimensions using compositional methods that analyze an abstract approximate systems obtained by projections on subsets of the state variables are described in [44] and [10].

The interpretation of V as the controller's output rather than disturbance transforms the reachability problem into some open-loop variant of controller synthesis [67, 68]. Hence it is natural that the optimization-based approach developed in [17] for synthesis, has also been applied to reachability computation [18]. On the other hand, reachability computation can be used to synthesize controllers in the spirit of dynamic programming, as has been demonstrated in [9] where a backward reachability operator has been used as part of an algorithm for synthesizing switching controllers.

Another class of methods, called *simulation-based*, for example [49, 42, 33, 37, 36, 1], attempts to obtain the same effect as reachability computation by finitely many simulations, not necessarily of extremal points as in the methods described in this paper. Such techniques may turn out to be superior for nonlinear systems whose dynamics does not preserve convexity and systems whose dynamics is expressed by programs that do not always admit a clean mathematical model.

Alternative approaches to verify continuous and hybrid systems algorithmically attempt to approximate the system by a simpler one, typically a finite-state automaton[7] [52, 7, 78]. This can be done by simply partitioning the state space into cubes and defining transitions between adjacent cubes which are connected by trajectories, or by more modern methods, inspired by program verification, such as *predicate abstraction* and *counter-example based refinement* [6, 26, 73]. It should be noted, however, that finite-state models based on space partitions suffer from the problem of *false transitivity*: the abstract system may have a run of the form $s_1 \to s_2 \to s_3$ while the concrete one has no trajectory $x_1 \to x_2 \to x_3$ passing through these regions. As a result, the finite-state model will often have too many spurious behaviors to be useful for verification.

Finally let me mention some other issues not discussed so far:

- Disturbance models: implicit in reachability computation is the assumption that the only restriction on the input signal is that it always remains in V. This means that it may oscillate in any frequency between the extremal values of V. Such a non realistic assumption may increase the reachable set and render the analysis too pessimistic. This effect can be reduced by composing the system with a bounded-variability non-deterministic model of the input generator but this will increase the size of the system. In fact, the technique of abstraction by projection [10] does the opposite: it projects away state-variables and converts them into bounded input variables. A more systematic study of precision/complexity tradeoffs could be useful here.
- Temporal properties: in our presentation we assumed implicitly that systems specifications are simply *invariance* properties, a subclass of safety properties which are violated once a trajectory reaches a forbidden state. It is, of course, possible to follow the usual procedure of taking a more complex temporal property, constructing its automaton and composing with the system model. This procedure will extend the discrete state-space of the system and will make the analysis harder by virtue of having more discrete transitions with the usual additional complications associated

[7] In fact, hybridization is another instance of this approach.

with detection of cycles in the reachability graph and extraction of concrete trajectories that realize them.

- Adaptive algorithms: although we attempt to be as general as possible, it should be admitted that different systems lead to different behaviors of the reachability algorithm, even for linear systems. If we opt for general techniques that do not require a dedicated super-intelligent user, we need to make the algorithms more adaptive to their own behavior and automatically explore different values of their parameters such as time steps, size and shapes of approximating polytopes, quick and approximate inclusion tests, different search strategies (for hybrid models) and more.
- Numerical aspects: the discourse in this paper treated real-valued computations as well-functioning black boxes. In practice, certain systems can be more problematic in terms of numerical stability or operate in several time scales and this should be taken into account.
- Differential-algebraic equations: many dynamical systems that model physical phenomena obeying conservation laws are modeled using differential-algebraic equations with relational constraints over variables and derivatives. In addition to the existing simulation technology for such systems, a specialized reachability theory should be developed, see for example [3].
- Discrete state explosion: the methods described here focus on scalability with respect to the dimensionality of the continuous state-space and tacitly assume that the number of discrete modes is not too large. This assumption can be wrong in at least two contexts: when discrete states are used to encode different parts of a high dimensional state-space as in hybridization or in the verification of complex control software when there is a relatively small number of continuous variables used to model the environment of the software which has many states. The problem of combining the symbolic representations for discrete and continuous spaces, for example BDDs and polytopes, is an unsolved problem already for the simplest case of timed automata.
- Finally, in terms of usability, the whole set-based point of view is currently not the natural one for practitioners (this is true also for verification versus testing in the discrete case) and the extraction of individual trajectories that violate the requirements as well as input signals that induce them will make reachability more acceptable as a powerful model debugging method.

Acknowledgments: This work benefitted from discussions with George Pappas, Bruce Krogh, Charles Rockland, Eugene Asarin, Thao Dang, Goran Frehse, Anoine Girard, Alexandre Donzé and Colas Le Guernic.

References

[1] Houssam Abbas, Georgios Fainekos, Sriram Sankaranarayanan, Franjo Ivančić & Aarti Gupta (2013): *Probabilistic temporal logic falsification of cyber-physical systems*. *ACM Transactions on Embedded Computing Systems (TECS)* 12(2s), p. 95, doi:10.1145/2465787.2465797.

[2] Matthias Althoff (2013): *Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets*. In: *HSCC*, ACM, pp. 173–182, doi:10.1145/2461328.2461358.

[3] Matthias Althoff & Bruce Krogh (2013): *Reachability Analysis of Nonlinear Differential-Algebraic Systems*. *Automatic Control, IEEE Transactions on* PP(99), doi:10.1109/TAC.2013.2285751.

[4] Rajeev Alur (2011): *Formal verification of hybrid systems*. In: *EMSOFT*, pp. 273–278, doi:10.1145/2038642.2038685.

[5] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis & Sergio Yovine (1995): *The Algorithmic Analysis of Hybrid Systems*. *Theoretical Computer Science* 138(1), pp. 3–34. Available at http://dx.doi.org/10.1016/0304-3975(94)00202-T.

[6] Rajeev Alur, Thao Dang & Franjo Ivancic (2006): *Counterexample-guided predicate abstraction of hybrid systems*. *Theoretical Computer Science* 354(2), pp. 250–271. Available at http://dx.doi.org/10.1016/j.tcs.2005.11.026.

[7] Rajeev Alur, Thomas A Henzinger, Gerardo Lafferriere & George J Pappas (2000): *Discrete abstractions of hybrid systems*. *Proceedings of the IEEE* 88(7), pp. 971–984, doi:10.1109/5.871304.

[8] E Asarin, O Maler, A Pnueli & J Sifakis (1998): *Controller synthesis for timed automata*. In: *Proc. System Structure and Control*, Elsevier. Available at http://www-verimag.imag.fr/PEOPLE/Oded.Maler/Papers/newsynth.pdf.

[9] Eugene Asarin, Olivier Bournez, Thao Dang, Oded Maler & Amir Pnueli (2000): *Effective synthesis of switching controllers for linear systems*. *Proceedings of the IEEE* 88(7), pp. 1011–1025, doi:10.1109/5.871306. Available at http://www-verimag.imag.fr/~maler/Papers/procieee.pdf.

[10] Eugene Asarin & Thao Dang (2004): *Abstraction by Projection and Application to Multi-affine Systems*. In: *HSCC*, *LNCS* 2993, Springer, pp. 32–47, doi:10.1007/978-3-540-24743-2_3. Available at http://springerlink.metapress.com/openurl.asp?genre=article{&}issn=0302-9743{&}volume=2993{&}spage=32.

[11] Eugene Asarin, Thao Dang & Antoine Girard (2007): *Hybridization methods for the analysis of nonlinear systems*. *Acta Informatica* 43(7), pp. 451–476. Available at http://dx.doi.org/10.1007/s00236-006-0035-7.

[12] Eugene Asarin, Thao Dang, Oded Maler & Olivier Bournez (2000): *Approximate Reachability Analysis of Piecewise-Linear Dynamical Systems*. In: *HSCC*, *LNCS* 1790, Springer, pp. 20–31, doi:10.1007/3-540-46430-1_6. Available at http://link.springer.de/link/service/series/0558/bibs/1790/17900020.htm.

[13] Eugene Asarin, Thao Dang, Oded Maler & Romain Testylier (2010): *Using Redundant Constraints for Refinement*. In: *ATVA*, pp. 37–51, doi:10.1007/978-3-642-15643-4_5. Available at http://www-verimag.imag.fr/~maler/Papers/redundant.pdf.

[14] Eugene Asarin, Oded Maler & Amir Pnueli (1995): *Symbolic Controller Synthesis for Discrete and Timed Systems*. In: *Hybrid Systems II*, 999, Springer, pp. 1–20, doi:10.1007/3-540-60472-3_1. Available at http://www-verimag.imag.fr/~maler/Papers/symbolic.pdf.

[15] Karl J Astrom (1970): *Introduction to stochastic control theory*. Academic Press.

[16] Jean-Pierre Aubin & Arrigo Cellina (1984): *Differential inclusions : set-valued maps and viability theory*. *Grundlehren der mathematischen Wissenschaften* 264, Springer-Verlag, doi:10.1007/978-3-642-69512-4.

[17] Alberto Bemporad & Manfred Morari (1999): *Control of systems integrating logic, dynamics, and constraints*. *Automatica* 35(3), pp. 407–427, doi:10.1016/S0005-1098(98)00178-2.

[18] Alberto Bemporad, Fabio Danilo Torrisi & Manfred Morari (2000): *Optimization-based verification and stability characterization of piecewise affine and hybrid systems*. In: *Hybrid Systems: Computation and Control*, Springer, pp. 45–58, doi:10.1007/3-540-46430-1_8.

[19] Oleg Botchkarev & Stavros Tripakis (2000): *Verification of Hybrid Systems with Linear Differential Inclusions Using Ellipsoidal Approximations*. In: *HSCC*, *LNCS* 1790, Springer, pp. 73–88, doi:10.1007/3-540-46430-1_10. Available at http://link.springer.de/link/service/series/0558/bibs/1790/17900073.htm.

[20] Christos G Cassandras & John Lygeros (2010): *Stochastic hybrid systems*. CRC Press.

[21] Franck Cassez, Alexandre David, Emmanuel Fleury, Kim G Larsen & Didier Lime (2005): *Efficient on-the-fly algorithms for the analysis of timed games*. In: *CONCUR*, Springer, pp. 66–80, doi:10.1007/978-3-540-75454-1_3.

[22] Xin Chen, Erika Ábrahám & Sriram Sankaranarayanan (2012): *Taylor model flowpipe construction for non-linear hybrid systems*. In: *Proc. RTSS12*, IEEE, pp. 183–192.

[23] Alongkrit Chutinan (1999): *Hybrid System Verification using Discrete Model Approximations*. Ph.D. thesis, Carnegie Mellon University.

[24] Alongkrit Chutinan & Bruce H Krogh (1999): *Verification of Polyhedral-Invariant Hybrid Automata Using Polygonal Flow Pipe Approximations*. In: *HSCC*, *LNCS* 1569, Springer, pp. 76–90, doi:10.1007/3-540-48983-5_10. Available at `http://link.springer.de/link/service/series/0558/bibs/1569/15690076.htm`.

[25] Alongkrit Chutinan & Bruce H Krogh (2003): *Computational techniques for hybrid system verification*. *IEEE Transactions on Automatic Control* 48(1), pp. 64 – 75. Available at `http://dx.doi.org/10.1109/TAC.2002.806655`.

[26] Edmund Clarke, Ansgar Fehnker, Zhi Han, Bruce Krogh, Joël Ouaknine, Olaf Stursberg & Michael Theobald (2003): *Abstraction and counterexample-guided refinement in model checking of hybrid systems*. *International Journal of Foundations of Computer Science* 14(04), pp. 583–604, doi:10.1142/S012905410300190X.

[27] Patrick Cousot & Radhia Cousot (1977): *Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints*. In: *POPL*, ACM, pp. 238–252, doi:10.1145/512950.512973.

[28] Thao Dang (2000): *Verification and Synthesis of Hybrid Systems*. Ph.D. thesis, Institut National Polytecnique de Grenoble.

[29] Thao Dang (2006): *Approximate Reachability Computation for Polynomial Systems*. In: *HSCC*, Springer, pp. 138–152, doi:10.1007/11730637_13.

[30] Thao Dang, Goran Frehse, Antoine Girard & Colas Le Guernic (2009): *Tools for the Analysis of Hybrid Models*. In: *Communicating Embedded Systems: Software and Design: Formal Methods*, John Wiley & Sons, Inc., pp. 227–251, doi:10.1002/9781118558188.ch7.

[31] Thao Dang, Colas Le Guernic & Oded Maler (2011): *Computing reachable states for nonlinear biological models*. *Theoretical Computer Science* 412(21), pp. 2095–2107, doi:10.1016/j.tcs.2011.01.014. Available at `http://www-verimag.imag.fr/~maler/Papers/nonlinear-bio-tcs.pdf`.

[32] Thao Dang & Oded Maler (1998): *Reachability Analysis via Face Lifting*. In: *HSCC*, *LNCS* 1386, Springer, pp. 96–109, doi:10.1007/3-540-64358-3_34. Available at `http://www-verimag.imag.fr/PEOPLE/Oded.Maler/Papers/facelift.pdf`.

[33] Thao Dang & Tarik Nahhal (2009): *Coverage-guided test generation for continuous and hybrid systems*. *Formal Methods in System Design* 34(2), pp. 183–213, doi:10.1007/s10703-009-0066-0.

[34] Thao Dang & Romain Testylier (2011): *Hybridization domain construction using curvature estimation*. In: *HSCC*, pp. 123–132, doi:10.1145/1967701.1967721.

[35] Thao Dang & Romain Testylier (2012): *Reachability analysis for polynomial dynamical systems using the Bernstein expansion*. *Reliable Computing* 17(2), pp. 128–152.

[36] Alexandre Donzé (2010): *Breach, a toolbox for verification and parameter synthesis of hybrid systems*. In: *CAV*, Springer, pp. 167–170, doi:10.1007/978-3-642-14295-6_17.

[37] Alexandre Donzé & Oded Maler (2007): *Systematic Simulation Using Sensitivity Analysis*. In: *HSCC*, *LNCS* 4416, Springer, pp. 174–189. Available at `http://dx.doi.org/10.1007/978-3-540-71493-4_16`.

[38] Goran Frehse, Rajat Kateja & Colas Le Guernic (2013): *Flowpipe approximation and clustering in space-time*. In: *HSCC*, pp. 203–212, doi:10.1145/2461328.2461361.

[39] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang & Oded Maler (2011): *SpaceEx: Scalable verification of hybrid systems*. In: *Computer Aided Verification*, pp. 379–395, doi:10.1007/978-3-642-22110-1_30. Available at http://www-verimag.imag.fr/~maler/Papers/spaceex-cav.pdf.

[40] Antoine Girard (2005): *Reachability of Uncertain Linear Systems Using Zonotopes*. In: *HSCC*, *LNCS* 3414, Springer, pp. 291–305, doi:10.1007/978-3-540-31954-2_19. Available at http://springerlink.metapress.com/openurl.asp?genre=article{&}issn=0302-9743{&}volume=3414{&}spage=291.

[41] Antoine Girard, Colas Le Guernic & Oded Maler (2006): *Efficient Computation of Reachable Sets of Linear Time-Invariant Systems with Inputs*. In: *HSCC*, *LNCS* 3927, Springer, pp. 257–271. Available at http://dx.doi.org/10.1007/11730637_21.

[42] Antoine Girard & George Pappas (2006): *Verification using simulation*. In: *Hybrid Systems: Computation and Control*, Springer, pp. 272–286, doi:10.1007/11730637_22.

[43] Mark R. Greenstreet (1996): *Verifying Safety Properties of Differential Equations*. In: *CAV*, *LNCS* 1102, Springer, pp. 277–287. Available at http://dx.doi.org/10.1007/3-540-61474-5_76.

[44] Mark R. Greenstreet & Ian Mitchell (1999): *Reachability Analysis Using Polygonal Projections*. In: *Hybrid Systems: Computation and Control*, *LNCS* 1569, Springer, pp. 103–116, doi:10.1007/3-540-48983-5_12. Available at http://link.springer.de/link/service/series/0558/bibs/1569/15690103.htm.

[45] Thomas A Henzinger, Pei-Hsin Ho & Howard Wong-Toi (1997): *HyTech: A model checker for hybrid systems*. In: *Computer aided verification*, Springer, pp. 460–463, doi:10.1007/3-540-63166-6_48.

[46] Thomas A Henzinger, Pei-Hsin Ho & Howard Wong-Toi (1998): *Algorithmic analysis of nonlinear hybrid systems*. *Automatic Control, IEEE Transactions on* 43(4), pp. 540–554, doi:10.1109/9.664156.

[47] Luc Jaulin, Michel Kieffer, Oliver Didrit & Éric Walter (2001): *Applied Interval Analysis*. Springer-Verlag, doi:10.1007/978-1-4471-0249-6.

[48] Mikael Johansson (2002): *Piecewise linear control systems*. Springer Verlag.

[49] Jim Kapinski, Bruce H Krogh, Oded Maler & Olaf Stursberg (2003): *On systematic simulation of open continuous systems*. In: *HSCC*, Springer, pp. 283–297, doi:10.1007/3-540-36580-X_22. Available at http://www-verimag.imag.fr/~maler/Papers/simulation.pdf.

[50] Wolfgang Kühn (1998): *Rigorously computed orbits of dynamical systems without the wrapping effect*. *Computing* 61(1), pp. 47–67, doi:10.1007/BF02684450.

[51] Wolfgang Kühn (1999): *Towards an optimal control of the wrapping effect*. In: *Developments in Reliable Computing*, Springer, pp. 43–51, doi:10.1007/978-94-017-1247-7_4.

[52] Robert P Kurshan & Kenneth L McMillan (1991): *Analysis of digital circuits through symbolic reduction*. *IEEE Trans. on CAD of Integrated Circuits and Systems* 10(11), pp. 1356–1371. Available at http://doi.ieeecomputersociety.org/10.1109/43.97615.

[53] Alexander B. Kurzhanski & Pravin Varaiya (2000): *Ellipsoidal Techniques for Reachability Analysis*. In: *HSCC*, *LNCS* 1790, Springer, pp. 202–214, doi:10.1007/3-540-46430-1_19. Available at http://link.springer.de/link/service/series/0558/bibs/1790/17900202.htm.

[54] Alexandr B Kurzhanski & István Vályi (1997): *Ellipsoidal Calculus for Estimation and Control*. Birkhauser, doi:10.1007/978-1-4612-0277-6.

[55] Alex A Kurzhanskiy & Pravin Varaiya (2007): *Ellipsoidal Techniques for Reachability Analysis of Discrete-Time Linear Systems*. *IEEE Transactions on Automatic Control* 52(1), pp. 26 –38. Available at http://dx.doi.org/10.1109/TAC.2006.887900.

[56] Gerardo Lafferriere, George J Pappas & Sergio Yovine (1999): *A new class of decidable hybrid systems*. In: *Hybrid Systems: Computation and Control*, Springer Berlin Heidelberg, pp. 137–151, doi:10.1007/3-540-48983-5_15.

[57] Colas Le Guernic (2005): *Calcul Efficace de l'Ensemble Atteignable des Systèmes Linéaires avec Incertitudes*. Master's thesis, Université Paris 7. Available at http://www.mpri.master.univ-paris7.fr/attached-documents/Stages-2005-rapports/rapport-2005-LeGuernic.pdf.

[58] Colas Le Guernic (2009): *Reachability Analysis of Hybrid Systems with Linear Continuous Dynamics*. Ph.D. thesis, Université Grenoble 1 – Joseph Fourier. Available at http://tel.archives-ouvertes.fr/docs/00/43/07/40/PDF/CLeGuernic_thesis.pdf.

[59] Colas Le Guernic & Antoine Girard (2009): *Reachability Analysis of Hybrid Systems Using Support Functions*. In: *CAV*, pp. 540–554. Available at http://dx.doi.org/10.1007/978-3-642-02658-4_40.

[60] Colas Le Guernic & Antoine Girard (2010): *Reachability analysis of linear systems using support functions*. *Nonlinear Analysis: Hybrid Systems* 4(2), pp. 250–262, doi:10.1016/j.nahs.2009.03.002.

[61] Daniel Liberzon (2003): *Switching in systems and control*. Springer, doi:10.1007/978-1-4612-0017-8.

[62] A V Lotov (1971): *Construction of domains of attainability for a linear discrete system with bottle-neck constraints*. *Aerophysics and Applied Mathematics*, pp. 113–119. In Russian.

[63] Alexander V Lotov, Vladimir A Bushenkov & Georgy K Kamenev (2004): *Interactive decision maps: Approximation and visualization of Pareto frontier*. 89, Springer, doi:10.1007/978-1-4419-8851-5.

[64] John Lygeros, Shankar Sastry & Claire Tomlin (2001): *The art of hybrid systems*. Available at robotics.eecs.berkeley.edu/~sastry/ee291e/book.pdf. Unpublished manuscript.

[65] Oded Maler (1998): *A unified approach for studying discrete and continuous dynamical systems*. In: *CDC*, 2, pp. 2083–2088, doi:10.1109/CDC.1998.758641. Available at http://www-verimag.imag.fr/~maler/Papers/unified.pdf.

[66] Oded Maler (2001): *Guest Editorial: Verification of Hybrid Systems*. *European Journal of Control* 7(1), pp. 357–365, doi:10.3166/ejc.7.357-365. Available at http://www-verimag.imag.fr/~maler/Papers/guest.pdf.

[67] Oded Maler (2002): *Control from Computer Science*. *Annual Reviews in Control* 26(2), pp. 175–187, doi:10.1016/S1367-5788(02)00030-5. Available at http://www.sciencedirect.com/science/article/B6V0H-485P0W5-3/2/b5160ff386c03f13f06db257df3547e0.

[68] Oded Maler (2007): *On optimal and reasonable control in the presence of adversaries*. *Annual Reviews in Control* 31(1), pp. 1–15, doi:10.1016/j.arcontrol.2007.02.001. Available at http://www-verimag.imag.fr/~maler/Papers/annual.pdf.

[69] Oded Maler (2010): *Amir Pnueli and the dawn of hybrid systems*. In: *HSCC*, pp. 293–295, doi:10.1145/1755952.1755953. Available at http://www-verimag.imag.fr/~maler/Papers/amir-cpsweek.pdf.

[70] Oded Maler (2011): *On under-determined dynamical systems*. In: *EMSOFT*, pp. 89–96. Available at http://www-verimag.imag.fr/~maler/Papers/under-det.pdf.

[71] Ian Mitchell & Claire Tomlin (2000): *Level Set Methods for Computation in Hybrid Systems*. In: *HSCC*, *LNCS* 1790, Springer, pp. 310–323, doi:10.1007/3-540-46430-1_27. Available at http://link.springer.de/link/service/series/0558/bibs/1790/17900310.htm.

[72] Ramon E Moore (1979): *Methods and applications of interval analysis*. SIAM, doi:10.1137/1.9781611970906.

[73] Stefan Ratschan & Zhikun She (2005): *Safety verification of hybrid systems by constraint propagation based abstraction refinement*. In: *HSCC*, Springer, pp. 573–589, doi:10.1145/1210268.1210276.

[74] Abraham J van der Schaft & Johannes M Schumacher (2000): *An introduction to hybrid dynamical systems*. 251, Springer London.

[75] Alexander Schrijver (1986): *Theory of Linear and Integer Programming*. Wiley.

[76] Paulo Tabuada (2009): *Verification and control of hybrid systems: a symbolic approach*. Springer, doi:10.1007/978-1-4419-0224-5.

[77] Romain Testylier & Thao Dang (2013): *NLTOOLBOX: A Library for Reachability Computation of Nonlinear Dynamical Systems*. In: *ATVA*, pp. 469–473, doi:10.1007/978-3-319-02444-8_37.

[78] Ashish Tiwari (2008): *Abstractions for hybrid systems*. *Formal Methods in System Design* 32(1), pp. 57–83, doi:10.1007/s10703-007-0044-3.

[79] Claire J Tomlin, Ian Mitchell, Alexandre M Bayen & Meeko Oishi (2003): *Computational techniques for the verification of hybrid systems*. *Proceedings of the IEEE* 91(7), pp. 986–1001, doi:10.1109/JPROC.2003.814621.

[80] Stavros Tripakis & Thao Dang (2009): *Modeling, verification and testing using timed and hybrid automata*. In: *Model-Based Design for Embedded Systems*, CRC Press, pp. 383–436, doi:10.1201/9781420067859-c13.

[81] Pravin Varaiya (1998): *Reach set computation using optimal control*. In: *Proc. KIT Workshop on Verification of Hybrid Systems*, Verimag, Grenoble, pp. 377–383, doi:10.1007/978-3-642-59615-5_15.

[82] Günter M. Ziegler (1995): *Lectures on Polytopes*. *Graduate Texts in Mathematics* 152, Springer, doi:10.1007/978-1-4613-8431-1.