Reachability Analysis for Cyber-Physical Systems: Are we there yet?

Xin Chen[†] and Sriram Sankaranarayanan [*]

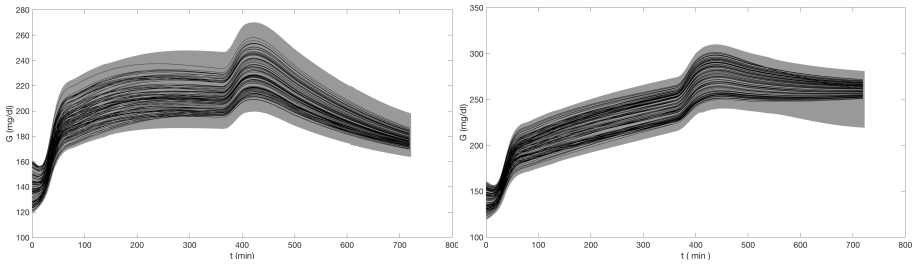
(†) University of Dayton, USA and (∗) University of Colorado Boulder, USA.

Abstract. Reachability analysis is a fundamental problem in verification that checks for a given model and set of initial states if the system will reach a given set of unsafe states. Its importance lies in the ability to exhaustively explore the behaviors of a model over a finite or infinite time horizon. The problem of reachability analysis for Cyber-Physical Systems (CPS) is especially challenging because it involves reasoning about the continuous states of the system as well as its switching behavior. Each of these two aspects can by itself cause the reachability analysis problem to be undecidable. In this paper, we survey recent progress in this field beginning with the success of hybrid systems with affine dynamics. We then examine the current state-of-the-art for CPS with nonlinear dynamics and those driven by "learning-enabled" components such as neural networks. We conclude with an examination of some promising directions and open challenges.

1 Introduction

Formal verification techniques attempt to exhaustively explore the behaviors of computational models that include *finite state machines* that model sequential circuits and network protocols; *push-down machines* that model function calls/returns in software; *Petri-net models* of concurrent systems or *timed automata* that model the execution of real-time systems. In each of the instances above, the reachability problem asks given a model, an initial set of configurations and a target unsafe set, whether the system starting at some initial state can reach an "unsafe" state in some finite number of steps. A reachability analyzer will either provide a proof that the unsafe set is not reachable or a *witness* execution that shows how to reach an unsafe state starting from an initial state.

Reachability analysis has been a powerful tool for checking properties of hardware circuits and software programs with success stories arising from their ability to discover bugs in these systems or prove their absence through exhaustive verification [42,69,20,117,40,75,27,26,48,68]. Since the early 90s, the formal methods and control theory communities have investigated so-called "hybrid" or "Cyber-Physical Systems" (CPS), that model computation interacting closely with a physical environment. Such systems have been mathematically captured by formalisms such as hybrid automata, that combine the evolution of continuous states through ordinary differential equations (ODEs) with discrete mode switches modeled using finite state automata. CPS include systems from a variety of safety-critical areas such as medical devices, control systems that help fly airplanes, power systems and autonomous vehicles. Modeling these systems and reasoning about the set of all reachable states can go a long way towards guaranteeing safe operation during deployment.



(a) (b)

Fig. 1. Reachable sets (in gray) showing the possible blood glucose levels of a patient controlled by two different instantiations of an automated insulin infusion algorithm taken from Chen et al [34]. Simulation trajectories are shown in black. The analysis proves for instance (a) that the blood glucose levels remain below 260 mg/dl over a 24 hour period, whereas for instance (b) it is unable to establish that bound.

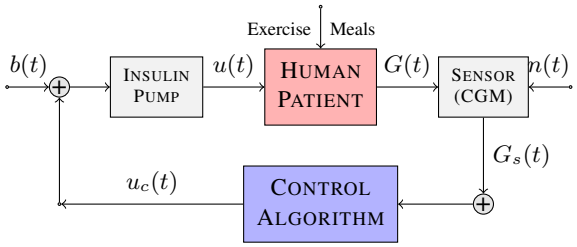


Fig. 2. Block Diagram of an Insulin Infusion Control System.

Consider the block diagram of an insulin infusion control system for patients with type-1 diabetes taken from our previous work [34]. Here, $b(t)$ represents external user commanded insulin, $u(t)$: the insulin infused to patient, $G(t)$, blood glucose level of the patient, $n(t)$: sensor measurement error (noise), $G_s(t)$: glucose level estimated/reported by sensor, and $u_c(t)$: insulin infusion commanded by the algorithm. The patient's blood glucose level is modeled using nonlinear human insulin-glucose model coupled with a controller that switches between various levels of insulin, based on the sensed blood glucose level of the patient. The reachable set estimates computed using the tool Flow* [33] establishes bounds the possible blood glucose levels over a 24 hour time period. Such a flowpipe can be used to establish upper and lower bounds on the value of the blood glucose levels as shown in figure 1. Further details are available from our ARCH 2017 paper [34].

In this paper, we present a brief overview of reachability analysis for Cyber-Physical Systems. We begin by formulating the problem in a formal manner and discuss cases when the problem is known to be decidable along with a brief mention of the broad class of approaches taken to solve the reachability problem. We focus on set-based techniques for systems with linear dynamics wherein powerful tools such as SpaceEx [56] and Hylaa [18] have pushed the state of the art to large hybrid systems with thousands of state variables. We then present some of the approaches for nonlinear systems, while illustrating why the problem is much more challenging when the dynamics are nonlinear. We discuss emerging areas of interest, including reachability analysis for neural

networks. This paper is not meant to be an exhaustive survey of results in this area. A recent survey by Althoff et al is recommended for the reader who wishes to learn more about set-based techniques [4]. The main purposes of this article are to (a) illustrate why the problem is important but challenging; (b) highlight some important approaches to the problem; and (c) highlight a few emerging areas where efficient and precise reachability analysis techniques will play an important role.

2 Hybrid Systems and Reachability Analysis

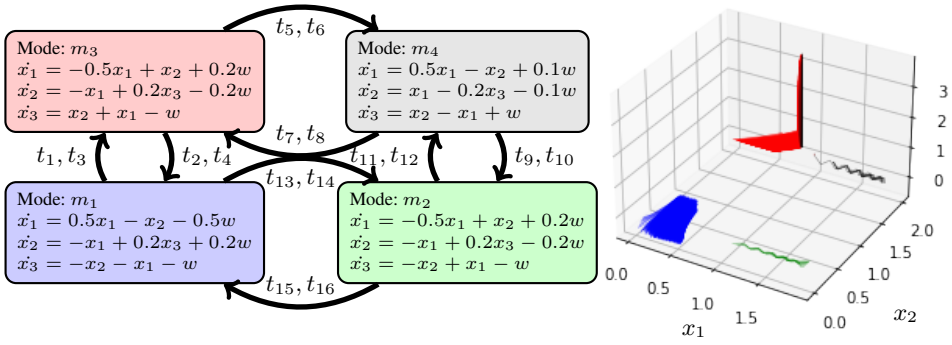
In this section, we will briefly review some of the fundamental concepts that include (a) models of hybrid (Cyber-Physical) systems; (b) the reachability analysis problem; (c) decidable cases for the problem and (d) a brief overview of existing approaches.

A Brief History: The formal study of hybrid (Cyber-Physical) systems was initiated in the early 1990s from the computer science and the control communities. In the controls community, the consideration of hybrid control systems began in the late 1980s as an attempt to formalize supervisory control wherein discrete-event systems are used to represent "higher level" decision making which may switch between multiple "lower level" control strategies to interact with a continuous plant [13]. Early modeling efforts for such systems include the work of Peleties and DeCarlo [97], Gollu and Varaiya [64] and Benveniste and Le Guernic [22]. In the computer science community, the problem of modeling and reasoning about reactive systems naturally led to the consideration of timed systems followed by hybrid systems [88]. The timed-automaton model of Alur and Dill augments automata with finitely many clocks that can trigger transitions between states which may in turn reset these clocks [10,11]. Hybrid systems can then be modeled augmenting these further with physical quantities that evolve according to simple differential equations [89,8,96].

The hybrid automaton model was proposed in order to unify the continuous evolution of state variables with switching due to mode changes within a single formalism. Detailed descriptions are available elsewhere [7,84,114].

Example 1. Figure 3 illustrates a hybrid automaton with four modes $\{m_1, \ldots, m_4\}$, continuous state variables $\{x_1, x_2, x_3\}$ and an external time-varying disturbance input w lying in the range $[-0.25, 0.25]$. The dynamics inside each mode and the transitions between modes are also shown. The transitions are defined by guards and reset maps, as shown in the figure. The figure also shows 5000 trajectories with randomly sampled initial conditions starting from mode m_1 and $x_1 \in [0.3, 0.5], x_2 \in [0.2, 0.4], x_3 \in [0, 0.4]$ with the disturbance in the range $[-0.25, 0.25]$. Each mode is shown in a different color. We note that only 6 out of the 5000 trajectories reach mode m_2 (green).

The example above shows the need for exhaustive simulations, since "corner case behaviors" that violate safety properties are often a concern. We have encountered more realistic systems wherein nearly 100 million random simulations do not expose a safety violation that can be discovered quite easily by a more exhaustive approach [121].



IDs	Guard	Reset	IDs	Guard	Reset
t_1, t_{11}	$x_2 \geq 1$	none	t_2, t_9	$x_2 \leq 1$	none
t_3, t_{12}	$x_2 \leq 0$	$x_2 := 1.9, x_3 := 0$	t_4, t_{10}	$x_2 \geq 2$	$x_2 := 0.1, x_3 := 0$
t_5, t_{13}	$x_1 \geq 1$	none	t_7, t_{15}	$x_1 \leq 1$	none
t_6, t_{14}	$x_1 \leq 0$	$x_1 := 1.9, x_3 := 0$	t_8, t_{16}	$x_1 \geq 2$	$x_1 := 0.1, x_3 := 0$

Fig. 3. Description of hybrid automaton and randomly simulated trajectories.

2.1 Reachability Analysis

Rather than rely on finitely many simulations, we wish to exhaustively explore the set of reachable states of a hybrid system, in order to decide if a given set of unsafe states is reachable starting from a set of initial conditions. This is known as the *reachability analysis* problem.

Definition 1 (Reachability Problem). *Given a hybrid system $\mathcal{H}$, initial set of states X_0, unsafe set X_u and time horizon T, is there any trajectory that starts from some state in X_0 and reaches some state in X_u, within the given time horizon T?*

The reachability analysis problems can be *finite time horizon* problems where T is finite, or *infinite time horizon* problems if $T = \infty$. Naturally, the latter class of problems are harder than the former. Although a finite time horizon seems restrictive, there are many reasons why it is important: (a) often, it is known that failures would manifest within a finite time horizon if at all; (b) in many cases the reachability analysis problem has uncertain time varying parameters that makes the model invalid for infinite time horizons; or (c) the infinite time horizon problem is often harder to solve than the finite time horizon problem.

Reachability analysis is a fundamental verification problem for hybrid systems. Important correctness properties of hybrid systems are naturally posed as safety properties. Reachability analysis can also be used as a primitive step for reasoning about more complex liveness properties. Therefore, the question of decidability of reachability problem

is of great interest. Unfortunately, it is known that the reachability analysis problem is undecidable for all but the simplest classes of hybrid systems.

Asarin, Maler and Pnueli showed that hybrid systems with piece-wise constant dynamics (the simplest dynamics possible) already have an undecidable reachability problems for systems with 3 or more state variables [15]. Specifically, their model considers a partitioning of the state-space by convex polyhedra where each partition has its dynamics of the form $\dot{\vec{x}} = \vec{c}$ for a fixed $\vec{c}$. At the same time, the reachability analysis problem is undecidable for non-linear dynamical systems without any switching [94]. The finite time horizon reachability problem for linear dynamical systems (also known as the "continuous Skolem-Pisot problem" [21]) has been shown to be decidable provided an open number-theoretic conjecture called the Schaunel conjecture is true [36]. Broadly, we note that undecidability arises separately from the presence of switching between modes even if the dynamics are simple, or just from the continuous dynamics themselves without switching. The reachability problem for systems combining both switching and linear/non-linear dynamics is thus a computationally hard problem.

In the past three decades since these results, a number of sub-classes of hybrid automata have been identified for which the reachability problem is decidable, starting with Henzinger et al [71] who defined the class of *initialized rectangular hybrid automata*. Subsequently, O-minimal hybrid systems that allow for a more general class of dynamics in each mode were introduced by Laffarriere et al [82]. These have been generalized by Vladimerou et al [118]. In general, decidability results place restrictions on the form of transitions between modes as well as the dynamics in each mode. These restrictions ensure that the resulting system has a finite bisimulation quotient which can be used to check any temporal logic property. However, such restrictions are often not met by the systems which we are interested in reasoning about. As a result, numerous approaches attempt to solve the reachability problem by *over-approximating* the reachable set of states, or proposing a semi-algorithm that may not terminate in the worst case. The former class of approaches can help us conclude that the unsafe states are not reachable but fail to provide concrete counterexamples, whereas the latter class of approaches can fail by exhausting computational resources. We will now summarize a few approaches for solving the reachability analysis problem.

Abstraction-Based Techniques: The goal is to construct a finite-state abstraction that can be refined, possibly using counterexamples. Once the abstraction is constructed, we solve the reachability problem on this abstraction. If the unsafe set in the abstract state-space cannot be reached, we conclude the same for the original system. However, abstract counterexamples can be *spurious*: i.e, they need not correspond to a real execution of the concrete system. This can be addressed by refining the abstraction to rule away such counterexamples [12,9,41,61]. Interestingly, the abstractions need not necessarily be finite state. For instance, Prabhakar et al present an approach that considers rectangular hybrid automata as abstractions [102]. Hybridization is yet another approach that relies on locally abstracting nonlinear dynamics by linear dynamics while accounting for the error [46]. Abstraction-based approaches are quite versatile since they can be applied to a large class of hybrid systems with nonlinear dynamics. However, these approaches typically resort to tiling the state-space into discrete cells in order to handle

complex nonlinear dynamics. This often limits the number of state variables that can be treated by these techniques.

Dynamic Programming (Hamilton-Jacobi) Approaches: In this approach, the more general problem of controlling a hybrid system (with control and disturbance inputs) is considered as a game between two players. The goal is to characterize a controllable region (termed as the viability kernel), a subset of the state-space which excludes the undesired set of states, such that the controller can keep the system within this region no matter what disturbance signal is applied. This approach was proposed by Lygeros, Tomlin and Sastry [85], and leads to a partial differential equation (PDE) that needs to be solved in order to compute the controllable region. Subsequent work by Mitchell and Tomlin uses level-set methods to solve this PDE [93,92]. The dynamic programming approach is quite powerful: it applies to nonlinear systems and can compute a set of control strategies for guaranteeing safety. We note, however, that the reachability problems we have considered thus far do not involve control inputs. However, solving PDEs requires expensive numerical methods whose complexity can be exponential in the number of state variables.

Deductive Approaches: Deductive approaches are based on proving that the unsafe states are unreachable from the initial set by obtaining *(positive) invariant sets* of the hybrid system, and proving that these sets contain the initial set but exclude the unsafe set. Such invariants can be synthesized automatically using techniques from optimization and algebraic geometry [110,115,103,60]. However, invariant construction techniques are quite limited in the kind of systems that can be proven correct. In general, they play a supporting role inside a theorem prover that is built on top of a logic that supports reasoning about hybrid systems. The work of Platzer et al has constructed the rich framework of differential dynamic logic [101,99] and integrated this inside a theorem prover Keymaera [100,98]. In general, deductive approaches can prove that unsafe sets are not reachable. It is incumbent upon the user to deduce how the failure of a proof can lead to the construction of a counterexample.

Set-Propagation: Set propagation approaches rely on a chosen family of sets to represent sets of states (examples include ellipsoids, polyhedra, Taylor models) [4]. At each step, the reachable set is represented as a union of sets in this family. These algorithms propagate these sets for a small time step Δ so that an approximation that is valid for time up to t is now valid for time up to $t + \Delta$. By repeatedly iterating this process, an over-approximation of the reachable sets up to a finite time horizon T is produced. Set propagation techniques have been investigated extensively for linear systems beginning with the pioneering work on the tool HyTech for rectangular hybrid automata [70] and followed by a quick succession of approaches for richer classes of hybrid systems permitting nonlinear dynamics [112,14]. Currently, set propagation techniques are capable of analyzing linear dynamical systems with more than a billion state variables [19], linear hybrid systems with hundreds of state-variables [56] and nonlinear systems with tens of state variables [33]. Due to the over-approximate nature of these techniques, they are unable to produce concrete counter-example. Furthermore, these approaches are mostly restricted to finite time horizon problems.

However, there are successful reachability analysis techniques that fail to fit neatly into any of the categories above, or deserve to be described on their own.

Constraint Solving Approaches: An important class of approaches uses constraint solvers to show that no counterexample trace with a given length/time bound exists for a reachability problem. Ratschan and She achieve this by constructing an abstraction that is refined using ideas borrowed from constraint programming [105]. Franzle et al use a bounded-model checking approach that encodes the reachability problem as a set of constraints [55,72]. More recently, Kong et al build on top of their previous work on the dReal solver for nonlinear constraints [57] to build a reachability analyzer called dReach [80]. An important advantage of constraint solvers lies in their ability to search in a *non chronological* manner. I.e, they can search for counterexamples or prove their absence without necessarily having to start from time $t = 0$. However, the same factors that make the problem challenging hamper their performance. For one, the ability to reason about dynamical systems inside a constraint solver is a challenge. dReach uses other reachability analysis tools for nonlinear dynamical systems to approximate the solution to ODEs. Another challenge lies in choosing how to iteratively subdivide a large state-space during constraint solving in order to zero in on a counterexample or rule out counterexamples altogether.

Falsification: Whereas most approaches cited so far focus on *verification*, which is typically defined as "the process of establishing the truth, accuracy, or validity of something", approaches for *falsification* focus on disproving correctness by searching for a counterexample that establishes that an unsafe state is reachable starting from some initial state. Recently, there have been many approaches towards falsification based on using robustness of trajectory (its minimum distance to the unsafe set) as a fitness function that is minimized repeatedly using optimization [1]. Although they do not have guarantees of exhaustiveness, falsification techniques have been more successful in the industry wherein they provide a form of "smart fuzz-testing" for CPS [78,49].

3 Set-Propagation Approaches

In this section, we present the so-called set propagation approach for solving the reachability analysis problem. These approaches construct an over-approximation of the reachable set by (a) choosing a family of set representations such as ellipsoids to over-approximate sets of states; and (b) iteratively propagating the reachable state over-approximation forward in time according to the semantics of the hybrid automaton. Rather than attempt an exhaustive survey, we will briefly describe these approaches and highlight some of the successes. As mentioned earlier, a comprehensive survey of many of these techniques is available elsewhere [4]. Set-propagation approaches are analogous to techniques such as symbolic model checking and abstract interpretation that are commonly used for verifying digital circuits and computer software [16,45].

3.1 Linear Hybrid Systems

Linear Hybrid Systems (LHS) are characterized by multiple modes (also known as locations) and continuous states $\vec{x}$. A configuration, also called a state, of an LHS is denoted by a pair $(\vec{x}, \ell)$ such that $\vec{x}$ is the current valuation of the state variables and ℓ is the current location. Starting from an initial state, an LHS evolves in the following way.

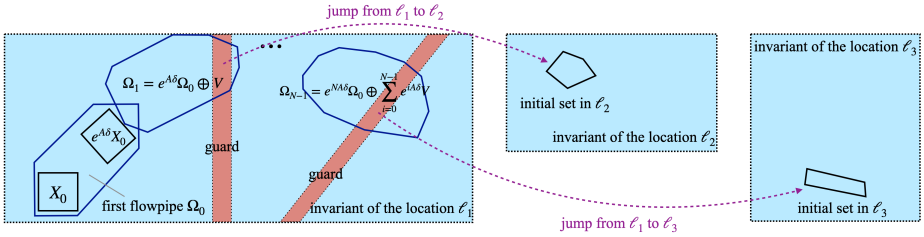


Fig. 4. Flowpipe construction for LHS.

Continuous Evolution. The state variable values change continuously within the location invariant under the continuous dynamics which is a linear ODE in the form of $\dot{x} = A\vec{x} + B\vec{w}$ associated with the current location. The parameters $\vec{w}$ are used to represent range-bounded uncertainties if there is any, and the invariant is defined by a conjunction of linear constraints over $\vec{x}$. In a continuous evolution, the location of the system does not change, and the values of $\vec{x}$ should satisfy the invariant.
Discrete Jump. The discrete dynamics of an LHS is defined by a set of transitions. The system instantly updates its current location according to the specification of a transition. More precisely, a transition can be made by satisfying the following requirements: (a) The current and new locations should be the start and end locations respectively of the transition; (b) The current state variable values should satisfy the transition *guard* which is defined by a conjunction of linear constraints over $\vec{x}$. A transition may also update the state variables $\vec{x}$ according to its linear reset rule.

Set-propagation approaches for LHS compute reachable sets for a bounded time horizon $[0, T]$ [1]. We illustrate the main algorithm in Fig. 4. Starting from a given initial state set X_0, the algorithm first over-approximates the reachable set by a convex set Ω_0 in the time interval of $[0, \delta]$ which is called the first time step according to a given step size $\delta > 0$. Next, we iteratively compute the sets $\Omega_1, \ldots, \Omega_{N-1}$, that are over-approximations of the reachable sets over the time intervals $[\delta, 2\delta], \ldots, [(N-1)\delta, N\delta]$, respectively, until $N\delta \geq T$. This step is usually done by repeatedly computing the flowpipes using the recurrent relation $\Omega_i = e^{A\delta}\Omega_{i-1} \oplus V$ wherein V is a convex set containing the impact from all uncertainties in a one-step evolution. When there is an invariant associated to the location, the flowpipes should also be intersected with it in order to exclude the unreachable states outside of the invariant. Finally, we compute over-approximations for the reachable sets under all possible discrete jumps, which themselves form initial sets in new locations. The algorithm repeatedly performs the three steps mentioned above, until all of executions in the time horizon are explored.

In order to represent sets, existing approaches use geometric objects such as polytopes [70,66,39,109], zonotopes [62] and ellipsoids [81], or symbolic representations for convex sets such as support functions [83]. These representations are closed under key operations that are performed by the reachability algorithms, including linear transformation and Minkowski sum in computing the recurrence relation. However, it is still challenging to handle discrete jumps, the main difficulty comes from the compu-

[1] Such reachable sets are often called flowpipes following the early work of Feng Zhao [120]

tation of the intersection with transition guards. Although a few of the representations such as polytopes are closed under intersections with sets defined by linear constraints, no representation can efficiently perform all the required set operations. Hence, much effort has been devoted to developing effective and efficient over-approximation algorithms for various intersection types, including ellipsoid/ellipsoid intersections [106], zonotope/hyperplane intersections [63], zonotope/polyhedron [3,5], and support function/support function [65]. The approaches are integrated into verification tools such as SpaceEx [56] and CORA [2].

Besides the above set-based approaches, a novel approach by Duggirala et al focuses on producing approximations at discrete time points using numerical simulations and the super-position principle for linear dynamics [51]. Such a technique is used in the tool Hylaa [18].

3.2 Nonlinear Hybrid Systems

NonLinear Hybrid Systems (NLHS) have an analogous structure to LHS except that the continuous dynamics may be defined by nonlinear ODEs, the guards and invariants may be defined by nonlinear constraints, and the reset rules of the jumps may also be nonlinear. Due to these nonlinearities, the reachability analysis on NLHS calls for a different class of approaches. The challenges are from answering the following two questions: *(i) How to compute the flowpipes for nonlinear ODEs?* and *(ii) How to compute nonlinear flowpipe/guard intersections?* We may categorize existing approaches as follows:

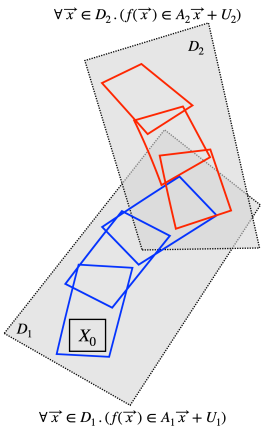


Fig. 5. Illustration of conservative linearization for nonlinear ODEs

Conservative Linearization of ODEs: It has already been shown that flowpipes for nonlinear ODEs can be effectively computed by repeatedly calling the following steps: (1) Conservatively linearizing the ODE to a range-bounded linear differential inclusion in the form of $\dot{\vec{x}} \in A\vec{x} + U$ in a local neighborhood in the state space; (2) Computing the flowpipes for the linear differential inclusion in the neighborhood. The algorithm goes to the step (1) with the last flowpipe which almost exceeds the neighborhood.

Althoff et al. [6] presented a framework that computing the reachable sets for a nonlinear system by conservatively linearizing the ODE on the fly. The linearization error is controlled by splitting the reachable sets. A more complex framework for over-approximating a nonlinear ODE by an LHS, which is also called *hybridization*, is presented by Dang et al. [46,47]. The approach computes bounded state subspaces which are called *hybridization domain* along the system executions, and linearizes the dynamics in those subspaces. Then the flowpipes can be obtained using an existing method for linear dynamics. Fig. 5 illustrates hybridization approach. The flowpipes for the non-

linear ODE $\dot{\vec{x}} = f(\vec{x})$ are computed based on two linear differential inclusions, each of which is an over-approximation of the nonlinear dynamics in its hybridization domain.

Verified Set-Valued Integration: Verified integration are set-based techniques which were introduced to provide guaranteed solutions for *initial value problems*: i.e, find $\vec{x}(t)$ for some time $t > 0$ for an ODE defined by $\dot{\vec{x}} = f(\vec{x}, t)$ with an initial condition $\vec{x}(0) \in X_0$. The main idea of the techniques is to iteratively compute a reachable set over-approximation over a time step. In each integration step, starting from the over-approximation set obtained at the end of the previous step, a new set which is guaranteed to contain the reachable set in the current step is computed by a set-based arithmetic such as interval arithmetic, and then verified by ensuring the contractiveness of the Picard operator over the set [91]. Several well-developed interval-based integration methods have already been implemented and released as tools such as VNODE-LP [95] and CAPD [77]. In order to better control the overestimation, Berz and Makino [87,25] developed the Taylor model-based integration approach. A *Taylor Model (TM)* is denoted by a pair (p, I) such that p is a polynomial and I is an interval remainder. A function $f(\vec{x})$ is over-approximated by a TM $(p(\vec{x}), I)$ over an interval domain, if for all $\vec{x} \in D$, we have that $f(\vec{x}) \in p(\vec{x}) + I$. Verified integration methods are also used in some constraint solving-based verification tools such as iSAT [54] and dReach [58].

Although the nonlinear continuous dynamics of an NLHS can be handled by the above methods, it is still very challenging to deal with the flowpipe/guard intersections since the guards may be defined by nonlinear constraints. Many reachability analysis frameworks or tools compute these intersections by constraint solving. Ariadne [23,24] uses intervals which are obtained from merging the interval solutions of the constraints defining the guard and flowpipes. In [104], Ramdani and Nedialkov described a method to compute an intersection by solving a constraint satisfiability problem, and use branch-and-prune to find the solution boxes. The method developed by Chen et al. [32] uses a combination of domain contraction and range over-approximation to over-approximate a TM flowpipe/guard intersection by a TM, and it is later implemented in the tool Flow* [33].

Besides, some other approaches such as the technique implemented in the tool C2E2 [50] which uses set propagation method under the hood but simulates trajectories to construct discrepancy functions.

We have briefly described reachability analysis techniques based on set-propagation in this section. Whereas the approaches for linear hybrid systems can now be considered *mature* by most reasonable standards, the same cannot be said for general nonlinear hybrid systems. For instance, our own tool Flow* supports many different heuristic strategies for computing reachable sets efficiently. The choice of such a strategy requires setting time steps, polynomial orders, aggregation heuristics and many other details that are internal to the algorithm. However, different choices of these parameters yield vastly different results in terms of computation speed and the overestimation error in the results. Understanding the interplay between these parameters will help improve the usability of nonlinear reachability analysis techniques.

4 Scaling Up Reachability Analysis

In this section, we briefly describe some novel approaches that have been applied to scale up reachability analysis, especially for nonlinear systems. As discussed previously, the work of Bak et al cleverly exploits the sparsity in the system's dynamics as well as the properties of the initial and unsafe sets to compute the projections of the reachable sets over linear systems with billions of state variables [19]. In this section, we will discuss some recent work on scaling up reachability analysis.

Exploiting Monotonicity: Monotone systems are those where there is a partial order between states in the state-space such that if $\vec{x}(0) \preceq \vec{y}(0)$ for two initial states, then $\vec{x}(t) \preceq \vec{y}(t)$ for the respective trajectories encountered starting from these initial states. Monotonicity is natural in many types of systems such as traffic networks. Coogan and Arcak show how monotone systems lend themselves to efficient computation of abstractions that can be used to solve reachability analysis problems [44,43]. In fact, their work also extends the classic notion of monotonicity to apply to a wider class of systems. Under these monotonicity assumptions, it can be shown that the reachable set for a hyper-rectangular set is obtained precisely by simulating two diagonally opposite corner points. As a result, it is possible to solve verification problems for monotone systems with large state spaces.

Exploiting Symmetries: Another approach that exploits special structure in the system concerns symmetries in the system description. These symmetries can be discrete symmetries wherein permutations of the state variables can lead to the original system back. The permutations define an equivalence class amongst the state variables, and therefore, a smaller system can be obtained by "lumping" system variables together in a natural manner. This approach has been shown to work for nonlinear systems derived from gene regulatory networks [28]. However, its application requires that the initial conditions of the lumped variables agree with each other. Another approach considers continuous (Lie) symmetries, including invariance of the system's dynamics to translations and rotations of the coordinate frames. This is a powerful approach that can be exploited to speed up reachability analysis. Maidens and Arcak exploit symmetries for backward reachability in order to synthesize controllers using the dynamic programming framework [86]. A different approach to ensuring efficiency by exploiting symmetry is considered by Sibai et al [111], particularly for the case when a system involves multiple agents. Their approach uses previously caches reachable set computations: for instance, some set $X_{t+\Delta}$ is reachable from some other set X_t in time Δ. Symmetry allows us to reuse this information for a different set Y that may not be the same as X_t but related to it through a transformation. An almost identical approach was also adopted (independently) by the second author jointly with Chou and Yoon, wherein they show how reachable sets can be pre-computed offline in order to support rapid table lookups to perform predictive runtime monitoring [38]. This approach was designed specifically to exploit invariance to rotation and translations for vehicle models.

Decompositions based on System Structure: Decompositions are a very promising approach to reducing reachability problems for systems over higher dimensional state-spaces into problems that involve multiple systems over a subset of the state variables.

The key idea is to consider how the state-variables in the dynamics depend on each other through a dependency graph.

Figure 6 shows an example of a Dubin's vehicle with a "sampled-data" control strategy where the control inputs u_1, u_2 are computed using the state at a previous time step. Therefore, for the duration of a time step Δ, they may be thought of as a constant. Thus, instead of considering 4 state variables together, the reachability algorithm can separately integrate the subsystems for ψ, v and use these in turn to separately compute reachable set estimates for x, y. These are effectively systems with a single state variable. This idea was considered independently by Mo Chen et al [30,29] in the context of the Hamilton-Jacobi approach and by the authors of this paper in the context of nonlinear set-based reachability [35]. In both cases, a dependency graph is constructed and decomposed into strongly connected components. Furthermore, our work also focused on approximate methods by "cutting" continuous feedback loops. Decomposition methods are very powerful in that they allow us to treat "loosely coupled" systems with hundreds of state variables. Recently, Sankaranarayanan used a tree-width decomposition approach to consider overlapping partitions of the system variables. The system is then projected into multiple abstract subsystems each involving one of the partitions. The key idea is that the partitions can exchange information using an algorithm inspired by belief propagation [108].

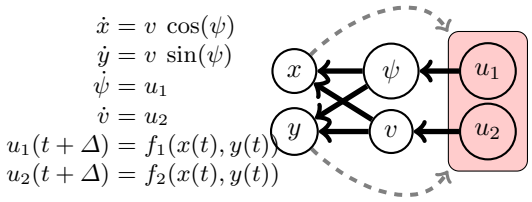


Fig. 6. A 2 dimensional Dubin's vehicle model and its dependency graph. The dashed line shows feedback from the vehicle position at a previous time step to the control inputs at the subsequent time step.

Although, we have presented a few promising approaches to scaling up reachability analysis, there are currently numerous challenges that require new approaches. We mention a few promising areas for future work.

Model Order Reductions: The reachability problem for large CPS often involve safety properties that are expressed over very few system variables. It is thus interesting to consider techniques akin to model-order reductions that can speed up reachability analysis. Model-order reductions have been explored in the past by using standard approaches in that area to reduce the dimensionality of the state-space [67,37]. However, these approaches do not preserve soundness. Recent approaches that have exploited the fact that initial conditions and unsafe sets involve a few of the system variables with great success and without sacrificing soundness for linear dynamical systems [19]. A new general approach to such reductions that allows us to avoid computing reachability information for "unnecessary" state variables in a sound manner is needed.

Koopman Operator-based Linearization: Another promising approach is to convert linear systems into nonlinear systems in a higher dimensional space through the theory

of Koopman operators [90]. The key idea here is to consider a new state-space in terms of functions $\{f_1(\vec{x}), \ldots, f_N(\vec{x})\}$ wherein the derivative of each f_i can be written as an affine function of the other functions. This helps us abstract the trajectories of the system by a linear system. Reachability analysis over this system gives us reachable set over-approximations of the original nonlinear systems. The key here is to discover appropriate basis functions f_i (so-called Koopman invariant subspace), and there is no guarantee that these functions will be polynomials. Earlier work by Sankaranarayanan explored an iterative approach to discovering a basis where f_i are all polynomials [107]. However, there is no guarantee that such a basis would exist. More recently, Bak et al present an algorithm that assumes that a Koopman-invariant subspace is known or approximated through techniques such as dynamic mode decomposition. It then shows how the resulting reachability analysis problem can be solved [17]. In general, ideas such as Koopman operator-based linearization provide alternatives to existing ways of abstracting nonlinear dynamics which could be an interesting way forward to make reachability analysis more scalable.

5 Neural Network Controlled Systems

With the rapid development of machine learning techniques, more and more CPS are using learning-enabled components such as neural networks for making decisions in strategic situations. Since most of such learning-enabled CPS are safety-critical, it is important to develop new methods for ensuring their safety. However, most of the verification methods developed for pure discrete or even hybrid systems can hardly be applied due to the complex system behavior produced by the interaction between the learning-enabled components and the others.

Recently, a great amount of work has been devoted to developing new formal methods for verifying *neural network controlled systems (NNCS)* which are a basic class of learning-enabled CPS but very challenging to verify. Fig.7 shows the formal model of NNCS. It is a class of sampled-data systems in which the plant, i.e., the continuous dynamics, is defined by an ODE over the state variable(s) x and the control input(s) u, while the controller is a *Feed-forward Neural Network (FNN)*. Given a control step size $\delta_c > 0$, at the time $t = k\delta_c$ for every $k = 0, 1, \ldots$, the controller reads the current state of the plant and computes the control input which will be used immediately for δ_c-time, i.e., in the current control step, by the plant. Since a control input is obtained from the FNN, the computation time is ignored in the system execution due to the fast response of neural networks.

NNCS are continuous systems but not necessarily differentiable due to the non-differentiable activation functions such as ReLU in the neural networks. Given an NNCS, the system execution from an initial state x_0 is deterministic and can be defined by a flow map function Φ such that $\Phi(x_0, t)$ denotes the system state at a future time $t \geq 0$. According to the behavior of NNCS, for $k = 0, 1, \ldots$, $x_k = \Phi(x_0, k\delta_c)$ is the initial state of the $(k+1)$-st control step, and the control input used in that step is derived as $u_k = \kappa(x_k)$ wherein $\kappa(\cdot)$ denotes the input-output mapping of the FNN controller. Hence, not only the reachable states but also the control inputs in a system execution are determined by the initial state x_0. Fig. 8 illustrates the dependency between a reachable

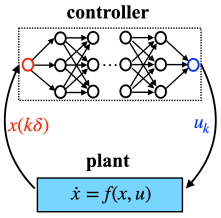


Fig. 7. Model of NNCS

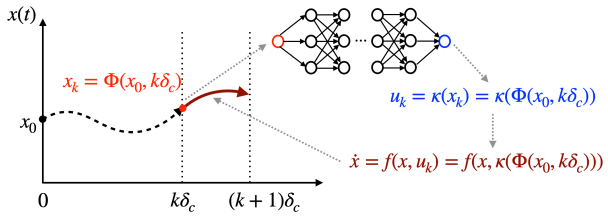


Fig. 8. Dependency on the initial state

state and the initial state. In the case of a set X_0 of initial states, the exact reachable set at a time $t \geq 0$ is denoted by $\Phi(X_0, t) = \{\Phi(x_0, t) \mid x_0 \in X_0\}$, and the set of the control inputs used in the $(k+1)$-st step is given by $U_k = \kappa(X_k)$ wherein $X_k = \Phi(X_0, k\delta_c)$. The core technique in a reachability analysis approach for NNCS is an algorithm to over-approximate the range of the flow map Φ over a time interval w.r.t. a given set of initial states.

Most of the existing reachability analysis methods use a set-propagation scheme to compute over-approximate reachable set segments, i.e., flowpipes, over a finite number K of control steps. Starting with a given initial set $\hat{X}_0 = X_0$, the main algorithm repeatedly performs the following two steps to compute the flowpipes in the $(k+1)$-st control step for $k = 0, 1, \ldots, K-1$:

(i) *Computing the output range of the controller.* In this step, a set $\hat{U}_k$ which is guaranteed to contain all of the control inputs in U_k is computed.

(ii) *Flowpipe computation for the plant.* The step computes the flowpipes for the plant ODE $\dot{x} = f(x, u)$ with the local initial set $\hat{X}_k$ and the control input range $u \in \hat{U}_k$. Then the local initial set of the next iteration is computed as $\hat{X}_{k+1}$ which is an over-approximation of $\Phi(X_0, (k+1)\delta_c)$.

They can be classified as follows, based on the over-approximation schemes.

Directly over-approximating reachable sets. A reachability analysis algorithm on NNCS can be developed as a combination use of a method for computing FNN output ranges and an existing flowpipe construction technique for ODEs. To do so, one may need to use a uniform set representation for FNN output ranges and ODE flowpipes. Many FNN output ranges analysis techniques [74,79,59,53,119,116] can be extended and work cooperatively with the existing reachability tools for ODEs [95,33,2]. The main advantage of this scheme is twofold. Firstly, there is no need to develop a new technique from scratch, and the correctness of the composed approach can be proved easily based on the correctness of the existing methods. Secondly, the performance of the approach is often good since it can use well engineered tools as primitives. However, on the other hand, the relationship between the control inputs and the plant states (see Fig.8) are not explicitly represented in this approach. This may lead to a overestimation when the plant dynamics is nonlinear or the initial set is large, making the resulting bounds less useful in proving properties of interest.

Over-approximating flow map functions. More accurate over-approximations can be obtained if a reachability method tries to over-approximate the flow map function Φ instead of its image. It is well-known that functional over-approximations such as TMs

have an apparent advantage in accuracy over the pure range over-approximation methods for nonlinear dynamical systems [31]. Recent work has applied interval, polynomial and TM arithmetic to obtain over-approximations for NNCS flow maps [52,73,76]. Those techniques are often able to compute more accurate flowpipes than the methods in Category (A). On the other hand, the functional over-approximation methods are often computationally expensive due to the computation of nonlinear multivariate polynomials for tracking the dependencies.

Neural Network Control Systems are an emerging area that has seen an explosion of interest in the recent years. Persistent challenges include the need to handle ever larger networks and also the need to integrate rich sensor inputs from sensors such as camera and LiDAR. This poses a hard modeling challenge that requires us to link the state of the system with the possible inputs that these sensors may provide. The work of Shoukry et al presents an interesting case for solving this challenge when the system's operating environment is known [113]. This paper represents a very promising line of work that can benefit from further investigation.

6 Conclusions

We have thus far introduced a wide variety of techniques that have been explored for solving the reachability analysis of CPS, integrating ideas from diverse disciplines, ranging from Logic to control theory. We have also briefly surveyed exciting new frontiers, including the emerging topic of verifying safety of systems controlled by neural networks. While it is clear that the research on reachability analysis techniques have come a long way, numerous challenges remain. For one, many of the techniques remain inaccessible to control engineers due to many reasons. There is a gap between the rich expressive modeling formalisms that are used by engineers such as Simulink/Stateflow, and the capabilities of existing reachability analysis tools that work on hybrid automata models. The translation from one to another is not simple. Tools like C2E2 are seeking to bridge this gap by allowing model specifications inside Stateflow [50], but more needs to happen along this front before such tools can be said to be developer friendly. Besides these practical concerns, there are numerous open challenges and new frontiers. One such area that has not been mentioned in this survey concerns the reachability analysis of stochastic hybrid systems. Another open area concerns reachability analysis for systems whose feedback control inputs are specified in an *implicit manner*: i.e, they are specified as minimizers of some cost functions. Such systems arise from many domains such as model-predictive control algorithms or physics-based models that are described using potential fields.

To conclude, we revisit the question in the title "*Are we there yet*?". Briefly, we would conclude at this time that reachability analysis of CPS has *gone places* without yet arriving at a destination!

Acknowledgments: We thank Klaus Havelund for helpful comments and suggestions. Sankaranarayanan gratefully acknowledges support from the NSF through award numbers 1815983, 1836900 and 1932189. Chen gratefully acknowledges the support from the US Air Force Research Laboratory (AFRL) under contract FA8650-16-C-2642. All opinions are those of the authors and not necessarily of NSF or AFRL.

References

1. Houssam Abbas, Georgios Fainekos, Sriram Sankaranarayanan, Franjo Ivancic, and Aarti Gupta. Probabilistic temporal logic falsification of cyber-physical systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 12(12s):95, 2013.
2. M. Althoff. An introduction to CORA 2015. In *Proc. of ARCH'15*, volume 34 of *EPiC Series in Computer Science*, pages 120–151. EasyChair, 2015.
3. M. Althoff, O. Stursberg, and M. Buss. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear Analysis: Hybrid Systems*, 4(2):233–249, 2010.
4. Matthias Althoff, Goran Frehse, and Antoine Girard. Set propagation techniques for reachability analysis. *Annual Review of Control, Robotics, and Autonomous Systems*, 4, 2021.
5. Matthias Althoff and Bruce H. Krogh. Avoiding geometric intersection operations in reachability analysis of hybrid systems. In *Proc. of HSCC'12*, pages 45–54. ACM, 2012.
6. Matthias Althoff, Olaf Stursberg, and Martin Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *Proc. of CDC'08*, pages 4042–4048. IEEE, 2008.
7. Rajeev Alur. *Principles of Cyber-Physical Systems*. MIT Press, 2015.
8. Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors, *Hybrid Systems*, pages 209–229, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.
9. Rajeev Alur, Thao Dang, and Franjo Ivančić. Counter-example guided predicate abstraction of hybrid systems. In *TACAS*, volume 2619 of *LNCS*, pages 208–223. Springer, 2003.
10. Rajeev Alur and David L. Dill. Automata for modeling real-time systems. In M.S. Paterson, editor, *ICALP 90: Automata, Languages, and Programming*, volume 443 of *Lecture Notes in Computer Science*, page 322–335. Springer-Verlag, 1990.
11. Rajeev Alur and David L. Dill. A theory of timed automata. volume 126, page 183–235, 1994.
12. Rajeev Alur, Thomas A. Henzinger, G. Lafferriere, and George Pappas. Discrete abstractions of hybrid systems. *Proc. of IEEE*, 88(7):971–984, 2000.
13. P.J. Antsaklis, K.M. Passino, and S.J. Wang. An introduction to autonomous control systems. *IEEE Control Systems Magazine*, 11(4):5–13, 1991.
14. Eugene Asarin, Thao Dang, and Oded Maler. The d/dt tool for verification of hybrid systems. In *CAV*, volume 2404 of *Lecture Notes in Computer Science*, pages 365–370. springer, 2002.
15. Eugene Asarin, Oded Maler, and Amir Pnueli. Reachability analysis of dynamical systems having piecewise-constant derivatives. *Theoretical Computer Science*, 138:35–66, 1995.
16. Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.
17. Stanley Bak, Sergiy Bogomolov, Parasara Sridhar Duggirala, Adam R. Gerlach, and Kostiantyn Potomkin. Reachability of black-box nonlinear systems after koopman operator linearization. In *Analysis and Design of Hybrid Systems (ADHS)*, volume 54 of *IFAC-PapersOnLine*, pages 253–258. Elsevier, 2021.
18. Stanley Bak and Parasara Sridhar Duggirala. Hylaa: A tool for computing simulation-equivalent reachability for linear systems. In *Proc. of HSCC'17*, pages 173–178. ACM, 2017.
19. Stanley Bak, Hoang-Dung Tran, and Taylor T. Johnson. Numerical verification of affine systems with up to a billion dimensions. HSCC '19, page 23–32, New York, NY, USA, 2019. Association for Computing Machinery.

20. Thomas Ball and Sriram K. Rajamani. The SLAM project: debugging system software via static analysis. In *POPL '02: Proceedings of the 29th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 1–3, New York, NY, USA, 2002. ACM.
21. Paul C. Bell, Jean-Charles Delvenne, Raphaël M. Jungers, and Vincent D. Blondel. The continuous skolem-pisot problem. *Theoretical Computer Science*, 411(40):3625–3634, 2010.
22. A. Benveniste and P. Le Guernic. Hybrid dynamical systems theory and the signal language. *IEEE Transactions on Automatic Control*, 35(5):535–546, 1990.
23. L. Benvenuti, D. Bresolin, A. Casagrande, P. Collins, A. Ferrari, E. Mazzi, A. Sangiovanni-Vincentelli, and R. Villa. Reachability computation for hybrid systems with Ariadne. In *Proc. of the 17th IFAC World Congress*. IFAC Papers-OnLine, 2008.
24. Luca Benvenuti, Davide Bresolin, Pieter Collins, Alberto Ferrari, Luca Geretti, and Tiziano Villa. Ariadne: Dominance checking of nonlinear hybrid automata using reachability analysis. In *Proc. of RP'12*, volume 7550 of *LNCS*, pages 79–91. Springer, 2012.
25. M. Berz and K. Makino. Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor models. *Reliable Computing*, 4:361–369, 1998.
26. Dirk Beyer, Thomas A. Henzinger, Ranjit Jhala, and Rupak Majumdar. The software model checker BLAST. *STTT*, 9(5-6):505–525, 2007.
27. Bruno Blanchet, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, and Xavier Rival. A static analyzer for large safety-critical software. In *Prog. Lang. Design & Implementation*, pages 196–207. ACM Press, 2003.
28. Luca Cardelli, Mirco Tribastone, Max Tschaikowski, and Andrea Vandin. Symbolic computation of differential equivalences. volume 51, pages 137–150, 04 2016.
29. M. Chen, S. L. Herbert, M. S. Vashishtha, S. Bansal, and C. J. Tomlin. Decomposition of Reachable Sets and Tubes for a Class of Nonlinear Systems. *ArXiv e-prints*, 2017.
30. Mo Chen, Sylvia Herbert, and Claire Tomlin. Exact and efficient hamilton-jacobi-based guaranteed safety analysis via system decomposition. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017. To appear, also at arXiv:1609.05248.
31. X. Chen. *Reachability Analysis of Non-Linear Hybrid Systems Using Taylor Models*. PhD thesis, RWTH Aachen University, 2015.
32. X. Chen, E. Ábrahám, and S. Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. In *Proceedings of the 33rd IEEE Real-Time Systems Symposium (RTSS'12)*, pages 183–192. IEEE Computer Society, 2012.
33. X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *Proc. of CAV'13*, volume 8044 of *LNCS*, pages 258–263. Springer, 2013.
34. Xin Chen, Souradeep Dutta, and Sriram Sankaranarayanan. Formal verification of a multibasal insulin infusion control model. In *Workshop on Applied Verification of Hybrid Systems (ARCH)*, page 16. Easychair, 2017.
35. Xin Chen and Sriram Sankaranarayanan. Decomposed reachability analysis for nonlinear systems. In *IEEE Real Time Systems Symposium (RTSS)*, page 13–24. IEEE Press, 2016.
36. Ventsislav Chonev, Joël Ouaknine, and James Worrell. On the skolem problem for continuous linear dynamical systems. In *ICALP 2016*, volume 55 of *LIPIcs*, pages 100:1–100:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.
37. Yi Chou, Xin Chen, and Sriram Sankaranarayanan. A study of model-order reduction techniques for verification. In Alessandro Abate and Sylvie Boldo, editors, *Numerical Software Verification*, volume 10381 of *Lecture Notes in Computer Science*, pages 98–113. Springer, 2017.
38. Yi Chou, Hansol Yoon, and Sriram Sankaranarayanan. Predictive runtime monitoring of vehicle models using bayesian estimation and reachability analysis. In *Intl. Conference on Intelligent Robots and Systems (IROS)*, pages 2111–2118. IEEE Press, 2020.

39. Alongkrit Chutinan and Bruce Krogh. Computing polyhedral approximations to flow pipes for dynamic systems. In *Proceedings of IEEE CDC*. IEEE press, 1998.
40. Edmund Clarke, Daniel Kroening, and Flavio Lerda. A tool for checking ANSI-C programs. In *TACAS*, volume 2988 of *lncs*, 2004.
41. Edmund M. Clarke, Ansgar Fehnker, Zhi Han, Bruce H. Krogh, Olaf Stursberg, and Michael Theobald. Verification of hybrid systems based on counterexample-guided abstraction refinement. In *TACAS*, volume 2619 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2003.
42. Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 1999.
43. Samuel Coogan. Mixed monotonicity for reachability and safety in dynamical systems. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 5074–5085. IEEE Press, 2020.
44. Samuel Coogan and Murat Arcak. Efficient finite abstraction of mixed monotone systems. In Antoine Girard and Sriram Sankaranarayanan, editors, *HSCC'15*, pages 58–67. ACM, 2015.
45. Patrick Cousot. *Principles of Abstract Interpretation*. MIT Press, 2021.
46. Thao Dang, Oded Maler, and Romain Testylier. Accurate hybridization of nonlinear systems. In *Proc. of HSCC'10*, pages 11–20. ACM, 2010.
47. Thao Dang and Romain Testylier. Hybridization domain construction using curvature estimation. In *Proc. of HSCC'11*, pages 123–132. ACM, 2011.
48. David Delmas and Jean Souyris. Astrée: from research to industry. In *Proc. 14th International Static Analysis Symposium, SAS 2007*, volume 4634 of *LNCS*, pages 437–451. Springer, Berlin, 2007.
49. Alexandre Donzé. BreachFlows: Simulation-Based Design with Formal Requirements for Industrial CPS (Extended Abstract). In *Workshop on Autonomous Systems Design (ASD 2020)*, volume 79 of *OpenAccess Series in Informatics (OASIcs)*, pages 5:1–5:5, 2020.
50. Parasara Sridhar Duggirala, Sayan Mitra, Mahesh Viswanathan, and Matthew Potok. C2e2: A verification tool for stateflow models. In Christel Baier and Cesare Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 68–82. Springer, 2015.
51. Parasara Sridhar Duggirala and Mahesh Viswanathan. Parsimonious, simulation based verification of linear systems. In *Computer Aided Verification (CAV)*, Lecture Notes in Computer Science, pages 477–494, Germany, 2016. Springer.
52. Souradeep Dutta, Xin Chen, and Sriram Sankaranarayanan. Reachability analysis for neural feedback systems using regressive polynomial rule inference. In Necmiye Ozay and Pavithra Prabhakar, editors, *Proc. of HSCC'19*, pages 157–168. ACM, 2019.
53. Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. Output range analysis for deep feedforward neural networks. In *Proc. of NFM'18*, volume 10811 of *LNCS*, pages 121–138. Springer, 2018.
54. Andreas Eggers, Nacim Ramdani, Nedialko S. Nedialkov, and Martin Fränzle. Improving SAT modulo ODE for hybrid systems analysis by combining different enclosure methods. In *Proc. of SEFM'11*, volume 7041 of *LNCS*, pages 172–187. Springer, 2011.
55. M. Fränzle, C. Herde, S. Ratschan, T. Schubert, and T. Teige. Efficient solving of large non-linear arithmetic constraint systems with complex Boolean structure. *JSAT—Journal on Satisfiability, Boolean Modeling and Computation, Special Issue on SAT/CP Integration*, 1:209–236, 2007.
56. G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: Scalable verification of hybrid systems. In *Proc. CAV'11*, volume 6806 of *LNCS*, pages 379–395. Springer-Verlag, 2011.

57. S. Gao, S. Kong, and E. M. Clarke. dReal: an SMT solver for nonlinear theories over the reals. In *Proc. CADE'13*, volume 7898 of *Lecture Notes in Computer Science*, pages 208–214. Springer, 2013.
58. Sicun Gao, Soonho Kong, and Edmund M. Clarke. Satisfiability modulo odes. In *Proc. of FMCAD'13*, pages 105–112. IEEE, 2013.
59. Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. AI2: safety and robustness certification of neural networks with abstract interpretation. In *Proc. of S& P'18*, pages 3–18. IEEE Computer Society, 2018.
60. Khalil Ghorbal, Andrew Sogokon, and André Platzer. A hierarchy of proof rules for checking positive invariance of algebraic and semi-algebraic sets. *Comput. Lang. Syst. Struct.*, 47:19–43, 2017.
61. Ranojoy Ghosh and Claire J. Tomlin. Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modeling: Delta-notch protein signaling. *IEE Transactions on Systems Biology*, 1(1):170–183, June 2004.
62. Antoine Girard. Reachability of uncertain linear systems using zonotopes. In *HSCC*, volume 3414 of *Lecture Notes in Computer Science*, pages 291–305. Springer, 2005.
63. Antoine Girard and Colas Le Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In *Proc. of HSCC'08*, volume 4981 of *Lecture Notes in Computer Science*, pages 215–228. Springer, 2008.
64. A. Gollu and P. Varaiya. Hybrid dynamical systems. In *Proceedings of the 28th IEEE Conference on Decision and Control,*, pages 2708–2712 vol.3, 1989.
65. Colas Le Guernic and Antoine Girard. Reachability analysis of hybrid systems using support functions. In *Proc. of CAV'09*, volume 5643 of *Lecture Notes in Computer Science*, pages 540–554. Springer, 2009.
66. Nicolas Halbwachs, Yann-Eric Proy, and Patrick Roumanoff. Verification of real-time systems using linear relation analysis. *Formal Methods in System Design*, 11(2):157–185, 1997.
67. Zhi Han and Bruce Krogh. Reachability analysis of hybrid control systems using reduced-order models. In *Proceedings of the American Control Conference*, volume 2, pages 1183 – 1189, 01 2004.
68. John Harrison. Formal methods at Intel - an overview. In *Proceedings of the Second NASA Formal Methods Symposium (NFM)*, 2010.
69. Klaus Havelund and Thomas Pressburger. Model checking JAVA programs using JAVA PathFinder. *Int. J. Software Tools Technol. Trans.*, 2(4):366–381, Mar 2000.
70. Thomas A. Henzinger and Pei-Hsin Ho. HYTECH: The Cornell hybrid technology tool. In *Hybrid Systems II*, volume 999 of *LNCS*, pages 265–293. Springer, 1995.
71. Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57(1):94 – 124, 1998.
72. C. Herde, A. Eggers, and T. Franzle, M. Teige. Analysis of hybrid systems using HySAT. In *Third International Conference on Systems, 2008. ICONS 08.*, pages 13–18. IEEE, 2008.
73. Chao Huang, Jiameng Fan, Wenchao Li, Xin Chen, and Qi Zhu. Reachnn: Reachability analysis of neural-network controlled systems. *ACM Trans. Embed. Comput. Syst.*, 18(5s):106:1–106:22, 2019.
74. Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *Proc. of CAV'17*, volume 10426 of *LNCS*, pages 3–29. Springer, 2017.
75. Franjo Ivančić, Ilya Shlyakhter, Aarti Gupta, and Malay K. Ganai. Model checking C programs using f-soft. In *ICCD*, pages 297–308. IEEE Computer Society, 2005.
76. Radoslav Ivanov, Taylor J. Carpenter, James Weimer, Rajeev Alur, George J. Pappas, and Insup Lee. Verifying the safety of autonomous systems with neural network controllers. *ACM Trans. Embed. Comput. Syst.*, 20(1):7:1–7:26, 2021.

77. T. Kapela, M. Mrozek, P. Pilarczyk, D. Wilczak, and P. Zgliczyński. Capd - a rigorous toolbox for computer assisted proofs in dynamics. Technical report, Jagiellonian University, 2010.
78. James Kapinski, Jyotirmoy V. Deshmukh, Xiaoqing Jin, Hisahiro Ito, and Kenneth R. Butts. Simulation-guided approaches for verification of automotive powertrain control systems. In *American Control Conference, ACC 2015, Chicago, IL, USA, July 1-3, 2015*, pages 4086–4095. IEEE, 2015.
79. Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *Proc. of CAV'17*, volume 10426 of *LNCS*, pages 97–117. Springer, 2017.
80. S. Kong, S. Gao, W. Chen, and E. M. Clarke. dreach: δ-reachability analysis for hybrid systems. In *Proc. of TACAS'15*, volume 9035 of *LNCS*, pages 200–205. Springer, 2015.
81. Alexander B. Kurzhanski and Pravin Varaiya. Ellipsoidal techniques for reachability analysis. In *HSCC*, volume 1790 of *Lecture Notes in Computer Science*, pages 202–214. Springer, 2000.
82. Gerardo Lafferriere, George Pappas, and Shankar Sastry. O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems*, 13:1–21, 02 2000.
83. Colas Le Guernic and Antoine Girard. Reachability analysis of linear systems using support functions. *Nonlinear Analysis: Hybrid Systems*, 4(2):250–262, 2010. IFAC World Congress 2008.
84. John Lygeros. Lecture notes on hybrid systems, 2004. Notes for ENSIETA short course.
85. John Lygeros, Claire Tomlin, and Shankar Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3), March 1999.
86. John Maidens and Murat Arcak. Exploiting symmetry for discrete-time reachability computations. *IEEE Control Systems Letters*, 2(2):213–217, 2018.
87. K. Makino and M. Berz. Remainder differential algebras and their applications. In M. Berz et al., editor, *Computational Differentiation: Techniques, Applications, and Tools*, pages 63–75. SIAM, 1996.
88. Oded Maler. Amir pnueli and the dawn of hybrid systems. In *Proc. Hybrid Systems: Computation and Control*, page 293–295. Association for Computing Machinery, 2010.
89. Oded Maler, Zohar Manna, and A. Pnueli. From timed to hybrid systems. In J.W. de Bakker, K. Huizing, W.-P. de Roever, and G. Rozenberg, editors, *Real Time: Theory in Practice*, volume 600 of *Lecture Notes in Computer Science*, page 447–484. Springer-Verlag, 1992.
90. Alexandre Mauroy, Igor Mezić, and Yoshihiko Susuki. *The Koopman Operator in Systems and Control*. Lecture Notes in Control and Information Sciences. Springer, 2020.
91. James D. Meiss. *Differential Dynamical Systems*. SIAM publishers, 2007.
92. Ian Mitchell. Toolbox of level-set methods. Technical report, 2007. UBC Department of Computer Science Technical Report TR-2007-11.
93. Ian Mitchell and Claire Tomlin. Level set methods for computation in hybrid systems. In *HSCC*, volume 1790 of *LNCS*, pages 310–323. Springer, 2000.
94. Cristopher Moore. Unpredictability and undecidability in dynamical systems. *Phys. Rev. Lett.*, 64:2354–2357, May 1990.
95. N. S. Nedialkov. Implementing a rigorous ode solver through literate programming. In A. Rauh and E. Auer, editors, *Modeling, Design, and Simulation of Systems with Uncertainties*, volume 3 of *Mathematical Engineering*, chapter Mathematical Engineering, pages 3–19. Springer Berlin Heidelberg, 2011.
96. X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An approach to the description and analysis of hybrid systems. In Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors, *Hybrid Systems*, pages 149–178, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

97. P. Peleties and R. DeCarlo. A modeling strategy with event structures for hybrid systems. In *Proceedings of the 28th IEEE Conference on Decision and Control,*, pages 1308–1313 vol.2, 1989.
98. André Platzer. *Logical Foundations of Cyber-Physical Systems*. Springer Publishing Company, Incorporated, 1st edition, 2018.
99. André Platzer and Edmund Clarke. Computing differential invariants of hybrid systems as fixedpoints. *Formal Methods in Systems Design*, 35(1):98–120, 2009.
100. André Platzer and Jan-David Quesel. KeYmaera: A hybrid theorem prover for hybrid systems. In *IJCAR*, volume 5195 of *LNCS*, pages 171–178. Springer, 2008.
101. André Platzer and Jan-David Quesel. Logical verification and systematic parametric analysis in train control. In Magnus Egerstedt and Bud Mishra, editors, *HSCC*, volume 4981 of *LNCS*, pages 646–649. Springer, 2008.
102. Pavithra Prabhakar, Parasara Sridhar Duggirala, Sayan Mitra, and Mahesh Viswanathan. Hybrid automata-based CEGAR for rectangular hybrid systems. *Form. Methods Syst. Des.*, 46(2):105–134, 2015.
103. Stephen Prajna and Ali Jadbabaie. Safety verification using barrier certificates. In *HSCC*, volume 2993 of *LNCS*, pages 477–492. Springer, 2004.
104. N. Ramdani and N. S. Nedialkov. Computing reachable sets for uncertain nonlinear hybrid systems using interval constraint-propagation techniques. *Nonlinear Analysis: Hybrid Systems*, 5(2):149–162, 2011.
105. Stefan Ratschan and Zhikun She. Safety verification of hybrid systems by constraint propagation based abstraction refinement. In *HSCC*, volume 3414 of *Lecture Notes in Computer Science*, pages 573–589. Springer, 2005.
106. Lluís Ros, Assumpta Sabater, and Federico Thomas. An ellipsoidal calculus based on propagation and fusion. *IEEE Trans. Syst. Man Cybern. Part B*, 32(4):430–442, 2002.
107. Sriram Sankaranarayanan. Change of basis abstractions for non-linear hybrid systems. *Nonlinear Analysis: Hybrid Systems*, 19:107–133, 2016.
108. Sriram Sankaranarayanan. Reachability analysis using message passing over tree decompositions. In *International Conference on Computer-Aided Verification (CAV)*, volume 12224 of *Lecture Notes in Computer Science (LNCS)*, page 604–628. Springer, 2020.
109. Sriram Sankaranarayanan, Thao Dang, and Franjo Ivančić. Symbolic model checking of hybrid systems using template polyhedra. In *TACAS*, volume 4963 of *LNCS*, pages 188–202. Springer, 2008.
110. Sriram Sankaranarayanan, Henny Sipma, and Zohar Manna. Constructing invariants for hybrid systems. *Formal Methods in System Design*, 32(1):25–55, 2008.
111. Hussein Sibai, Navid Mokhlesi, Chuchu Fan, and Sayan Mitra. Multi-agent safety verification using symmetry transformations. In *TACAS*, volume 12078 of *Lecture Notes in Computer Science*, pages 173–190. Springer, 2020.
112. Bruno I. Silva, K. Richeson, Bruce H. Krogh, and Alongkrit Chutinan. Modeling and verification of hybrid dynamical system using checkmate. In *ADPM 2000*, 2000. available online from http://www.ece.cmu.edu/~webk/checkmate.
113. Xiaowu Sun, Haitham Khedr, and Yasser Shoukry. Formal verification of neural network controlled autonomous systems. In *HSCC*, pages 147–156. ACM, 2019.
114. Paulo Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.
115. Ashish Tiwari and Gaurav Khanna. Non-linear systems: Approximating reach sets. In *HSCC*, volume 2993 of *Lecture Notes in Computer Science*, pages 477–492. Springer, 2004.
116. Hoang-Dung Tran, Xiaodong Yang, Diego Manzanas Lopez, Patrick Musau, Luan Viet Nguyen, Weiming Xiang, Stanley Bak, and Taylor T. Johnson. NNV: the neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In *Proc. of CAV'20*, volume 12224 of *LNCS*, pages 3–17. Springer, 2020.

117. Willem Visser, Klaus Havelund, Guillaume Brat, SeungJoon Park, and Flavio Lerda. Model Checking Programs. *Automated Software Engineering*, 10(2):203–232, Apr 2003.
118. Vladimeros Vladimerou, Pavithra Prabhakar, Mahesh Viswanathan, and Geir Dullerud. Stormed hybrid systems. In *ICALP*, number PART 2 in Lecture Notes in Computer Science, pages 136–147, 2008.
119. Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. Formal security analysis of neural networks using symbolic intervals. In *Proc. of USENIX Security 2018*, pages 1599–1614. USENIX Association, 2018.
120. Feng Zhao. *Automatic Analysis and Synthesis of Controllers for Dynamical Systems Based on Phase-Space Knowledge.* PhD thesis, 1998.
121. Aditya Zutshi, Sriram Sankaranarayanan, Jyotirmoy Deshmukh, and Xiaoqing Jin. Symbolic-numeric reachability analysis of closed-loop control software. In *Hybrid Systems: Computation and Control (HSCC)*, pages 135–144. ACM Press, 2016.