Interconnected Automata and Linear Systems: A Theoretical Framework in Discrete-Time

Eduardo D. Sontag[⋆]

Department of Mathematics
Rutgers University
New Brunswick, NJ 08903, USA
sontag@control.rutgers.edu

Abstract. This paper summarizes the definitions and several of the main results of an approach to hybrid systems, which combines finite automata and linear systems, developed by the author in the early 1980s. Some related more recent results are briefly mentioned as well.

1 Introduction - The Need for Hybrid Systems

Linear control theory is well-developed and highly sophisticated, and is widely applied in areas ranging from aerospace to automotive control. Linear systems provide highly accurate models of many physical systems; furthermore, the use of linear systems as "robust" controllers often allows the tolerance of even severe model nonlinearities and uncertainties. However, it remains a fact that many continuous physical processes cannot be satisfactorily modeled linearly, nor can be adequately regulated by means of linear controllers alone.

It has long been recognized that the control of more complex systems than those handled by the linear theory will require *switching mechanisms* (discontinuities) of various types; an early discussion of when such discontinuities are unavoidable can be found in [12]; see also the textbook [10], Section 4.8. Thus it is necessary to study hybrid designs in which the *controller* incorporates switching as well as linear elements.

As a parallel and independent development, the spread of consumer electronics has made relevant the control of devices which themselves include logical elements; in this context, it is essential to understand the *modeling* of mixed linear/switched mechanisms.

Automata theory and related areas of computer science provide a powerful set of tools for studying logical operations. Thus it is natural to attempt to develop a systems theory that combines aspects of automata and linear systems, exploring the capabilities of interconnections of both classes of systems. Given that each subclass in itself is already well-understood, it is interesting to ask how much more powerful such interconnections can be expected to be, and whether a systematic and elegant theoretical unifying framework can be developed.

[⋆] Supported in part by US Air Force Grant F49620-95-1-0101

There are two dual aspects in which the power of hybrid systems may be exhibited in this context:

- As *models of systems to be controlled*: one may expect that –via piecewise-linear approximations– it will be possible to model fairly arbitrary dynamic behavior with high precision.
- As *candidates for controllers:* the integration of logical and arithmetic capabilities allows performing control operations which neither finite automata nor finite-dimensional linear systems can carry out by themselves: finite automata cannot deal with all the information in continuous variables, while linear systems cannot exhibit switching behavior.

In the early 1980s, the author wrote the paper [7], whose purpose was to propose a theoretical foundation for such hybrid systems. This paper provided several basic theorems on representation and control, based on the use of a certain logic formalism together with elementary tools of "piecewise linear algebra" developed in the companion paper [8].

Given the resurgence of interest in hybrid systems, it seems timely to present an expository summary of a few of the main concepts and results in these papers, as well as to mention briefly some later work by the author and collaborators which dealt with the study of some particular subclasses. The presentation is informal; in particular, no proofs are given, since they are readily available from the cited literature. Since the audience for this conference –and surely for the proceedings– is so heterogeneous, many of the explanations are of necessity totally redundant for computer scientists, while many others are absolutely obvious to control theorists –apologies to the respective audiences are offered in advance.

Time Scales

In the context of systems with discontinuities, several technical difficulties arise when using continuous time models, including fundamental questions such as the lack of existence theorems guaranteeing solvability of evolution equations. These problems dissappear when using instead *discrete* time increments. As a matter of fact, because of the use of digital devices, modern control design is largely already turning to the latter. Through the device of sampling (measure the plant at discrete intervals, apply a constant control during the next period), continuous-time physical processes are seen by the regulator as a discrete-time system. Thus we took in [7] the point of view of defining hybrid systems only in discrete-time. Of course, the subject of interactions among components operating at distinct time scales is a challenging and most important area of research, with great practical consequences; on the other hand, the use of a common time scale allows one to focus on system-theoretic and control issues which are somewhat obscured when having to worry about additional technical problems.

Theory, Verification, Design

As already mentioned, the paper [7], as is the case with most papers in control theory, was by and large concerned with theory, as opposed to algorithm design. In general, research in control theory includes three complementary objectives: *theoretical analysis*, verification, and design. The first aspect deals with issues such as: (1) the analysis of potential capabilities of classes of systems, both as models of systems to be controlled (representational power) and as controllers, (2) the derivation of necessary and sufficient conditions characterizing properties such as controllability, (3) the classification of systems under natural equivalence relations (changes of variables, action of "feedback group"), or (4) theorems guaranteeing existence and uniqueness of "internal" black box representations of given "external" behaviors. One of the main issues is the reduction of what are *a priori* infinite-dimensional questions, posed in spaces of sequences or continuous functions (controls) –e.g.: is a system controllable; is a given trajectory optimal?– to finite-dimensional ones which can be expressed in terms of finitely many unknowns –e.g., is the controllability Lie algebra full rank? does the given trajectory satisfy the Euler-Lagrange equations?. Such theoretical analysis is not in principle focused on effective computational techniques; rather, obtaining "finite" characterizations is the first goal. Of course, computational feasibility is a desirable ultimate goal, but the problems of control theory are hard enough even before such issues are considered. This part of control theory has very much a "pure mathematical" flavor.

Regarding the issue of *verification*, at an abstract level, this includes all necessary-conditions results, such as the "verification theorems in optimal control" (the theory of the Hamilton-Jacobi equation and "regular synthesis"), but in the current context one means the search for *computational tests* for properties such as controllability or optimality. Typically, the types of nonlinear systems for which such algorithms have been developed are those for which theoretical analysis is very simple. In general, there are difficult open computational questions regarding the computational implementation of abstract mathematical characterizations of most systems properties. The topic of *design* includes the development of computer-aided tools. Linear systems theory has been extremely successful in the formulation and solution of nontrivial design problems and their practical implementation, but few computationally-friendly classes of nonlinear systems have been identified.

The class of behaviors that can be represented by the systems encompassed by the approach in [7] is extremely large, so it should come as no surprise that many of the basic verification and design objectives are NP-hard (or worse). Nonetheless, the basic theoretical framework is useful even from the computational point of view, since it affords an umbrella under which one can formulate subclasses of systems, and restricted problems, among which computational issues can be compared.

2 Piecewise Linear Systems as a Unifying Model

Recall that linear systems (discrete-time) are described by evolution equations of the type

$$x(t+1) \;=\; Ax(t) + Bu(t) \;, \tag{1}$$

(or, as we summarize by dropping the "t" argument and using "+" to denote a unit time-shift, $x^+ = Ax+Bu$). The state $x(t)$ evolves in some finite-dimensional

Euclidean space $\mathbb{R}^n$, control (or "input") values $u(t)$ are taken in some space $\mathbb{R}^m$, and $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are matrices that define the system dynamics. Often, one adds a measurement or read-out map $y(t) = Cx(t)$ to this basic setup. Models of this type constitute the "bread and butter" of modern control design; their theory is well-understood (see e.g. [10]) and computer aided design packages are widely available.

Piecewise linear (or more precisely, piecewise-affine) systems arise if one has *different affine transitions in different parts of the state space and/or the input-value set*, and each of these pieces is described by linear equalities and inequalities. In addition, one may allow different affine measurement maps in different parts. Linear systems are a particular case (just one region). Multiple regions may appear naturally in many ways, as in the examples which we discuss next.

2.1 Some Motivating Examples

As a first illustration, consider the case in which there are logical decisions on input values. For instance, systems with actuators subject to saturation are quite common in applications: valves that cannot open more than a certain limit, control surfaces in aircraft that cannot be deflected more than a certain angle, and so forth. In such cases, a pure linear model (1) is not appropriate; instead one needs to consider a piecewise linear system of the type:

$$x^+ \; = \; Ax + B\,\mathrm{sat}\,(u) \tag{2}$$

where, setting the saturation levels at ± 1, we write $\mathrm{sat}\,(u_1, \ldots, u_m)$ for the vector whose ith component is u_i if $|u_i| \leq 1$ and $\mathrm{sign}\,(u_i)$ otherwise. (See e.g. [13] and references there, for such saturated-input systems, as well as feedback laws for them which combine saturations and linear transformations –for definiteness, the results in [13] are given for continuous-time systems, but analogous theorems hold in discrete-time.) Dually, it often the case that measurement devices may saturate, in which case it is proper to use $y = \mathrm{sat}\,(Cx)$ instead of linear observations $y = Cx$. (See e.g. [3, 4] for some recent results for such models.)

Alternatively, there might be a specified a hyperplane H in $\mathbb{R}^n$, and six matrices $A_0, A_+, A_-, B_0, B_+, B_-$, so that the state at time $t+1$ is $A_-x + B_-u$, $A_0x + B_0u$, or $A_+x + B_+u$ depending on whether the the current state $x(t)$ is on one side of the hyperplane, the hyperplane itself, or the opposite side respectively.

Yet another possibility is to have transitions that take different linear forms on different regions of the *joint* space of states of controls. For example, it may be the case that transitions that would result in a new state which falls in a certain set S are disallowed; in that case a special value, say $x = x_0$, which may indicate a flagged condition, should be produced. We model this situation by the transitions: $x^+ = Ax + Bu$ if $(x, u) \notin S$, and $x^+ = x_0$ if $(x, u) \in S$. A variation is that case in which underflows and overflows of state variables are truncated; this is represented (again taking max and min levels at ± 1 for simplicity) by an equation of the type

$$x^+ \; = \; \mathrm{sat}\,(Ax + Bu) \; . \tag{3}$$

It corresponds in the current context to systems in which $2n$ hyperplanes in $\mathbb{R}^n \times \mathbb{R}^m$ are given (namely, $A_ix + B_iu = \pm 1$, $i = 1, \ldots, n$, where A_i and B_i are the ith rows of A and B respectively) and transitions are expressed by a different affine map in each of the regions that result. These models are sometimes called "recurrent neural networks," treated for instance in [1, 2, 5, 6]; later we mention some results that hold specifically for this class.

Finally, an even further enrichment of the model is obtained if we allow, as part of the specification of the system, the addition of explicit constraints on controls and states. Thus, the state space and control-value sets are taken to be subsets $\mathcal{X}$ and $\mathcal{U}$ of $\mathbb{R}^n$ and $\mathbb{R}^m$ respectively, which indicate *a priori* restrictions on the allowed ranges of variables. To make the theory stay "piecewise linear", we ask that these sets be definable in terms of a finite number of linear equalities and inequalities. These are called *piecewise linear* (from now on, "PL" for short) sets in [7, 8] (a more formal definition is given below). Note that, in particular, all finite subsets of an Euclidean space, and all its linear subspaces, are PL sets. Given a PL set Z, a map $P : Z \to X$ into another PL set is said to be piecewise linear if there is a partition of Z into finitely many PL subsets $Z_1, \ldots, Z_k$ so that the restriction of P to each of the subsets Z_i is an affine map.

2.2 The Formal Setup

Thus a *piecewise linear system* is specified by a triple $(\mathcal{X}, \mathcal{U}, P)$ consisting of two PL sets $\mathcal{X}$ and $\mathcal{U}$ (the state space and control-value set, respectively) and a PL map $P : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$. Dynamically, one interprets such a system as describing a recursion

$$x^+ \;=\; P(x,u) \tag{4}$$

for updating states on the basis of previous states and inputs. A PL measurement map $\mathcal{X} \to \mathcal{Y}$ into a set of possible output values is incorporated into the model if restrictions on observations need to be taken into account. We emphasize again that linear systems are a particular example of this concept. Finite automata are, too: we may identify the states of any given automaton with a finite set of integers $\{1, \ldots, k\}$, and the possible input values with a set $\{1, \ldots, \ell\}$; each of these two sets is a PL set (as a subset of $\mathbb{R}$) and the transition map of the automaton can be represented as a PL map. Later we remark why arbitrary interconnections of linear systems and automata also give rise to PL systems.

The definition of PL sets and PL maps in terms of partitions is cumbersome and unnecessarily complicated from a mathematical point of view. A far simpler but equivalent definition is as follows: The *PL subsets* of $\mathbb{R}^n$ are those belonging to the smallest Boolean algebra that contains all the open halfspaces of $\mathbb{R}^n$. A map $f : X \to Y$ between two PL subsets X and Y of $\mathbb{R}^a$ and $\mathbb{R}^b$ respectively, is a *PL map* if its graph is a PL subset of $\mathbb{R}^a \times \mathbb{R}^b$. (It is not hard to show that these definitions are equivalent to the informal definitions given earlier.)

By a *PL set* one means a PL subset of some $\mathbb{R}^n$; it is obvious how to define PL subsets of PL sets, isomorphisms of PL sets, etc; several "category theoretic" aspects of PL algebra are covered in [7].

It is useful at this point to introduce the first order theory of the real numbers with addition and order. That is, we take the first-order language L consisting of constants r and unary functions symbols $r(\cdot)$, for each real number r (the latter corresponding to "multiplication by the constant r"), as well as binary function symbol $+$ and relation symbols $>$ and $=$. A basic fact is that a quantifier elimination theorem holds: *every set defined by a formula in L is a PL set*. That is to say, for any formula $\Phi(x)$ with n free variables $x = x_1, \ldots, x_n$, the set $\{x \,|\, \Phi(x)\}$ is a PL set. (Of course, we can enlarge the language by adding symbols for sets and maps already known to be PL.) This fact is very simple to establish (see e.g. [7]) and it provides a very convenient tool for establishing the basic theoretical properties of PL systems. Moreover, the proofs of these facts are constructive, in that the actual quantifier algorithm could be in principle used to compute feedback laws and the like.

Another constructively-proved fact is the following "global implicit function theorem" which can also be found in [7]): Assume that $\phi : X \times Y \to \mathbb{R}^n$ is a PL map, and assume that for each x the equation $\phi(x, y) = 0$ can be solved for y. Then there is a PL map $\pi : X \to Y$ so that $\phi(x, \pi(x)) = 0$ for all x. (Equivalently: for any PL subset $R \subseteq X \times Y$ with onto projection into X, there is a PL map $\pi : X \to Y$ (a "section") so that $(x, \phi(x)) \in R$ for all $x \in X$.) This fact is central to the existence of feedback controllers.

The main results in [8] regard the classification of PL sets (from which we may deduce, in turn, classification properties of PL systems); We describe them very briefly in this paragraph, which can be skipped without loss of continuity. Two PL sets X and Y are said to be isomorphic if there is a PL map $\phi : X \to Y$ which is one-to-one and onto (or, equivalently, since the graph of the inverse is the transpose of the original graph, ϕ has a PL inverse). Identifying isomorphic PL sets, the class of all such sets turns out to be endowed with a natural structure of semiring, which is isomorphic to the quotient of $\mathbb{N}[x, y]$ (polynomials in two variables with nonnegative integer coefficients) by the smallest congruence that includes the equations $x = 2x + 1$, $y^2 = 2y^2 + y$, and $y = x + y + 1$. This provides a characterization of the Grothendieck group of the category, as well as a generalization of the Euler characteristic for polyhedra. Moreover, it provides an algorithm for deciding if two PL sets (given in terms of formulas in L) are isomorphic, via results on decidability of word problems and results of Eilenberg and Schützenberger on finitely generated commutative monoids.

2.3 Interconnections

We mentioned above that arbitrary interconnections of linear systems and finite automata can be modeled by PL systems. This is quite obvious, but it is worth sketching a proof simply as an illustration of the use of the formalism afforded by the language L.

Assume given an automaton with finite state space Q and input-value space $T = \{t_1, \ldots, t_{|T|}\}$, also a finite set, and transition function $\delta : Q \times T \to Q$. We consider the case where the state q of the automaton is used to switch among $|Q|$ possible linear dynamics:

$$\begin{aligned} x^+ &= A_q x + B_q u + c_q \\ q^+ &= \delta(q, h(x, u)) \end{aligned}$$

where $A_1, \ldots, A_q$ are matrices of size $n \times n$, $B_1, \ldots, B_q$ are matrices of size $n \times m$, and $c_1, \ldots, c_q$ are n-vectors, and where $h : \mathbb{R}^n \times \mathbb{R}^m \to T$ is a PL map (representing quantized observations of the linear systems). (As before, we are not displaying time arguments; for instance, in the first equation we mean $x(t+1) = A_{q(t)}x(t) + B_{q(t)}u(t) + c_{q(t)}$.) In order to represent this as a PL system, we first identify Q with the set of integers $\{1, \ldots, |Q|\}$. Then the system just described is a PL system with states in the PL subset $\mathbb{R}^n \times \{1, \ldots, |Q|\}$ of $\mathbb{R}^{n+1}$. To show this fact, we need to see that the update equation can be defined using the language L. Indeed, (x, q, u, x^+, q^+) belongs to the graph if and only if it belongs to one of the sets $F_{ij}(x, q, u, x^+, q^+)$, for some $i, j = 1, \ldots, |Q|$, where each such set is described by the sentence:

$$(q = i) \ \& \ \Phi_j(x, u) \ \& \ \big(x^+ - A_i x - B_i u - c_i = 0\big) \ \& \ \Delta_j(i, q^+) \ ,$$

where $\Phi_j(x, u)$ is the property "$h(x, u) = t_j$" (described by a PL map), and Δ_j is characteristic function of the set of pairs so that $\delta(i, t_j) = k$.

Conversely, any PL system can be written as an interconnection of the above form, if the state space is $\mathbb{R}^n$ and the input set is $\mathbb{R}^m$. Indeed, assume that the original transitions have the form $x^+ = A_i x + B_i u + c_i$ if $x \in L_i$, for a given partition into k PL sets L_i. Then we may view these equations as those of a switched system, letting $Q := \{1, \ldots, k\}$ and using $h(x, u) := (j_1, \ldots, j_k)$ where for each i, j_i is the index of that set L_{j_i} for which $A_i x + B_i u + c_i \in L_{j_i}$; the update equation for the finite states is then given by $\delta(i, (j_1, \ldots, j_k)) = j_i$.

3 A Summary of Results From [7]

We now briefly summarize, in informal terms, some of the main results of the basic paper [7]. For reasons of space we must omit most precise definitions and statements, which can be found in that reference, and concentrate instead on providing the main intuitive ideas.

3.1 Finite-Time Problems

The first part of the paper covers topics that are extremely simple, at least once that the basic PL setup has been developed. It concerns problems that can be theoretically solved by simple application of the elimination of quantifiers and "implicit function" results mentioned above. These are *finite horizon* problems, in which a fixed time interval is considered.

A typical problem of this type, and its solution, are as follows. For a fixed but arbitrary time interval $[0, T]$, we wish to decide if every initial state at time $t = 0$ can be controlled to a desired target state $x(T) = x^*$ by a suitable application of controls in (4), and, if the answer is positive, whether there is any feedback law $K : \mathcal{X} \to \mathcal{U}$ so that, for any initial state $x(0) = x_0$, the recursive solution of the closed-loop equations

$$x^+ \ = \ P(x, K(x))$$

results in $x(T) = x^*$. Such feedback designs are of central interest in control theory, since they have obvious error-correction (noise tolerance) properties.

In the current context, when the system is a PL system, checking the property of controllability amounts to checking the truth of a sentence in L (namely, "for all x_0 there exist $u_0, \ldots, u_{T-1}$ so that (solving recursively) there results $x(T) = x^*$" where the "solving recursively" statement can obviously be written as a formula in L, using iterated compositions of P. Of course, the complexity of the formula increases exponentially as a function of the time horizon T, but at least in principle this reasoning shows that the problem is decidable and suggests an algorithm.

More interestingly perhaps, the following theorem holds: if the system is controllable to x^*, then there is a feedback controller $K(x)$ as above (the converse is obviously also true). The construction of K is by means of a straightforward dynamic programming argument, using the "implicit function" result at each step. Thus PL systems are a very well-behaved class from a theoretical point of view, much as linear systems are: if the controllability problem can be solved at all, it can also be solved by feedback (definable within the class being considered, namely PL maps). This is a desirable property, which fails for other reasonable general classes of nonlinear systems (e.g. polynomial or analytic transitions).

Many other problems can be posed and analyzed in an analogous fashion. Among the ones treated in [7] are the existence of observers (state estimators or "filters"), stabilization using dynamical controllers, and systems inverses. We omit details, due to lack of space.

3.2 Asymptotic Problems

More interesting than finite-horizon problems are questions involving infinite-time, and in particular asymptotic, behavior. Most of the results in [7] are in connection with such issues. We discuss now a representative result of this type.

Assume that a system to be controlled (the "plant") is described by

$$\frac{dz}{dt} = f(z(t), v(t)), \tag{5}$$

that is, the state $z(t) \in \mathbb{R}^n$ satisfies the set of first order differential equations specified by the coordinates of the vector function f and $v(t) \in \mathbb{R}^m$ is the control applied at time $t \in [0, \infty)$. (Technical assumptions will be made more precise after we describe the intuitive ideas.) In addition, there is given a function h from states to outputs $y(t) \in \mathbb{R}^p$ which indicates which measurements are available to a controller at time t:

$$y(t) = h(z(t)). \tag{6}$$

The question is whether it is possible to stabilize the system (5), to a desired equilibrium state z^* (without loss of generality, we assume that $z^* = 0$, and $f(0,0) = 0$, $h(0) = 0$), while subject to the constraint of using only the information provided by the output measurements (6). More specifically, we wish to know what intrinsic properties of the system guarantee that there is a *PL* system that can stabilize the system in closed loop.

Since PL systems are defined only in discrete-time, it is first necessary to clarify how one uses a PL system (4) in order to control a continuous-time plant. The meaning is the standard one in control theory: one uses *sample and hold*. Assume a sampling period $\delta > 0$ has been picked. At each sampling instant

$k = 0, \delta, 2\delta, \ldots$, the following events take place: (1) the output $y(k)$ is measured, (2) the discrete-time PL system makes a transition into the next state specified by its update equation (4) (where the input "u" at time k is $y(k)$, so $\mathcal{U} = \mathbb{R}^p$), and (3) a constant control signal $v(t) \equiv v_k$ is applied to the plant during the next inter-sampling interval $(k\delta, (k+1)\delta)$, according to a fixed feedback rule $v_k = K(x(k), y(k))$ which is a PL function of the current state $x(k)$ of the controller and the current observation. (See e.g. [10] for a definitions, examples, and elementary properties of sampling.)

Next we explain the meaning of closed-loop stabilization. For the purposes of this short exposition, we will assume that an arbitrary but fixed compact subset $\mathcal{Z}$ of the state space $\mathbb{R}^n$ has been given, and stabilization means that, for some fixed initial state x_0 of the PL controller, every trajectory of the system (5) obtained by the procedure just sketched, starting from any $z(0) \in \mathcal{Z}$, is such that $z(t)$ is well-defined for all $t \geq 0$ and $\lim_{t\to\infty} z(t) = 0$. (A stronger Lyapunov stability-like property can be required –see [7]– namely that if $z(0)$ is small then the obtained trajectory should remain small.)

Observe that for there to exist any type of stabilizing controller, PL or not, it is necessary that the system (5) be *null-asymptotically controllable* (n.a.c.) when starting from the subset $\mathcal{Z}$: for each $z(0) \in \mathcal{Z}$, there is some control function $v(\cdot)$ so that the solution of (5) converges to zero. (Proof: look at the $v(t)$ produced by a controller, if one exists.) Theorem 3.11 in [7] then states that conversely, under assumptions that are quite mild in the context of nonlinear control, a PL controller exists if the plant has the n.a.c. property.

We describe these assumptions now (in a stronger form than needed, so as to make the discussion concise). The first is that the mappings $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and $h : \mathbb{R}^n \to \mathbb{R}^p$ are real-analytic (admitting locally convergent power series representations around each point in their domains). It is important to note that, while including usual descriptions of mechanical systems (which are obtained by combining trigonometric, polynomial, and other analytic functions), this assumption does rule out switching behavior in the system itself, or even infinitely differentiable but non-analytic nonlinearities. The second assumption is that the Jacobians of the maps f and h at the origin are well-behaved: if A, B, and C are the matrices for which $f(x,u) = Ax + Bu + o(x,u)$ and $h(x) = Cx + o(x)$, then the linearized system $\dot{z} = Az + Bv$, $y = Cz$ is stabilizable and detectable (these constitute generically satisfied rank conditions on the triple of matrices (A, B, C); see [10]). This hypothesis is essential for the proof (it is in fact also necessary for the existence of a controller if exponential convergence is required). Finally, we assume that the plant is *observable*. This means (cf. [10]) that given any two states z_1 and z_2, there is some control function (which depends on the particular pair (z_1, z_2)) so that when applied to the system, different measurement signals $y(t)$ result when starting at z_1 or at z_2. This hypothesis can be relaxed considerably, and can be replaced by a condition which is necessary if a controller exists (a "nonlinear detectability" condition).

The proof of Theorem 3.11 provides a PL controller whose state space consists of a cartesian product of an Euclidean space and a finite set, and whose dynamics are described in terms of "if-then-else" linear equality and inequality decisions and linear operations. Usually a stronger property for the controller is desirable, namely that convergence to zero occur for every initial state of the controller. This means, in a practical sense, that a sudden and unobserved state change in the plant, due to noise or unmodeled inputs, will not affect convergence (since the controller, starting from the state at the given time, still regulates the plant). This stronger property is called "strong regulation" in [7], and the main theorem there, valid under slightly stronger observability assumptions on the system, assures such regulation by PL systems (cf. Theorem 3.15).

4 Computational Complexity

We conclude with some remarks about computational issues. As remarked earlier, finite-horizon problems are decidable for PL systems. Thus it is of interest to study their computational complexity. Unfortunately, there is a rather negative result in that regard. To explain this result, given in [9] and not difficult to establish, we first recall some basic concepts from logic. Given a fixed piecewise-linear system, a fixed time horizon, and a pair of initial and target states x_0 and x^* respectively, asking if there is any control which steers x_0 to x^* is a purely-existential problem, or a "$\exists$" problem, for the language of piecewise linear algebra, because it is possible to write a logical formula of the type "there exists u so that $\Phi(u)$" which is true if and only if the property holds (and Φ does not involve any free variables besides the components of u which represent the control sequence; Φ is simply the sentence that asserts that the composition of the dynamics T times, using this control, lands the state at x^* when starting from x_0). Other (still finite-horizon) problems in control are not formulated originally in $\exists$ form. For instance, to ask if the whole system is controllable to x^* in T steps would require a formula of $\forall\exists$ type, namely a formula that reads "for all x there exists u such that $\Phi(x,u)$" whose truth is equivalent to the desired controllability (and now Φ has the coordinates of the initial state x represented by a set of variables, as well as the control). Another variant appears in design problems. For instance, given a parametric form for a closed-loop controller, say $P(\lambda)$, asking that some value of the parameter result in a feedback law which controls each state to zero in T steps would be given by an $\exists\forall$ formula ("there is some parameter λ so that, for each initial state state x, $\Phi(x,u)$"). Even more alternations of quantifiers might appear. For example, in the context of "control Lyapunov functions" one might ask whether there is a value for a parameter λ so that a scalar "energy" function $V_\lambda(x)$ decreases along suitable trajectories, giving rise to a $\exists\forall\exists$ formula ("there is a λ so that, for each x, there is some u so that either $x = 0$ or $V_\lambda(P(x,u)) < V_\lambda(x)$"). In the same manner, one can define of course $\exists\forall\ldots\exists$ types of problems, for all finite sequences of quantifiers.

Roughly stated, the "polynomial hierarchy" in logic and computer science is obtained in this same way when the basic quantifier-free formulas Φ are propositional formulas, and the variables over which one quantifies are Boolean-valued. Problems are in the class NP (non-deterministic polynomial time) if they can be described by just $\exists$ formulas, and in P (polynomial time) if they can be described with no quantifiers at all. It is widely believed, and one of the most important open problems in theoretical computer science to prove, that the various levels are very different in complexity. Thus, not only should P be different from NP, but problems whose definition requires $\forall\exists$ should be much harder to solve than those in NP, and so forth going "up" along the hierarchy.

The main result in [9] was that problems in any given level, such as for instance $\exists\forall\exists$, for PL systems are of exactly *same complexity* (in a precise sense of reduction of one problem to another) as problems in the corresponding level of the polynomial hierarchy. Thus one has a complete understanding of complexity for such problems modulo the same understanding for the classical hierarchy, including decidability in polynomial space, and a rich theory of complexity when using parallel computing.

The situation is radically different for *infinite* horizon problems, such as asking if a system is controllable (in some finite but not prespecified number of steps). Obviously, such problems will be in general undecidable, as it is easy to encode a Turing machine halting problem into PL behavior. On the other hand, it is perhaps surprising that even for "mildly" PL systems such as those given by an Equation as in (3), undecidability holds, as we discuss next.

4.1 A Special Subclass: Saturated Transitions

A special class of PL systems is that modeled by the saturated-transition systems ("recurrent neural networks") of the type displayed in Equation (3). Models like this are of interest for several different reasons, including their computational universality (discussed below), approximation properties (cf. [11]), and use in experimental "neural network" work; they arise naturally when linear systems have variables subject to amplitude limitations. One might think that control problems for such systems, being so close to linear systems, and appearing so often in the literature, may be simpler to solve than problems for more complicated classes of PL systems. We show next, through a simple controllability question, that this is quite far from being true.

We call a state ξ *null-controllable* if there is some some nonnegative integer k and some input sequence $u(0), \ldots, u(k-1)$ which steers ξ to $x(k) = 0$. This is as considered earlier, except that now we are not assuming that the time $k = T$ has been fixed in advance. Before proceeding further with this class, note for purposes of comparison that if there would be no saturation, we would be studying the standard class of linear systems (1), and for linear systems one can determine null-controllability of a state ξ in a computationally simple manner. Indeed, ξ is null-controllable for (1) if and only if the null-controllability property is verified with $k = n$ (this is a standard elementary fact; see for instance Lemma 3.2.8 in [10]); thus a state ξ is null-controllable if and only if $A^n\xi$ is in the the reachability space of (1), that is, the span of the columns of $B, AB, \ldots, A^{n-1}B$. This property can in turn be checked by Gaussian elimination, so it can be verified in a number of algebraic operations that is polynomial in n and m ("strong polynomial time"). Alternatively, we may ask the question of null-controllability in a bit-computational (Turing-machine) model, assuming that the entries of the matrices A and B, as well as the coordinates of the state ξ, and all rational (as opposed to arbitrary real) numbers, and are each given by specifying pairs of integers in a binary basis. Then the fact is that null-controllability of a state ξ for the system (1) can be checked in a number of elementary Turning-machine steps which is polynomial in the size of the input data, that is, the total number of bits needed to specify A, B, ξ. Thus, the problem is in the class "P" of polynomial-time computable problems. (From now on, we use the Turing machine model, to stay close to classical computational complexity.)

Thus it is natural to ask if adding a saturation can change matters in a fundamental way. The answer is yes. In fact, the change is as big as it could be: *For saturated linear systems (3), the null-controllability question is recursively unsolvable.*

In other words, there is no possible computer program which, when given A, B, ξ with rational entries, can answer after a finite amount of time "yes" if the state ξ is null-controllable for the corresponding system, and "no" otherwise. (In particular, there is no possible characterization in terms of rank conditions, such as was available for linear systems, nor any characterization in terms of checking higher-order algebraic conditions in terms of polynomials constructible from the entries of the matrices and vector in question.) The proof of this fact relies upon the work on simulation of Turing machines by devices such as (3); see [5]. From that work it follows that there exists a certain matrix A (with n approximately equal to 1000 in the construction given in [5], and most entries being 0, 1, or certain small rational numbers) for which there is no possible algorithm that can answer the following question: "Given ξ, is there any integer k so that the first coordinate of the solution of

$$x(t+1) = \operatorname{sat}(Ax(t)) \ , \quad x(0) = \xi \tag{7}$$

has $x_1(k) = 1$?" (Of course, (7) is a particular case of (3), when $B = 0$.) Moreover, the matrix A is built in such a manner that the above property is impossible to check even if ξ is restricted to be a vector with the property that the solution of (7) has $x_1(t) \in \{0, 1\}$ for all $t = 0, 1, \ldots$. It is easy to convert the problem "is $x_1(k) = 1$ for some k?" to "is $x(k) = 0$ for some k?" simply by changing each coordinate update equation $x_i(t+1) = \operatorname{sat}(\ldots)$ to $x_i(t+1) = \operatorname{sat}(\ldots - \alpha x_1(t))$, where α is a positive integer bigger than the possible maximum magnitude of the expression "$\ldots$". While $x_1(t) = 0$ nothing changes, but if x_1 ever attains the value 1 then the next state is $x = 0$. So the null-controllability question is also undecidable, even in the case in which the system is this one particular system of dimension about 1000 (which in the proof corresponds to a simulation of a universal Turing machine, with the initial condition ξ corresponding to the program for such a machine). This negative result shows that adding a saturation has changed the problem dramatically from the linear case.

One may of course ask about related problems such as observability. For instance, given a system (3) and a linear output map $y = Cx$, one may ask for the decidability of the problem, for a given state ξ: "is ξ indistinguishable from 0?" Again this is essentially trivial for linear systems (just check if ξ is in the kernel of the Kalman observability matrix), but the problem becomes undecidable for saturated systems (take $Cx := x_1$ and use the above construction; as $Cx(t) = x_1(t)$ is always zero or one, distinguishability from zero is equivalent to determining if it is ever one).

While on the topic of the systems of type (3), we should point out that when real (as opposed to merely rational) coefficients are allowed for the matrices A and B, and the initial state, it is possible to formulate precisely the question of determining the computational power of such devices. The, perhaps surprising, answer, is that they are computationally no less powerful than essentially arbitrary continuous discrete-time systems (up to polynomial time speedups). This makes such models of PL systems a universal model for "real number" computation. Moreover, their capabilities can be understood in the context of "Turing machines that consult sparse oracles," in the language of computational complexity; the reader is referred to [6] for this topic.

References

1. Albertini, F., and E.D. Sontag, "Identifiability of discrete-time neural networks," *Proc. European Control Conference*, Groningen, June 1993, pp. 460-465.
2. Albertini, F., and E.D. Sontag, "State observability in recurrent neural networks," *Systems & Control Letters* **22**(1994): 235-244.
3. Koplon, R.M., L.J. Hautus, and E.D. Sontag, "Observability of linear systems with saturated outputs," *Lin. Alg. Appl.* **205-206**(1994): 909-936.
4. Koplon, R.M. and E.D. Sontag, "Sign-linear systems as cascades of automata and continuous variable systems," *Proc. IEEE Conf. Decision and Control, San Antonio, Dec. 1993*, IEEE Publications, 1993, pp. 2290-2291.
5. Siegelmann, H.T., and E.D. Sontag, "On the computational power of neural nets," *J. Comp. Syst. Sci.* **50**(1995): 132-150.
6. Siegelmann, H.T., and E.D. Sontag, "Analog computation, neural networks, and circuits," *Theor. Comp. Sci.* **131**(1994): 331-360.
7. Sontag, E.D., "Nonlinear regulation: The piecewise linear approach," *IEEE Trans. Autom. Control* **AC-26**(1981): 346-358.
8. Sontag, E.D., "Remarks on piecewise-linear algebra," *Pacific J.Math.*, **98**(1982): 183-201.
9. Sontag, E.D., "Real addition and the polynomial hierarchy," *Inform. Proc. Letters* **20**(1985): 115-120.
10. Sontag, E.D., *Mathematical Control Theory: Deterministic Finite Dimensional Systems*, Springer, New York, 1990.
11. Sontag, E.D., "Systems combining linearity and saturations, and relations to "neural nets," in *Proc. Nonlinear Control Systems Design Symp., Bordeaux, June 1992* (M. Fliess, Ed.), IFAC Publications, pp. 242-247.
12. Sontag, E.D., and H.J. Sussmann, "Remarks on continuous feedback," *Proc. IEEE Conf. Decision and Control, Albuquerque, Dec.1980*, pp. 916-921.
13. Sussmann, H.J., E. Sontag, and Y. Yang, "A general result on the stabilization of linear systems using bounded controls," *IEEE Trans. Autom. Control* **39**(1994): 2411-2425.

This article was processed using the LaTeX macro package with LLNCS style