On some Relations between Dynamical Systems and Transition Systems*

Eugene Asarin[1] and Oded Maler[2]

[1] Institute for Information Transmission Problems, 19 Ermolovoy st., Moscow, Russia, asarin@ippi.msk.su

[2] Spectre – Verimag, Miniparc-zirst, 38330 Montbonnot, France, Oded.Maler@imag.fr

Abstract. In this paper we define a precise notion of abstraction relation between continuous dynamical systems and discrete state-transition systems. Our main result states that every Turing Machine can be realized by a dynamical system with piecewise-constant derivatives in a 3-dimensional space and thus the reachability problem for such systems is undecidable for 3 dimensions. A decision procedure for 2-dimensional systems has been recently reported by Maler and Pnueli. On the other hand we show that some non-deterministic finite automata cannot be realized by any continuous dynamical system with less than 3 dimensions.

1 Introduction

There has been recently an increasing interest in models of *hybrid systems*, i.e., systems that combine intercommunicating discrete and continuous components (see [9], [12], [3]). The introduction of these models is motivated by a real practical concern: more and more computers (discrete transition systems) are nowadays embedded within real-world control loops such as in avionics, process control, robotics and consumer products – to mention a few application areas. The analysis and prediction of the combined behavior of these embedded systems require formal tools that cut across existing disciplinary boundaries: the real-world is usually modeled by control engineers as a continuous dynamical system while computer scientists investigate the dynamics of discrete systems.

This line of research raises interesting theoretical problems concerning the relations between these two types of dynamics, and concerning the applicability of various techniques, originating in program verification, to the automatic analysis of dynamical systems. In this paper we give a precise definition of abstraction of a continuous dynamical system by a discrete transition system. Then

* This research was supported in part by the France-Israel project for cooperation in Computer Science, by the European Community ESPRIT Basic Research Action Projects REACT (6021) and by Research Grant #93-012-884 of the Russian Foundation of Fundamental Research. Verimag is a joint laboratory of cnrs, inpg, ujf and Verilog sa. Spectre is a project of inria. The final version was prepared while the first author was visiting the Dept. of Mathematics of the University Paris 12 - Val de Marne.

we introduce a class of hybrid systems that has attracted most of the attention within hybrid systems research, namely systems with piecewise-constant derivatives (PCD systems). We show that such systems, even without explicit discrete "jumps" (i.e., the trajectories are continuous but not smooth) can simulate in a reasonable sense *every* effective transition system and hence their reachability problem is in general undecidable. This fact should be compared with the decision procedure introduced in [10] for such systems with 2 dimensions.

The rest of the paper is organized as follows: In section 2 we give the necessary preliminary definitions. In section 3 we show how PCD systems can realize deterministic finite and push-down automata. For non-deterministic automata we show in section 4 that three dimensions are necessary and sufficient for their realization by dynamical systems. In section 5 we show how to realize every Turing machine by a 3-dimensional and thus prove our main undecidability result. Finally we discuss some related work.

2 Preliminaries

Let Q be a set of "states". A state sequence is a function $\sigma : \mathbb{N} \to Q$. A transition system without input is a pair $\mathcal{A} = (Q, \delta)$ where Q is a set of states and $\delta \subseteq Q \times Q$ is a transition relation. We say that $\mathcal{A}$ is deterministic if δ is a function from Q to Q and call $\mathcal{A}$ an automaton when Q is finite. A *run* of such a system starting at some $q \in Q$ is a (finite or infinite) state sequence $\sigma = \sigma[1], \sigma[2], \ldots$ such that $\sigma[1] = q$ and for every $i \geq 1$, $(\sigma[i], \sigma[i+1]) \in \delta$. The set of all possible runs starting at q is denoted by $L(\mathcal{A}, q)$, and the set of possible runs starting at some $q \in Q_0 \subseteq Q$ is denoted by $L(\mathcal{A}, Q_0)$.

Let Q_1 and Q_2 be two sets of states, and let $\psi : Q_1 \to Q_2$ be a surjective (possibly partial) function called the *state-abstraction* function. Then for every sequence $\sigma \in Q_1^* \cup Q_1^\omega$, its abstract image is $\psi(\sigma) = \tau \in Q_2^* \cup Q_2^\omega$ defined by $\tau[i] = \psi(\sigma[j_i])$ where $j_0 = 0$ and for every $i \geq 1$, $j_i = \min\{j : j > j_{i-1} \wedge \psi(\sigma[j]) \neq \perp\}$. Note that if ψ is total than $j_i = i$ and $\tau[i]$ is simply $\psi(\sigma[i])$.

Definition 1 (Abstraction of Transition Systems) *Let $\mathcal{A}_1 = (Q_1, \delta_1)$ and $\mathcal{A}_2 = (Q_2, \delta_2)$ be two transition systems. We say that $\mathcal{A}_2$ is an abstraction of $\mathcal{A}_1$ (or that $\mathcal{A}_1$ realizes $\mathcal{A}_2$) via some function $\psi : Q_1 \to Q_2$ if*

$$(\forall q_2 \in Q_2)(\forall q_1 \in \psi^{-1}(q_2))(\sigma \in L(\mathcal{A}_1, q_1) \Rightarrow \psi(\sigma) \in L(\mathcal{A}_2, q_2)) \qquad (1)$$

and

$$(\forall q_2 \in Q_2)(\forall \sigma_2 \in L(\mathcal{A}_2, q_2))(\exists q_1 \in Q_1)(\exists \sigma_1 \in L(\mathcal{A}_1, q_1)) : \sigma_2 = \psi(\sigma_1) \qquad (2)$$

This fact is denoted by $\mathcal{A}_2 \preceq_\psi \mathcal{A}_1$ or simply $\mathcal{A}_2 \preceq \mathcal{A}_1$.

The first condition says that the concrete system $\mathcal{A}_1$ can produce a sequence only if its abstract image is allowed in the abstract system $\mathcal{A}_2$. The second condition says that every abstract sequence is realized by some concrete sequence.

A *time segment* is any interval $[0, r] \subseteq I\!R^+$ including $I\!R^+$ itself. The dynamical systems we consider are over $X = I\!R^d$ considered as a d-dimensional Euclidean[3] space. In other words we have d state-variables ranging over the reals. Points (or vectors) in X are denoted by boldface letters, e.g., $\mathbf{x}$. An open (closed) half-space is a set of points $\mathbf{x} \in X$ satisfying $\mathbf{a} \cdot \mathbf{x} < b$ ($\mathbf{a} \cdot \mathbf{x} \leq b$). A *convex polyhedral set* is a finite intersection of half-spaces.

A *trajectory* in X is a continuous function $\xi : T \to X$ where T is a time interval. The first step in establishing some connection between trajectories and state-sequences is to map points of X into states.

Definition 2 (Convex State Abstraction) *A convex state abstraction is a partial function $\varphi : X \to Q$ such that for every $q \in Q$, $\varphi^{-1}(q)$ is a convex relatively-open[4] set.*

The φ-abstract trajectory of ξ is a function $\eta : T \to Q \cup \{\perp\}$ defined as $\eta(t) = \varphi(\xi(t))$. After having discretized space we discretize time:

Definition 3 (Signature) *Let ξ be a trajectory, and let η be its φ-abstract trajectory for a state abstraction φ. The (timed) φ-signature of ξ is a sequence $\tau = (q[1], l[1], u[1]), (q[2], l[2], u[2]), \ldots$ where for every i, $\tau[i] = (q[i], (l[i], u[i]))$ iff*

1. $l[i] = \inf\{t > u[i-1] : \eta(t) \neq \perp\} \quad (u[0] = 0)$
2. $u[i] = \inf\{t > l[i] : \eta(t) = \perp\}$
3. $\eta(t) = q[i]$ *for some (and every)* $t \in (l[i], u[i])$

The sequence $\sigma \in Q^* \cup Q^\omega$ obtained by projecting away the temporal intervals is called the (untimed) φ-signature of ξ and is denoted by $\varphi(\xi)$.

The fact that regions do not intersect implies that every state-transition must involve a passage through an undefined region. The generators of trajectories are dynamical systems:

Definition 4 (Dynamical System) *An (autonomous) dynamical system is $\mathcal{H} = (X, f)$ where X is the state-space and f is a partial function from X to X such that $\frac{d^+\mathbf{x}}{dt} = f(\mathbf{x})$ is the differential equation governing the evolution of $\mathbf{x}$. A trajectory of $\mathcal{H}$ starting at some $\mathbf{x}_0 \in X$ is a function $\xi : T \to X$ such that $\xi()$ is a solution of the equation with initial condition $\mathbf{x} = \mathbf{x}_0$, i.e., $\xi(0) = \mathbf{x}_0$ and for every t, $f(\xi(t))$ is defined and is equal to the right derivative of $\xi(t)$.*

We denote the set of all trajectories starting at some $\mathbf{x} \in X$ by $M(\mathcal{H}, \mathbf{x})$. A differential equation has a uniqueness property if for every $\mathbf{x}$ there is at most one solution. In this case the system is said to be deterministic.

---

[3] We assume familiarity with elementary topological and geometrical notions, such as connectedness, closure, interior, boundary, relative boundary and convexity – see [5] for a readable introduction.

[4] A set R is relatively-open if every point has a neighborhood such that its intersection with the affine hull of R is contained in R. Points and open intervals, for example, are relatively-open but not open in a d-dimensional space, $d \geq 2$.

Having defined dynamical systems and abstractions of trajectories into sequences we can define the proper notion of abstraction between dynamical systems and transition systems.

Definition 5 ($\mathcal{A} \preceq_\varphi \mathcal{H}$) *A transition system $\mathcal{A}$ is an abstraction of a dynamical system $\mathcal{H}$ ($\mathcal{H}$ realizes $\mathcal{A}$) via a state-abstraction φ if*

$$(\forall q \in Q)(\forall \mathbf{x} \in \varphi^{-1}(q))(\xi \in M(\mathcal{H}, \mathbf{x}) \Rightarrow \varphi(\xi) \in L(\mathcal{A}, q)) \tag{3}$$

and

$$(\forall q \in Q)(\forall \sigma \in L(\mathcal{A}, q))(\exists \mathbf{x} \in \varphi^{-1}(q))(\exists \xi \in M(\mathcal{H}, \mathbf{x})) : \sigma = \varphi(\xi) \tag{4}$$

This fact is denoted by $\mathcal{A} \preceq_\varphi \mathcal{H}$.

The first condition says that *the signature of every trajectory of $\mathcal{H}$ is a sequence of $\mathcal{A}$*. The second conditions means that *every sequence of $\mathcal{A}$ is a signature of some trajectory in $\mathcal{H}$*. This definition is almost identical to definition 1, except for the fact that here we needed additional machinery for going from a dense time domain to a discrete one.

A variant of dynamical systems we are going to employ are systems with piecewise-constant (and thus discontinuous) derivatives.

Definition 6 (PCD System) *A piecewise-constant derivative (PCD) system is a dynamical system $\mathcal{H} = (X, f)$ where f is a (possibly partial) function from X to X such that the range of f is a finite set of vectors $C \subset X$, and for every $\mathbf{c} \in C$, $f^{-1}(\mathbf{c})$ is a finite union of convex polyhedral sets.*

In other words, a PCD system consists of partitioning the space into convex polyhedral sets ("regions"), and assigning a constant derivative $\mathbf{c}$ ("slope") to all the points sharing the same region. The trajectories of such systems are broken lines, with the breakpoints occurring on the boundaries of the regions. PCD systems are deterministic. An example of a PCD system appears in figure 1.

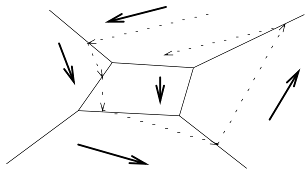

Fig. 1. A 2-dimensional PCD system. The slopes are depicted in dark lines, while the dashed line indicates a sample trajectory.

Given an effective description of dynamical system $\mathcal{H}$, the reachability problem for $\mathcal{H}$, denoted by *Reach*$(\mathcal{H}, \mathbf{x}, \mathbf{x}')$ is the following: *Given $\mathbf{x}, \mathbf{x}' \in X$, are there $\xi \in M(\mathcal{H}, \mathbf{x})$ and $t > 0$ such that $\xi(t) = \mathbf{x}'$?* For PCD systems with 2 variables (described by linear inequalities and slope vectors) this problem has been proven decidable in [10].

3 Realization of Finite and Push-down Automata

First we show how every finite-state deterministic automaton can be realized by a 3-dimensional PCD system (a deterministic finite automaton without input is a rather trivial object and the construction is presented here just because it underlies the more complicated constructions for infinite-state machines).

Claim 1 (Realization of Finite Automata) *For every finite deterministic automaton $\mathcal{A}$ there is a 3-dimensional PCD system $\mathcal{H}$ such that $\mathcal{A} \preceq \mathcal{H}$.*

Proof: Suppose the automaton has n states. The realizing system is defined over the subset $[1,n] \times [0,1] \times [0,n]$ of $I\!R^3$. It consists of the following regions (we call the state variables x, y, and z):

Region	Defining conditions	Derivative
F	$(z=0) \wedge (y<1)$	$\dot{x}=0, \dot{y}=1, \dot{z}=0$
U_{ij}	$(x=i) \wedge (y=1) \wedge (z<j)$	$\dot{x}=0, \dot{y}=0, \dot{z}=1$
B_{ij}	$(z=j) \wedge (x+(j-i)y=j) \wedge (y>0)$	$\dot{x}=j-i, \dot{y}=-1, \dot{z}=0$
D	$(z>0) \wedge (y=0)$	$\dot{x}=0, \dot{y}=0, \dot{z}=-1$

The regions U_{ij} and B_{ij} are defined for every i,j such that $\delta(q_i)=q_j$. As a state-abstraction we take $\varphi(x,y,z)=q_i$ if $(x=i) \wedge (y=z=0)$, and $\varphi(x,y,x)=\perp$ otherwise. An example of this construction appears in figure 2. It can be verified that the system goes from $(i,0,0)$ to $(j,0,0)$ iff $\delta(q_i)=q_j$: At F the system advances y until $(i,1,0)$, then at U_{ij} it goes "up" until it reaches the surface $z=j$ at $(i,1,j)$. On that surface in region B_{ij} it goes diagonally to $(j,0,j)$ and finally in D it goes down to $(j,0,0)$. Note that all the regions leading to $(j,0,j)$ are located on the same surface. ∎

Remark: This technique can be applied to the realization of non-deterministic automata by *non-deterministic* PCD systems. All we have to do is to modify the definition of a PCD system to allow non-determinism on the (relative) boundaries of the regions. Then if both (q_i,q_j) and (q_i,q_k) are possible transitions, $j<k$, the system will bifurcate in $(i,1,j)$ between B_{ij} (going to q_j) and U_{ik} (going up until $z=k$ and then to B_{ik}). But, as we will show later, non-deterministic automata can be realized by *deterministic* PCD systems.

The results concerning infinite-state transition-systems will be based on stack machines. A stack is an element of Σ^ω where $\Sigma=\{0,\ldots,k-1\}$. We define the following two functions: PUSH: $\Sigma \times \Sigma^\omega \rightarrow \Sigma^\omega$ and POP: $\Sigma^\omega \rightarrow \Sigma \times \Sigma^\omega$ as PUSH$(v,S)=v \cdot S$ and POP$(v \cdot S)=(v,S)$.

Definition 7 (PDAs) *A deterministic pushdown-automaton (PDA) is a transition system $\mathcal{A}=(Q \times \Sigma^\omega, \delta)$ for some $Q=\{q_1,\ldots q_n\}$ such that δ is defined using a finite collection of statements of one of the following two forms:*

```
q_i: S :=PUSH(v,S);          q_i: (v,S) :=POP(S);
     GOTO q_j                     IF v = 0 GOTO q_{i_0};
                                  ...
                                  IF v = k-1 GOTO q_{i_{k-1}};
```

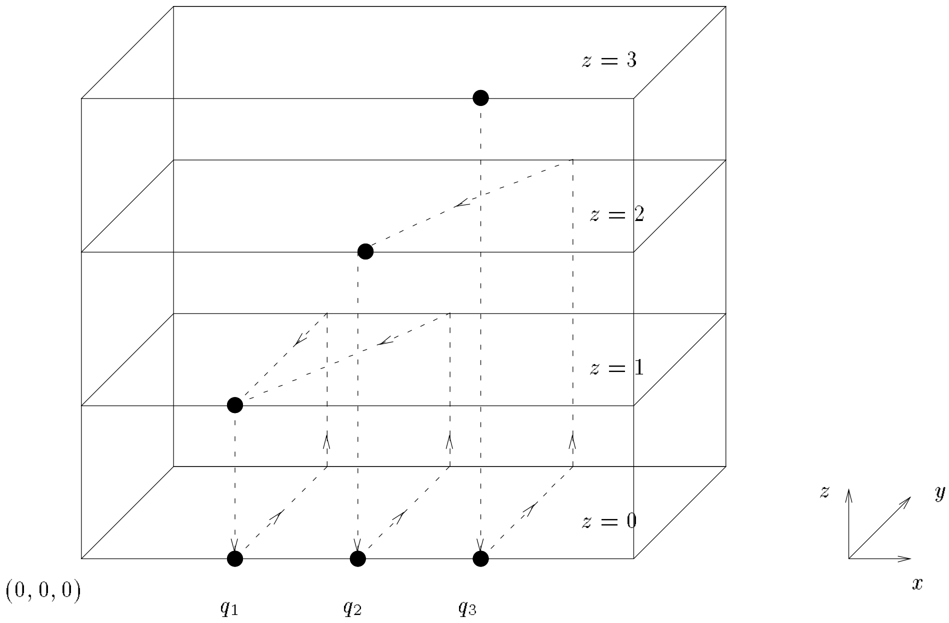

Fig. 2. Simulating a 3-state automaton with $\delta(q_1) = \delta(q_2) = q_1$ and $\delta(q_3) = q_2$.

The contents of a stack is denoted by $S = s_1 s_2 \ldots$ where s_1 is the top of the stack. We define an encoding function $r : \Sigma^\omega \to [0,1]$ as $r(S) = \sum_{i=1}^{\infty} s_i k^{-i}$. We assume that $k-1$ appears only finitely many times in the stack, hence r is injective. It is easily verified that the stack operations have arithmetic counterparts that operate on the representation:

$$S' = \textsc{push}(v,S) \text{ iff } r(S') = (r(S)+v)/(k)$$
$$(S',v) = \textsc{pop}(S) \text{ iff } r(S') = kr(S) - v$$

Claim 2 (Realization of PDAs) *Every PDA can be realized by a* 3*-dimensional PCD system.*

Sketch of Proof: For simplicity we assume $k = 2$ and $\Sigma = \{0,1\}$. Consider the three planar sub-systems depicted in figure 3 and a trajectory segment starting at $\mathbf{x} = (x,0)$, $x \in [0,1]$ and ending at $\mathbf{x}' = (x',1)$. It can be verified that either:

$$\begin{array}{ll} x' = (x+1)/2 & \textsc{push } 1 \\ x' = x/2 & \textsc{push } 0 \\ x' = 2x - 1/2 & \textsc{pop} \end{array}$$

Thus if $x = r(S)$ at the "input port" $(y = 0)$ of a PUSH element, then $x' = r(S')$ at the "output port" $(y = 1)$ of that element where S' is the resulting stack.

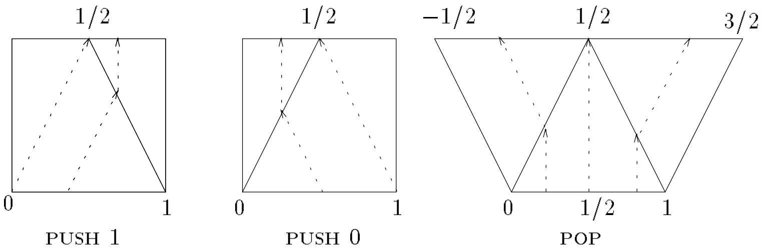

Fig. 3. The basic elements.

For the POP element we have two output ports $-1/2 \leq x < 1/2$ and $1/2 \leq x < 3/2$. If the top of the stack was 0 the trajectory reaches the left port with $x' = r(S') - 1/2$, otherwise it goes to the right port with $x' = r(S') + 1/2$. In both cases the value of x' (relative to the port's "origin") encodes the new content of the stack. Thus all that remains to do is to take for every q_i an element corresponding to its stack operation, place it with the origin in position, say, $(2i, 0, 0)$ and use the third dimension in order to connect the output ports back to the input ports according to the GOTO's (see figure 4). This is similar to the previous construction except for the fact that the connections are via two-dimensional "bands" and thus two families of trajectories going to the same state q_i cannot be merged on the same plane ($z = j$) but only while going "down". Finally the state-abstraction function is defined as $\varphi(x, y, z) = (q_i, S)$ iff $y = z = 0$, $2i \leq x < 2i + 1$ and $S = r^{-1}(x - 2i)$. ∎

4 Realizing Non-Deterministic Automata

In this section we show that every non-deterministic automaton can be realized by a 3-dimensional deterministic PCD system, and that certain non-deterministic automata cannot be realized using less than 3 dimensions.

Claim 3 (Transitivity of Abstraction) *Let $\mathcal{H} = (X, f)$ be a PCD system and let $\mathcal{A}_1 = (Q_1, \delta_1)$, $\mathcal{A}_2 = (Q_2, \delta_2)$ be two transition systems such that $\mathcal{A}_1 \preceq_\varphi \mathcal{H}$, $\mathcal{A}_2 \preceq_\psi \mathcal{A}_1$, and $\psi \circ \varphi : Q_1 \to X$ is convex. Then $A_2 \preceq_{\psi\circ\varphi} \mathcal{H}$.*

Proof: Follows from the definitions. ∎

Claim 4 *Let $\mathcal{A} = (Q, \delta)$ be a non-deterministic automaton. Then there exists a PDA $\mathcal{D} = (Q \times \Sigma^\omega, \Delta)$ and a mapping $\psi : Q \times \Sigma^\omega \to Q$ such that $\mathcal{A} \preceq_\psi \mathcal{D}$.*

Proof: Assume $Q = \{q_1, \ldots, q_k\}$, and let $\Sigma = \{0, \ldots, k\}$. Define Δ as a collection of statements of the form:

$$\begin{array}{l} q_i\colon (v, S) := \text{POP}(S); \\ \quad \text{IF } (q_i, q_v) \in \delta \text{ GOTO } q_v \end{array}$$

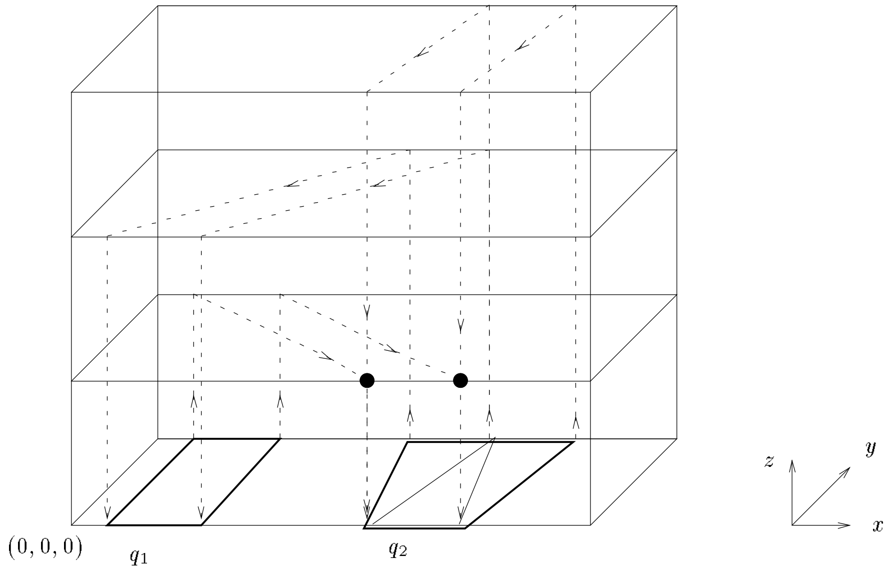

Fig. 4. Simulating a PDA with 2 states, defined by: q_1 : S :=PUSH$(1, S)$; q_2 : (v, S) :=POP(S); If $v = 1$ THEN GOTO q_2 ELSE GOTO q_1. Note the place where the two GOTOs to q_1 merge.

One can see that if $((q_{i_1}, S_1), (q_{i_2}, S_2)) \in \Delta$ then $(q_{i_1}, q_{i_2}) \in \delta$, and that for every run $q_{i_1}, q_{i_2}, \ldots$ of $\mathcal{A}$ there is a stack S_1 such that $(q_{i_1}, S_1), (q_{i_2}, S_2), \ldots$ is a run of $\mathcal{D}$. Hence by letting $\psi(S, q) = q$ for every $q \in Q$ we have $\mathcal{A} \preceq_\psi \mathcal{D}$. ■

Corollary 5 *Every non-deterministic finite automaton can be realized by a 3-dimensional PCD system.*

Remark: The initial stack configuration contains information concerning all the future non-deterministic choices.[5] "Finite" stacks over Σ, i.e., sequences from $\{1, \ldots, k\}^* 0^\omega$, encode finite runs of $\mathcal{A}$. The corresponding trajectory of the realizing PCD system will be "stuck" at the 0 output port of the last state.

Claim 6 (Planar PCD ⇒ Planar Graph) *A non-deterministic automaton can be realized by a 2-dimensional dynamical system only if its transition graph is planar.*

Proof: Consider a deterministic automaton $\mathcal{A} = (Q, \delta)$ (see figure 5-a) realized by a 2-dimensional system $\mathcal{H}$ via a convex state-abstraction function φ. We ignore self-loops in the transition graph of $\mathcal{A}$ since they do not affect planarity. For every $q_i \in Q$ we let $R_i = \varphi^{-1}(q_i)$. For every pair of states q_i, q_j such that $(q_i, q_j) \in \delta$ we pick two points $\mathbf{x}_{ij} \in R_i$ and $\mathbf{y}_{ij} \in R_j$ and a trajectory ξ of $\mathcal{H}$, such that a "segment" of ξ (denoted by ξ_{ij}) joins the points $\mathbf{x}_{ij}$ and $\mathbf{y}_{ij}$ without

[5] A reminiscent of the "prophecy" of [1].

passing through any other R_k (see figure 5-b). Note that because $\mathcal{H}$ is deterministic, if ξ_{ij} and ξ_{kj} have a common point, they coincide after this point (e.g., the point $\mathbf{z}$ in figure 5-b). For every R_i we take a closed bounded convex subset R'_i containing all the abovementioned points $\mathbf{x}_{ij}$ and $\mathbf{y}_{ki}$ (the closed curves in figure 5-b). Then we choose a point $\mathbf{v}_i \in R'_i$ and define a continuous transformation that maps all points in R'_i into $\mathbf{v}_i$, and is one-to-one on all points not belonging to any R'_i (see figure 5-c). We denote the trajectory from $\mathbf{v}_i$ to $\mathbf{v}_j$ by ξ'_{ij}. Finally, for every i, we let $\hat{R}_i$ be the set of all points that belong to two or more trajectories ξ'_{ji} and ξ'_{ki} (for example, $\hat{R}_2$ is the trajectory segment from $\mathbf{z}$ to $\mathbf{v}_2$ in figure 5-c). Once again, by continuous transformation we map $\hat{R}_i$ to $\mathbf{v}_i$ and get an embedding of the transition graph of $\mathcal{A}$ into the plane (figure 5-d)). ∎

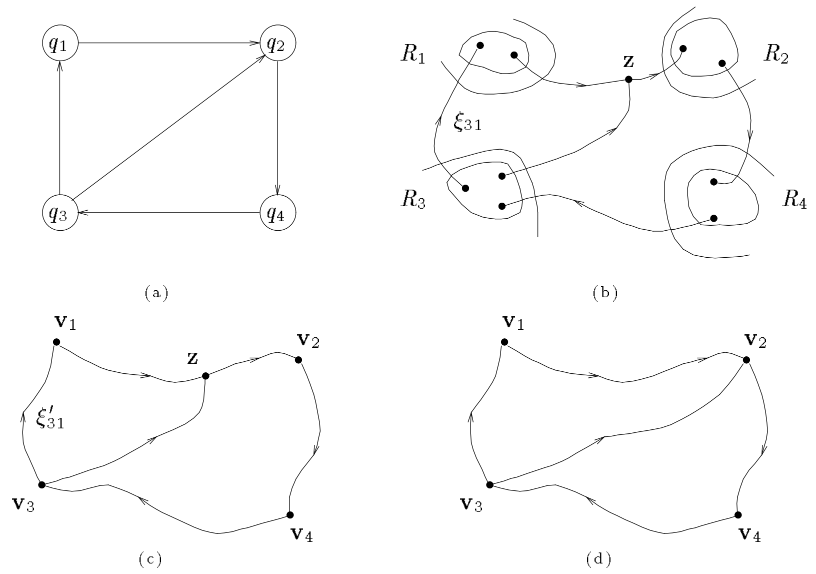

Fig. 5. (a) An automaton. (b) Trajectories in a realizing planar system. (c) Mapping all regions into representative points. (d) An embedding of the transition graph in the plane.

Corollary 7 (Necessity of 3 Dimensions) *Some non-deterministic automata cannot be realized by deterministic dynamical system of dimensionality smaller than* 3.

5 Realization of Two-stack and Turing Machines

The construction of claim 2 generalizes naturally to automata having two stacks (2PDAs). We can define an encoding function $\bar{r} : \Sigma^\omega \times \Sigma^\omega \rightarrow [0,1] \times [0,1]$

by letting $\bar{r}(S_1, S_2) = (r(S_1), r(S_2))$. This way every configuration of the two stacks can be encoded by a point $\mathbf{x} = (x_1, x_2, 0)$ in a two-dimensional input port. The elements that simulate the stack operations (PUSH(v, S_1), PUSH(v, S_2), POP(S_1) and POP(S_2)) operate on the appropriate dimension (according to the stack involved) and leave the other dimension intact. As an example, an element corresponding to PUSH$(0, S_1)$ appears in figure 6. From this we can immediately conclude:

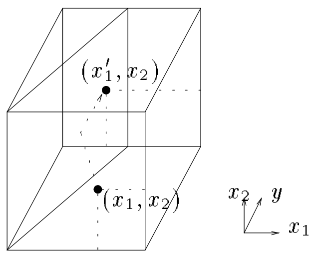

Fig. 6. An element simulating the operation PUSH$(0, S_1)$

Claim 8 *Every 2PDA (and hence any Turing machine) can be realized by a 4-dimensional PCD system.*

Proof: As in claim 2 we pick n elements, arrange them along a line and connect output ports to input ports. The connections should now "carry" two-dimensional information about the configuration, and thus consist of three-dimensional "tubes". The merging of several tubes going to the same state can be done by employing a fourth dimension (no figure), in the same way as two-dimensional "bands" were merged in the case of one-stack PDAs. ∎

But we can do better. First, recall from the standard proof of the equivalence of Turing machines and 2PDAs that some constraints can be imposed on the type of 2PDAs used.

Definition 8 (Normal 2PDA) *Let $\Sigma = \{0, 1, 2\}$. A configuration $(S_1, S_2) \in \Sigma^\omega \times \Sigma^\omega$ is normal if both S_1 and S_2 belong to $\{1, 2\}^* 0^\omega$. A 2PDA is normal if it never pushes 0 to any of the stacks.*

It can be easily verified that normality of the configurations is preserved by normal 2PDAs and that normal 2PDAs can realize Turing machines.

Let $C_0 \subseteq Q \times \Sigma^\omega \times \Sigma^\omega$ be a set of configurations. We denote by $S(C_0, i, j)$ the set $\{(S_1, S_2) \in \Sigma^\omega \times \Sigma^\omega : (\exists \sigma \in L(\mathcal{A}, C_0))(\exists k > 0)(\sigma[k] = (q_i, S_1', S_2') \wedge \sigma[k+1] = (q_j, S_1, S_2)\}$. In other words, $S(C_0, i, j)$ is the set of all 2-stack configurations with which a transition from q_i to q_j can take place in any run starting from C_0. The set of tops of theses stacks, $T(C_0, i, j)$ is defined as $T(C_0, i, j) = \{(v_1, v_2) : \exists (S_1, S_2) \text{ s.t. } (v_1 S_1, v_2 S_2) \in S(C_0, i, j)\}$.

Definition 9 (Separated and Flat States) *Let C_0 be a set of 2PDA configurations. A state q_k of the 2PDA is C_0-separated if for every $q_i \neq q_j \in Q$, $T(C_0, i, k) \cap T(C_0, j, k) = \emptyset$. A state q_k of a 2PDA is C_0-flat if for every j, $S(C_0, j, k) \subseteq \{0^\omega\} \times \Sigma^\omega$. A 2PDA is C_0-regular if each of its states is either C_0-separated or C_0-flat.*

In other words, q_k is separated if, upon entering q_k, the values on the tops of the stacks are sufficient to tell whether we come from q_i or q_j. A state is flat if it is always entered with the first stack empty.

Claim 9 *Any regular 2PDA can be realized*[6] *in 3 dimensions.*

Proof: The only obstacle was the need to merge two or more "tubes" entering the same input port of a state q. If q is separated these tubes do not overlap on the input port and the connections can be made. If q is flat, the relevant information at the input port of q is one-dimensional and all incoming "bands" can be glued together (see figure 7). ∎

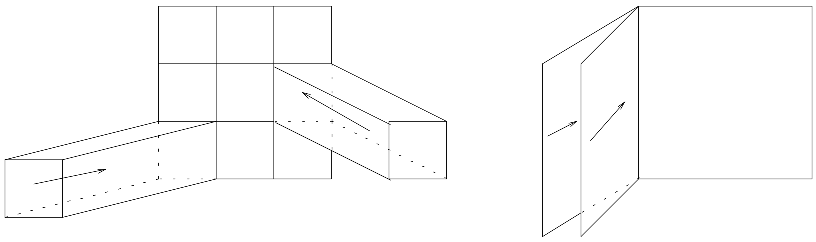

Fig. 7. Realizing entrances to input ports of separated (left) and flat (right) states.

What we are going to show is, informally speaking, that every normal 2PDA can be transformed into a regular 2PDA. The idea is simple: each time after performing a stack operation, we empty one stack while pushing its contents into the other. Then we perform the GOTO's, that is, merge several "bands" that contain one-dimensional information. Before entering the new input port we decode the one-dimensional representation back into two stacks.

We for every $i, j \in \{1, \ldots, n\}$, we define a machine $Encoder_{ij}$ and a machine $Decoder_j$. An encoder takes two normal stack configurations $S_1 = x_1, \ldots x_l 0^\omega$ and $S_2 = y_1, \ldots y_m 0^\omega$ ($x_i, y_i \in \{1, 2\}$) and converts them into $S_1 = 0^\omega$ ("empty" stack) and $S_2 = x_l \ldots x_1 0 y_1 \ldots y_m 0^\omega$. The decoder does the reverse operation. These two machines are described below:

[6] To be more precise, a weaker notion of realization is required which refers only to trajectories starting at C_0.

Encoder$_{ij}$		Decoder$_j$	
E–Entry$_{ij}$:	S_2 :=PUSH$(0, S_2)$	D–Entry$_j$:	S_1 :=PUSH$(0, S_1)$
	GOTO E–Loop$_{ij}$		GOTO D–Loop$_j$
E–Loop$_{ij}$:	(S_1, v) :=POP(S_1)	D–Loop$_j$:	(S_2, v) :=POP(S_2)
	IF $v = 0$ GOTO E–Exit$_{ij}$		IF $v = 0$ GOTO D–Exit$_j$
	IF $v = 1$ GOTO E–Mov1$_{ij}$		IF $v = 1$ GOTO D–Mov1$_j$
	IF $v = 2$ GOTO E–Mov2$_{ij}$		IF $v = 2$ GOTO D–Mov2$_j$
E–Mov1$_{ij}$:	PUSH$(1, S_2)$	D–Mov1$_j$:	PUSH$(1, S_1)$
	GOTO E–Loop$_{ij}$		GOTO D–Loop$_j$
E–Mov2$_{ij}$:	PUSH$(2, S_2)$	D–Mov2$_j$:	PUSH$(2, S_1)$
	GOTO E–Loop$_{ij}$		GOTO D–Loop$_j$
E–Exit$_{ij}$:	GOTO D–Entry$_j$	D–Exit$_j$:	GOTO q_j

Claim 10 *Let $\mathcal{A} = (Q \times \Sigma^\omega \times \Sigma^\omega, \delta)$ be a normal 2PDA and let N be the set of all normal configurations. Then there is an N-regular 2PDA $\mathcal{A}' = (Q' \times \Sigma^\omega \times \Sigma^\omega, \delta')$ that realizes $\mathcal{A}$.*

Proof: We let Q' be the union of Q and the set of states of the corresponding encoders and decoders. The transition function is the union of the transitions of the encoders and decoders with the following variation of δ: every original $\mathcal{A}$-statement of the form q_i ... GOTO q_j is replaced by q_i ... GOTO E–Entry$_{ij}$. It can be easily verified that every q_j is now separated (it is entered only from D–Exit$_j$), that E–Entry$_{ij}$ is separated (it is entered only from q_i) and that D–Entry$_j$ is N-flat (all trajectories starting with a normal configuration will enter the encoder with a normal configuration and will leave the encoder with one stack empty). The other states of the decoders and encoders are obviously separated. By defining $\psi : Q' \times \Sigma^\omega \to Q \times \Sigma^\omega$ as $\psi(q, S) = (q, S)$ iff $q \in Q$ and S is normal, we obtain the desired abstraction. This construction is drawn schematically in figure 8. ∎

Corollary 11 (Main Result) *Any 2PDA can be realized by a 3-dimensional PCD system.*

Corollary 12 (Undecidability) *The reachability problem for 3-dimensional PCD systems is undecidable.*

Proof: Otherwise we could translate every reachability problem of a Turing machine, into a reachability problem between two rational points in a 3-dimensional realizing system, and solve the halting problem. ∎

6 Discussion

We have shown several interesting connections between topological properties of dynamical systems and their computational expressiveness. We have demonstrated a class of simple low-dimensional dynamical systems the trajectories of

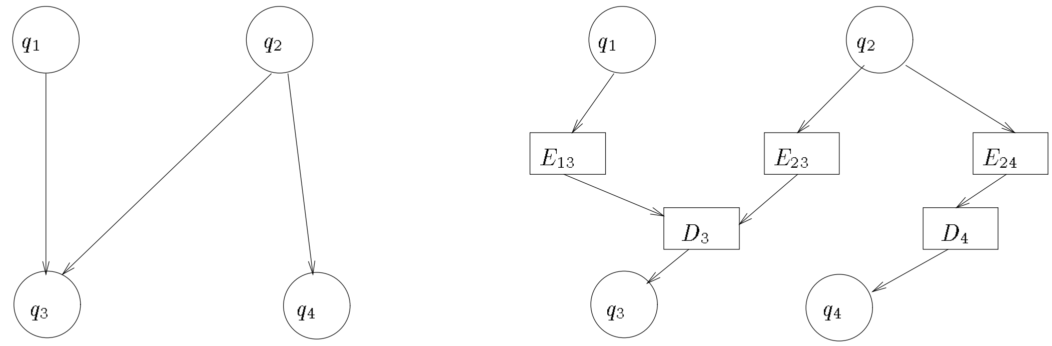

Fig. 8. Augmenting a 2PDA with encoders and decoders.

which can be effectively and precisely computed (given a rational initial condition) and yet their reachability problem is undecidable.

There have been other works on simulation of transition systems by dynamical systems. For example, in [7] the boolean transition function of an automaton, defined over $\{0,1\}^m$, is realized by its continuous extension to $[0,1]^m$ using arithmetical operations. Similarly stacks have been simulated by rational arithmetic in [6]. In these works, however, the simulating system is *already defined over discrete time*, i.e., using iterated maps of the form $x_{n+1} = f(x_n)$. Our construction, on the other hand, uses continuous-time systems.

There have been various undecidability results for other variants of hybrid systems with piecewise-constant derivatives (*timed automata* [2] or *integration graphs* [8]), but those were obtained in a richer model where a transition between regions is accompanied by a discrete change, and the trajectories are discontinuous. In [4] automata were simulated (without a precise formal definition of this term) by smooth dynamical systems defined over a state-space of certain symmetric matrices. Those systems have high dimensionality that grows with the size of the automaton.

The closest work to ours has been reported in [13] where stack machines were constructed from optical elements such as mirrors and lenses. These constructions were used to prove undecidability of the ray tracing problem. It should be noted, however, that optical systems, as well as billiard models, require a richer model, where the phase-space is $2n$-dimensional (the direction in each spatial dimensions is also a state variable) and the trajectories are discontinuous in this phase-space (the direction or the velocity goes through an abrupt change). Hence the equivalence between our PCD results and theirs is an optical illusion.

Finally Putnam [11], while attempting to prove the thesis *every open physical system realizes every automaton*, uses a notion of abstraction we find implausible.

Consider, for example a deterministic automaton without input, generating the sequence $(q_1q_2)^\omega$. Then the system $\frac{dx}{dt}=1$ (or any other system with a non-cyclic behavior) realizes the automaton by letting $\varphi(x)=q_1$ when $2i<x<2i+1$, and $\varphi(x)=q_2$ when $2i+1<x<2i+2$ for any integer $i\geq 0$. This abstraction works only if we consider a fixed initial state (otherwise we need a different abstraction for each state) and, moreover, φ is topologically rather complex: $\varphi^{-1}(q)$ is a union of infinitely many disconnected sets, which contradicts our intuition concerning abstractions.

Acknowledgements

A. Bouajjani contributed to an earlier undecidability result for 6 dimensions that inspired the current construction. The comments of anonymous referees helped in improving the presentation.

References

1. M. Abadi and L. Lamport, On the existence of refinement mappings, in *Proc. of the Third Annual Symposium on Logic in Computer Science*, pages 165–175. IEEE Computer Society Press, 1988.
2. R. Alur and D.L. Dill, Automata for modeling real-time systems, In M.S. Paterson, editor, *Proc. of ICALP'90*, Lect. Notes in Comp. Sci. 443, pages 322–335, Springer-Verlag, 1993.
3. R. Alur, C. Courcoubetis, T. Henzinger, and P. Ho, Hybrid automata: An algorithmic approach to the specification and analysis of hybrid systems. In A. Ravn and H. Rischel, editors, *Workshop on Hybrid Systems*, Lect. Notes in Comp. Sci. Springer-Verlag, 1993.
4. R.W. Brockett, Smooth dynamical systems which realize arithmetical and logical operations, In H. Nijmeijer and J.M. Schumacher, editors, *Three Decades of Mathematical Systems Theory*, pages 19–30, Lect. Notes in Control and Information Sciences, Springer-Verlag, 1989.
5. A. Brondsted, *An Introduction to Convex Polytopes*, Springer-Verlag, New-York, 1983.
6. M. Cosnard, M. Garzon and P. Koiran, Computability properties of low-dimensional dynamical systems, In P. Enjalbert, A. Finkel and K.W. Wagner, editors, *Proc. of the 10th Ann. Symp. on Theoretical Aspects of Computer Science*, pages 365–373, Lect. Notes in Comp. Sci. 665, Springer-Verlag, 1993.
7. R.A. DeMillo and R.J. Lipton, Defining software by continuous smooth functions, *IEEE Trans. on Software Engineering*, Vol. 17, No. 4, 1991.
8. Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine, Integration graphs: A class of decidable hybrid systems, In A. Ravn and H. Rischel, editors, *Workshop on Hybrid Systems*, Lect. Notes in Comp. Sci. Springer-Verlag, 1993.
9. O. Maler, Z. Manna, and A. Pnueli, From timed to hybrid systems, In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, *Proceedings of the REX Workshop "Real-Time: Theory in Practice"*, volume 600 of *Lect. Notes in Comp. Sci.*, pages 447–484. Springer-Verlag, 1992.

10. O. Maler and A. Pnueli, Reachability analysis of planar multi-linear systems, In C. Courcoubetis, editor, *Proc. of the 5th Workshop on Computer-Aided Verification*, Elounda, Greece, volume 697 of *Lect. Notes in Comp. Sci.*, pages 194–209. Springer-Verlag, 1993.
11. H. Putnam. *Representation and Reality*, MIT Press, Cambridge, MA, 1988.
12. X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, An approach to the description and analysis of hybrid systems, In A. Ravn and H. Rischel, editors, *Workshop on Hybrid Systems*, Lect. Notes in Comp. Sci. Springer-Verlag, 1993.
13. J.H. Reif, J.D. Tygar, and A. Yoshida, The computability and complexity of optical beam tracing, in *Proc. 31st Annual Symposium on Foundations of Computer Science*, St. Louis, Missouri, 106–114, IEEE Press 1990.