Cognitive Neurodynamics manuscript No.
(will be inserted by the editor)

Tabor, W. (2009). A dynamical systems perspective on the relationship between symbolic and non-symbolic computation. Cognitive Neurodynamics, 3(4): 415-427.

Whitney Tabor

In press, *Cognitive Neurodynamics*, **2009**

A Dynamical Systems Perspective on the Relationship between Symbolic and Non-symbolic Computation

Received: date / Revised: date

Abstract It has been claimed that connectionist (artificial neural network) models of language processing, which do not appear to employ "rules", are doing something different in kind from classical symbol processing models, which treat "rules" as atoms (e.g., McClelland & Patterson, 2002) . This claim is hard to assess in the absence of careful, formal comparisons between the two approaches. This paper formally investigates the symbol-processing properties of simple dynamical systems called *affine dynamical automata*, which are close relatives of several recurrent connectionist models of language processing (e.g., Elman, 1990). In line with related work (Moore, 1998; Siegelmann, 1999), the analysis shows that affine dynamical automata exhibit a range of symbol processing behaviors, some of which can be mirrored by various Turing machine devices, and others of which cannot be. On the assumption that the Turing machine framework is a good way to formalize the "computation" part of our understanding of classical symbol processing, this finding supports the view that there is a fundamental "incompatibility" between connectionist and classical models (see Fodor & Pylyshyn, 1988; Smolensky, 1988; beim Graben, 2004b). Given the empirical successes of connectionist models, the more general, super-Turing framework is a preferable vantage point from which to consider cognitive phenomena. This vantage

The foundation for this work was accomplished in collaboration with Dalia Terhesiu. Thanks to the attendees at the Workshop on Dynamical Systems in Language at the University of Reading in September, 2008, and to the University of Connecticut Group in Philosophical and Mathematical Logic for helpful comments.

Whitney Tabor
Department of Psychology, U-1020
University of Connecticut
Storrs, CT 06269
USA
Tel.: 860-486-4910
Fax: 869-486-2760
E-mail: whitney.tabor@uconn.edu

may give us insight into ill-formed as well as well-formed language behavior and shed light on important structural properties of learning processes.
Keywords: connectionism, artificial neural networks, Chomsky Hierarchy, Turing Computation, Super-Turing computation, dynamical automata, dynamical recognizers, grammar

1 Introduction

In the 1980s and 1990s, Jerry Fodor and Zenon Pylyshyn, on the one hand, and Paul Smolensky, on the other, had a debate about the relationship between symbolic and connectionist (artificial neural network) approaches to cognition. Part of the discussion centered on a notion of "compatibility". Fodor & Pylyshyn (1988, 1995) argued that the symbolic approach is right about the nature of cognition, and thus that, if connectionism is incompatible with the symbolic approach, it must be rejected. In fact, they argued that there is only one sense in which the connectionist approach might be compatible with the symbolic approach and that is as an "implementation" (see [43]): it might, for example, describe how the primitive symbols of symbol systems are instantiated in physical brains. Crucially, on this implementation view, there is a clean division between the "lower" implementation level and the "higher" symbolic level of description such that all the causal relations at the symbolic level can be described without reference to properties at the implementation level. This claim has important implications for cognitive science: it suggests that cognitive scientists concerned with "high level" phenomena (presumably language, memory, conceptual structure, etc.) need to pay no attention to connectionism or any other implementation mechanism in order to successfully construct a theory of the phenomena. Fodor & Pylyshyn's description of what goes on in symbolic cognition is also in line with the view that the computational processes of high level cognition fall within the domain of "effective computation" as identified by the Church-Turing thesis: high level cognitive computation can be fully formulated within the framework of so-called "Turing Computation".

Smolensky (1988, 1995a, 1995b) took a contrary position, arguing for "incompatibility" between connectionism and the symbolic approach. He distinguished between "implementation" and "refinement", arguing that there is a way of doing connectionist modeling of cognition in which the models are not implementations but refinements of the symbolic models. He argued that the sense of implementation that Fodor & Pylyshyn must mean (in order to push the point about the irrelevance of connectionism to high level cognition) is the sense used in computer science, where a low-level description (e.g., an assembly language description like MIPS or SPARC) is an implementation of a high-level description (e.g., BASIC or JAVA). In this case, the high-level description and the low-level description contain the same algorithmic information: a programmer employing the high level language will perceive no difference in functionality whether the high level language is implemented, for example, in MIPS or SPARC. A low-level refinement, by contrast, contains additional algorithmic information that is lacking in a high-level description. He says, "Far from the conclusion that '*nothing* can be gained by going to

the lower-level account', there is *plenty* to be gained: completeness, precision, and algorithmic accounts of processing, none of which is generally available at the high level." (Smolensky, 1995, p. 168)

This claim of novelty and relevance for connectionist models would be vindicated if it could be shown that connectionist models do something systematically different from symbolic models and that empirical evidence favors the connectionist approach. In fact, connectionist networks have had unusual empirical success in several domains where classical approaches have not made much headway. Among these are the learning of syntactic structure from word sequence data [5,6,34,33], the modeling of quasi-regular behavior in phonology and morphology [17,26,39,38], and the derivation of overregularization as a consequence of the learning process in language acquisition [7,27,36]. In each of these cases, the models exhibit surprising, empirically justified behaviors and this makes them interesting to the field of cognition. However, without formal insight into the relation of the connectionist models to symbolic models, it is not clear what fundamental conclusions these results imply for the field. It might be that the connectionist models are simply an alternative form of symbolic model, perhaps one in which the symbols refer to more fine-grained features of mental computation than those in classical cognitivist theories. It is noteworthy that some researchers participating in the debate put a large amount of weight on subtle verbal contrasts which are not obviously clarifying: e.g. the contrast between whether the language system "uses mechanisms that are combinatorial and sensitive to grammatical structure and categories" [25] or whether rules are "approximate descriptions of patterns of language use; no actual rules operate in the processing of language" [22]. In this paper, in order to make headway on the issue, I adopt a particular, formal approach, examining a simple type of dynamical system that is closely related to the Elman network [5]. I make the assumption that classical symbolic computation can be appropriately understood as computation by some type of Turing device (a review of Turing devices is provided below). I then argue, following [24] and [40], that the dynamical models include this kind of computation as one possibility but also include additional, super-Turing behaviors. I further argue that understanding these additional behaviors and their relationship to classical behaviors may be helpful for making new headway in the cognitive sciences.

In a recent development of the connectionist/symbolist debate, beim Graben [2] suggests that we take advantage of insights from dynamical systems theory, especially the method of *symbolic dynamics* [4], to clarify the discussion. In particular, it is helpful to note that the state space of a connectionist model is generally a real vector space so there is a continuum of possible states. Focusing on discrete update (iterated map) dynamics, the method of symbolic dynamics adopts a finite partition of such a state space and treats the partition indices as symbols. Thus, the iterating dynamical system on the vector space is associated with an iterating dynamical system on the symbol space. This correspondence gives rise to two ways of describing the system, which beim Graben & Atmanspracher (2006) [13] refer to as the "ontic" (vector space) and "epistemic" (symbolic alphabet) levels. Beim Graben (2004b) suggests that the ontic/epistemic distinction provides a good model for the

subsymbolic/symbolic (low/high) distinction discussed in cognitive science. He goes on to suggest that the dynamical notion of *topological equivalence* (or *topological conjugacy*) can help formalize the concept of compatibility between descriptions. Two dynamical systems, $f : X \to X$ and $g : Y \to Y$ are *topologically conjugate* if there is a homeomorphism $h : X \to Y$ satisfying $g \circ h = h \circ f$. A *homeomorphism* is a continuous, one-to-one function. Topological conjugacy is a kind of structural correspondence: the states of two conjugate systems are in complete, point-to-point correspondence, and the patterns of transitions between states also correspond perfectly across the systems. A particularly strong kind of topological conjugacy occurs when the partition is a *generating partition*. A partition is generating if the future of the boundaries of the partition subdivides the space arbitrarily finely. In this case, almost all the information about the details of the continuous dynamics can be reconstructed from the information about how the symbols are sequenced. For many dynamical systems, however, there is no generating partition and it is not possible to choose a single partition for which the symbolic dynamics reveals (almost) all the detail about the subsymbolic (continuous) dynamics. Such cases, beim Graben (2004b) maintains [2], should count as cases of incompatibility between symbolic and subsymbolic dynamics. The fact that they exist among connectionist networks reveals, against Fodor & Pylyshyn's position, a fundamental incompatibility between the symbolic and connectionist approaches.

In this paper, I side with Smolensky and beim Graben in arguing for incompatibility between connectionist and symbolic systems. I also endorse beim Graben's emphasis on the value of a dynamical systems perspective. However, I'll suggest that beim Graben's formalization of the notion of incompatibility is not the most useful one, and if we adopt a formulation more suitable to the central questions facing cognitive science, then the incompatibility is deeper than has been demonstrated in the aforementioned papers. Likewise, the lack of completeness and precision that Smolensky mentions is certainly valid, but it does not seem to argue for a strong change in our approach to cognitive science. After all, the high-level algorithms that run on digital computers do not completely and precisely describe the physical actions of the computer chips; yet this lack does not, in any significant sense except for the remote possibility of hardware errors, require that we pay attention to chip physics when describing the information processing behavior of digital computers. Beim Graben (2004b)'s examples of dynamical systems with incompatible epistemic and ontic dynamics are generally cases in which one partition induces symbolic dynamics corresponding to some familiar or simple algorithm (e.g., parsing a sentence, categorizing objects based on features, or a simple finite-state process). None of the examples have generating partitions (at least under the parameterizations considered), so none exhibit strong informational equivalence across the levels. I will argue, however, that such cases should not interest us very much. They are all situations in which a vector space dynamical system can be used to generate a familiar algorithmic process via symbolic dynamics and there is nothing unexpected in the behavior at the symbolic level. In particular, Fodor & Pylyshyn's claim of separability seems to hold: if our interest is in the higher level description

(i.e. the symbolic dynamics), then we don't need the vector space description to characterize these dynamics.

Smolensky (1995a) [43] also states that "algorithmic accounts of processing" are not available at the higher level for the connectionist systems he has in mind (p. 168). This *is* news if it means that no algorithm at all will suffice to describe the high-level behaviors of some of the systems. However, Smolensky does not provide evidence that such cases exist. A main purpose of this paper is to show, in keeping with [40], that such cases do exist, and to suggest that they have implications for the type of phenomena we expect to observe in high level cognition. In making this point, I'll note that beim Graben's focus on the difference between systems with generating partitions and those which lack them is very helpful. However, I'll argue that it is actually the cases *with* generating partitions that exhibit the kind of incompatibility we should be interested in. I reach this conclusion by a simple argument: the case of greatest interest is the case in which a connectionist model does something that a classical symbol system cannot do; under certain conditions, connectionist models with real-valued weights compute non-Turing computable functions [40,52]. In fact, they can only do this when there *is* a generating partition. Assuming that Turing computation is a good formalization of classical symbolic computation, I argue that "incompatibility" of classical and symbolic computation should be associated with the availability, not the lack, of a generating partition.

1.1 Overview

The essence of the present work is an analysis of the computational properties of *affine dynamical automata*. Section 2 reviews basic distinctions between types of formal languages. Section 3 defines affine dynamical automata and their associated formal languages. Section 4 presents several theorems which support a structural classification of all parameter settings of an interesting subset of affine dynamical automata. Section 5 presents a parameter space map based on this classification. I suggest that such maps offer a useful new perspective on cognitive processes like development and learning. Section 6, the Conclusion, considers the implications of these findings for cognitive science, returning to the question about the compatibility or incompatibility of the connectionist and symbolic views.

Affine dynamical automata are a type of dynamical system with feedback. The present work is thus closely related to other work that asks how complex computation can be accomplished by feedback dynamical systems. Many efforts in this regard have been directed at defending the claim that recurrent connectionist models, a type of feedback dynamical system, can handle the recursive computations that appear to underlie natural language syntax [6,28,42,48]. Recently several projects have adopted some of these dynamical recursion mechanisms to model neural data on real time language processing [12,14–16] (Gerth & beim Graben, this issue [11]). In all of these projects, the focus is on getting the dynamical system to exhibit a complex behavior that the classical paradigm already handles. The current work takes a more general perspective, noting that that feedback dynamical systems can

handle complex recursive computations and also handle processes that are not computable by Turing machines. The focus is on clarifying the relationships between these different behaviors within the general framework of super Turing computation and on highlighting the possible usefulness of the more general perspective to the field of cognition.

2 Relevant formal language classes

This section describes the classes of formal languages that are important in the discussion below.

A formal language is standardly defined as a set of finite-length strings drawn from a finite alphabet [18]. Here, I extend the definition to include one-sided infinite length strings over a finite alphabet. Many of the same classificational principles apply to this more general case. The Chomsky hierarchy [3] classifies finite-sentence formal languages on the basis of the type of computing mechanism required to generate (or recognize) all and only the strings of the language. For example, the language L_1, consisting of the strings $\{ab, abab, ababab, \ldots\}$, can be generated/recognized by a computing device with a finite number of distinct states. Such a device is called a "Finite State Automaton" (FSA) and its language is called a "Finite State Language" . A more powerful type of device, the "Pushdown Automaton" (PDA) consists of a finite state automaton combined with an unbounded stack (first-in, last-out) memory. The top (last-in) symbol of the stack and the current state of the FSA jointly generate/predict the next symbol at each point in time. Each PDA language can be generated by a "Context Free Grammar" (CFG) and vice versa, so PDAs and Context Free Grammars are equivalent formalisms. A CFG is a finite list of constituency rules of the form, A $\rightarrow A_1 A_2 \ldots A_n$ where n is a finite positive integer (possibly different for each rule in the list). The language $L_2 = \{ab, aabb, aaabbb, \ldots\}$ (also called "$a^n b^n$"), consisting of the strings with a positive whole number of "a"'s followed by the same number of "b"s, can be processed by a PDA (or CFG), but not by a FSA. Arguments have several times been advanced on linguistic grounds that human language processing employs something similar in computational capability to a Context Free Grammar (or PDA) [10], though the current consensus is that a slightly more powerful device called a Tree Adjoining Grammar [21] provides the best characterization of the formal patterning of the syntax of natural languages [37]. An even more powerful device than those so far mentioned is the "Turing Machine" (TM). A TM consists of a FSA that controls an infinite tape memory (unbounded number of slots, the FSA controller can move step-by-step along the tape, either backwards or forwards, reading symbols and then either leaving them untouched or overwriting them as it goes). Chomsky (1956) [3] noted that the devices on this hierarchy define successively more inclusive sets of formal languages: every FSA language can be generated by a PDA; every PDA

language can be generated by a TM; but the reverse implications do not hold.[1]

Consider the set of all one-sided infinite strings formed by right-concatenation of strings drawn (with replacement) from a countable source set. If the source set is the language of a Chomsky Hierarchy device, we say that the device generates/recognizes the set of one-sided infinite strings. If, on the other hand, a set of one-sided infinite strings has no such source set, then we say that that set of one-sided infinite strings is not Turing machine computable.

Although they are not conventionally treated as a level on the Chomsky Hierarchy, there is a proper subset of the set of FSA languages that will be of relevance later in the present discussion: the Finite Languages. These can be specified with a finite list of finite-length strings. There are also sets of strings that are not generated by any Turing Machine, the most powerful device on the Chomsky Hierarchy. These string-sets are sometimes called "super-Turing" languages [40]. These will also be relevant in the discussion below.

3 Formal Paradigm: Affine Dynamical Automata

Pollack (1991) defined *dynamical recognizers*: suppose $f_i : X \rightarrow X$ for $i = 1, \ldots, K$ are functions on the continuous space, X. Given a string of integers, σ, drawn from the set $\{1, \ldots, K\}$, we start the recognizer in a specified initial state in X and apply the functions corresponding to the integers of σ in order. If, at the end of applying the functions, the system is in a specified subset of the space (sometimes called the "accepting region"), then the string σ is said to be accepted by the dynamical recognizer. The set of strings accepted by the recognizer is a formal language. Moore (1998) [24] explores conditions under which dynamical recognizers produce languages in various computational classes related to the Chomsky Hierarchy. One notable result is that, by choosing functions and regions judiciously, one can make dynamical recognizers for all Chomsky Hierarchy classes as well as for all super-Turing languages [24].

Here, I focus on a class of devices called "affine dynamical automata". These are a subclass of the "dynamical automata" defined in [48] which have similar computational properties to dynamical recognizers. Unlike dynamical recognizers, which can execute any function at any time, the functions of dynamical automata have restricted domains, so they are useful for modeling organisms (like language producers) whose history restricts the set of possible (or probable) behaviors at each point in time. A few preliminary definitions support the main definitions.

(1) **Def.** An *affine function* $f : d \subset \mathbf{R} \rightarrow \mathbf{R}$ is a function of the form $f(h) = ah + b$, where a and b are real numbers. If $b = 0$, then f is *linear*.

[1] Chomsky (1956) [3] also identified the set of Linear Bounded Automata which define a class of languages ("Context Sensitive Languages") that have the PDA languages as a proper subset, and are themselves a proper subset of the TM languages. Context Sensitive Languages are not of central concern in the present study.

If $b \neq 0$, then f is *strictly affine*.

(2) **Def.** The *length* of a string σ is denoted $|\sigma|$ and is equal to the number of characters in σ if σ has a finite number of characters and is called "infinite" if σ does not have a finite number of characters.

(3) **Def.** If an infinite string has an initial character, it is called a *right sided infinite string*. If it has a final character, it is called a *left-sided infinite string*.

We will not be concerned here with the difference between left-sided and right-sided infinite strings, so, for convenience, we assume that all infinite strings are right-sided infinite (they have an initial character but no final character).

(4) **Def.** The *length n initial substring* of a string σ of length $\geq n$ is denoted $\sigma[n]$ and consists of the first n characters of σ in order.

(5) **Def.** A finite string σ is a *proper initial substring* of a string σ' either finite or one-sided infinite, if $\sigma'[|\sigma|] = \sigma$ and $|\sigma'| > |\sigma|$.

(6) **Def.** An *affine dynamical automaton* is a device

$$M_{\mathbf{h}_0} = (H, F, \mathbf{h}_0) \tag{1}$$

where $H = \{\mathbf{h} \in \mathbf{R}^{\mathbf{N}} : 0 \leq h_j \leq 1, j \in \{1, \ldots, N\}\}$, $F = \{f_1, f_2, \ldots, f_K\}$ is a set of affine functions that map H into itself, and $\mathbf{h}_0 \in H$ is the initial state. The domain, d_i, of each function, f_i, is $\{h \in [0,1] : \text{f\_i(h)} \in [0,1]$ where f_i has the same functional form as f_i but unrestricted domain$\}$. For a string, σ, of integers drawn from $\Sigma = \{1, \ldots, K\}$, the system starts at $\mathbf{h}_0$ and, if possible, invokes the functions corresponding to the integers of σ in order. A string is said to be *traceable* under $M_{\mathbf{h}_0}$ if every function corresponding to its integers can be invoked, in order, from left to right, starting from $\mathbf{h}_0$. A string is *maximal* under $M_{\mathbf{h}_0}$ if it is traceable and it is not an initial substring of any longer, traceable string. The set of maximal strings under $M_{\mathbf{h}_0}$ is the *language of* $M_{\mathbf{h}_0}$ and is denoted $L(M_{\mathbf{h}_0})$. $M_{\mathbf{h}_0}$ is said to *generate* and *recognize* its language and each string of its language.

For M_{h_0} an affine dynamical automaton, I will use the notation M to refer to the set of affine dynamical automata M_{h_0} with $h_0 \in [0,1]$.

Affine dynamical automata are closely related to connectionist networks with sigmoidal activation functions. For example, Simple Recurrent Networks [5,6] can approximately process infinite-state formal languages by using the contraction and expansion of the so-called "linear" (or central) section of the sigmoid to traverse fractal sets [30,32,31,54]. Closely related devices called

"Fractal Grammars" employ affine dynamical automata as their core computational mechanism and exactly process infinite state languages in a manner similar to that of Simple Recurrent Networks [48,49]. In particular, the affine maps of affine dynamical automata play an analogous role to the linear sections of the sigmoids. Thus, the study of affine dynamical automata may be informative about how connectionist networks can process complex languages.

Affine dynamical automata can generate both finite and infinite length strings:

(7) **Def.** Let $M = (H, F, \mathbf{h}_0)$ be an affine dynamical automaton. Let σ be a maximal string of M_{h_0}. If the length of σ is finite, then σ is a *finite sentence* of M_{h_0}. If the length of σ is not finite, then σ is an *infinite sentence* of M_{h_0}.

(8) **Def.** An affine dynamical automaton is *proper* if $H = \bigcup_{i=1}^{K} d_i$, i.e., if at least one of the automaton's functions can be applied at every point in H.

In a proper affine dynamical automaton all the sentences have infinite length. Although standard formal language theory focuses on finite sentence languages, the theory extends naturally to infinite sentence languages. I focus on proper affine dynamical automata here.

Even two function affine dynamical automata generate a rich variety of formal languages. The next section supports this claim by providing examples of affine dynamical automata that generate finite languages, finite state (non finite) languages, context free (non finite state) languages, and super-Turing languages. In fact each of these cases exists among both linear and and strictly affine dynamical automata.

4 Range of phenomena within the class

4.1 Finite Languages

A cobweb diagram illustrating one trajectory of the affine system

$$\begin{aligned} f_1(h) &= h + \tfrac{1}{8} \\ f_2(h) &= h + \tfrac{3}{8} \\ h_0 &= 1/6 \end{aligned} \tag{2}$$

is shown in Figure 1. This trajectory corresponds to the sentence, "1 2 1 1". Only sentences satisfying

$$\frac{1}{6} + \frac{3n}{8} + \frac{m}{8} < \frac{7}{8} \tag{3}$$

where $m \geq 0$ is the number of 1's and $n \geq 0$ is the number of 2s can be generated under this system. Since there are only a finite number of such cases, this automaton generates a finite language.

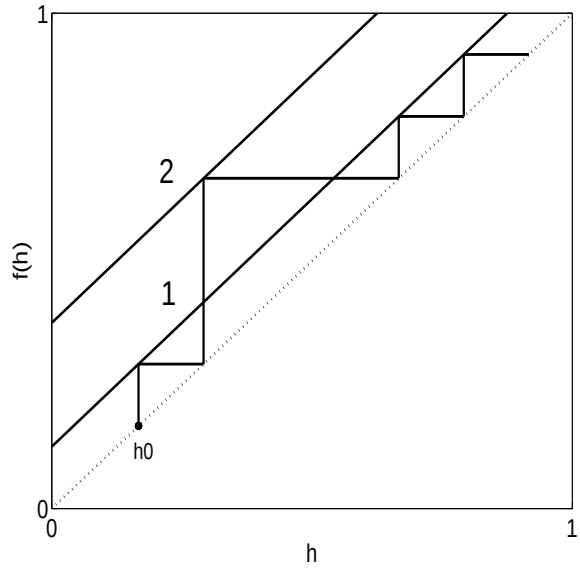


Fig. 1 A sample trajectory of $f_1(h) = h + 1/8, f_2(h) = h + 3/8, h_0 = 1/6$. The state space is the interval [0, 1]. The line segment labeled "i", for i $\in \{1, 2\}$, is a plot of $f_i(h)$. The zig-zag line indicates the sample trajectory.

The linear system

$$\begin{aligned} f_1(h) &= \tfrac{4h}{3} \\ f_2(h) &= 2h \\ h_0 &= \tfrac{1}{4} \end{aligned} \tag{4}$$

generates the finite language {"111", "12", "211", "22"}.

Every linear system with $a_1 > 1$ and $a_2 > 1$ and $h_0 > 0$ generates a finite language.

4.2 Finite State Languages

If the initial state in the previous example is replaced with $h_0 = 0$, then the system generates a finite-state language that is not a finite language. Figure 2 depicts a FSA that generates/recognizes this language. I call this FSA BA for "Bernoulli Automaton" because one can think of it as a nonprobabilistic version of the Bernoulli Process. The circle corresponds to the single state of the machine. The system always starts in the state labelled "S". It moves between states by following arcs. The label on each arc indicates the symbol that is generated when the system traverses the arc. The Bernoulli Automaton generates all infinite sentences on the two-symbol alphabet $\Sigma = \{1, 2\}$. In fact, every linear system with $h_0 = 0$ generates $L(BA)$ and every linear system with $a_1 \leq 1$ and $a_2 \leq 1$ generates $L(BA)$ for all initial states.

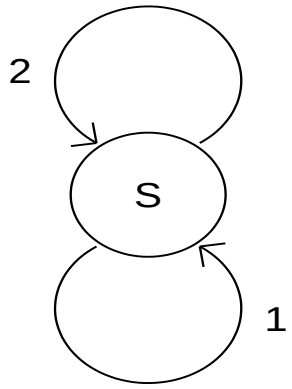


Fig. 2 The Bernoulli Automaton.

Figure 3a shows trajectories of the affine system

$$\begin{aligned} f_1(h) &= 2h \\ f_2(h) &= -\tfrac{1}{2}h + \tfrac{7}{6} \\ h_0 &= \tfrac{1}{5} \end{aligned} \tag{5}$$

This system is generated by the non-deterministic finite-state machine shown in Figure 3b.

4.3 Context Free Languages

Context free languages (i.e., languages generated/recognized by PDAs or CFGs) arise when contraction and expansion are precisely matched [48]. In particular, the following theorem states conditions under which linear dynamical automata generate context free languages:

Thm. 1 Let $DA = ([0,1], \{f_1(h) = a_1 h, f_2(h) = a_2 h\}, h_0 > 0)$ be a linear dynamical automaton. If

$$a_1 = a_2^{-\frac{\alpha}{\beta}} \tag{6}$$

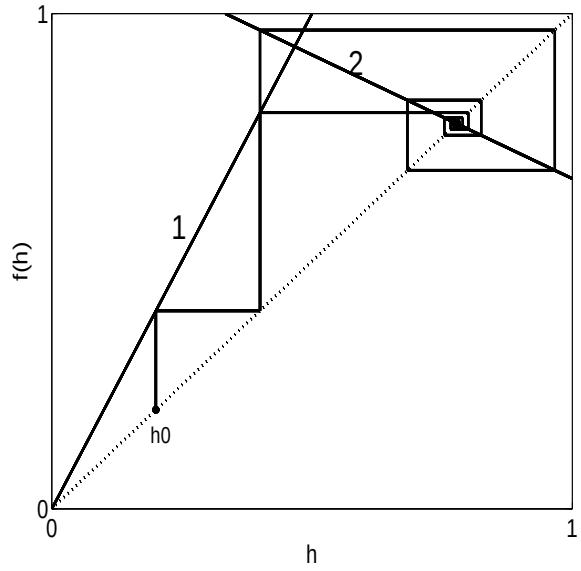
where α and β are positive integers and a_1 and a_2 are not both 1, then $L(DA)$ is a context free language that is not a finite state language.

Appendix A1 provides a proof.

Figure 4 illustrates a trajectory and provides a context free grammar for a linear dynamical automaton that generates context free languages. A corresponding affine case is:

$$\begin{aligned} f_1(h) &= 2h - \tfrac{1}{3} \\ f_2(h) &= \tfrac{1}{2}h + \tfrac{1}{6} \\ h_0 &= 1 \end{aligned} \tag{7}$$

a. Trajectories.



b. Finite state diagram.

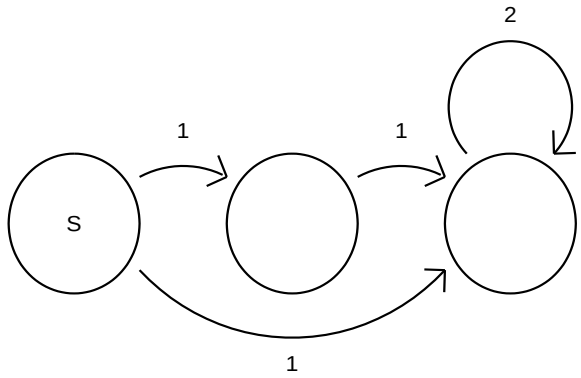


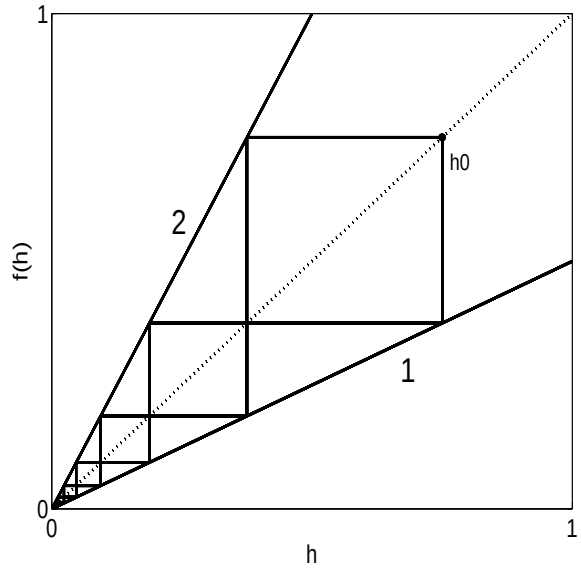
Fig. 3 Trajectories (a) and finite state diagram (b) for the system, $f_1(h) = 2h, f_2(h) = -1/2h + 7/6, h_0 = 1/5$.

4.4 Super-Turing Processes

Super Turing behavior occurs in the linear regime when the system has both expansion and contraction, but the two are not related by a rational power. To show this, some background is needed.

(9) **Def.** Let $M_{h_0} = (H, F, \mathbf{h}_0)$ be a dynamical automaton with $f_i(h) = a_i h + b_i$. If $a_i \neq 0$ for $i \in \{1, \ldots, |F|\}$, then let $F^{-1} = \{f_i^{-1}(h) = \frac{h-b_i}{a_i}$ for

a. Trajectories.



b. Context Free Grammar.

S → 1 S 2 S
S → ϵ

Fig. 4 Trajectories (a) and context free grammar (b) for the system, $f_1(h) = 2h$, $f_2(h) = \frac{1}{2}h$, $h_0 = \frac{3}{4}$.

$i \in \{1, \ldots, |F|\}\}$. Then M_{h_0} is said to be *invertible* with inverse $M_{h_0}^{-1} = (H, F^{-1}, \mathbf{h}_0)$.

Consider the set of linear dynamical automata, $DA = (H = [0,1], \{f_1(h) = a_1 h, f_2(h) = a_2 h\}, h_0 \in H)$ with $a_1 > 1$ and $0 < a_2 < 1$. Note that d_1, the domain of f_1 is $[0, 1/a_1]$. Thus, $b = 1/a_1$ is the boundary between the subset of H on which f_1 can apply and the subset of H on which it cannot. b is called an *internal partition boundary* of the state space. Consider DA^{-1}, the inverse of DA:

$$DA^{-1} = ([0,1], \{f_1 = \frac{h}{a_1}, f_2 = \frac{h}{a_2}\}, \Sigma = [1,2], h_0 \in [0,1]) \tag{8}$$

DA^{-1} computes the inverse of the history of DA for each h.

(10) **Def.** The *future* of a state, $h \in H$, is the set $\{x \in H : x = \Phi_\sigma(h)$ for some $\sigma \in \Sigma^*\}$.

That is, the future of a state h is the set of all states that the system can visit when started at h, including h itself.

Lemma Let DA be a set of invertible affine dynamical automata with inverses DA^{-1}. Consider an internal partition boundary, b of DA. If some subset of DA_b^{-1} is dense in an uncountable set in H, then for uncountably many initial states h_0, DA_{h_0} generates a super Turing language.

A proof of the Lemma is provided in Appendix A2. The Lemma provides a way of showing that some linear dynamical automata exhibit super Turing behavior.

Thm. 2 Let $DA = (H = [0,1], \{f_1(h) = a_1 h, f_2(h) = a_2 h\})$ be a linear dynamical automaton. If

$$a_1 = a_2^{-\gamma} \tag{9}$$

where γ is a positive irrational number, then there are states $h \in H$ for which $L(DA_h)$ is not generable/recognizable by any Turing device.

Theorem 2 is proved in Appendix A2.

Figure 5 shows a linear dynamical automaton that generates super-Turing languages. The illustration shows a single trajectory starting from the initial state, $h_0 = \frac{3}{4}$. I do not know if this particular initial state generates a super-Turing language. But, by Thm. 2, there is bound to be a state very close to this initial state that generates a super-Turing language. Thus, the figure is illustrating an approximation of a super-Turing case. Even this approximation appears to be less regular than the trajectories in the finite, finite-state, and context-free examples discussed above.

It is a little easier to show the existence of super Turing behaviors in the affine case. The chaotic Baker Map [4] is equivalent to the affine dynamical automaton,

$$\begin{aligned} f_1(h) &= 2h \\ f_2(h) &= 2h - 1 \end{aligned} \tag{10}$$

The inverse of the Baker Map (BM^{-1}) is

$$\begin{aligned} f_1(h) &= \tfrac{1}{2}h \\ f_2(h) &= \tfrac{1}{2}h + \tfrac{1}{2} \end{aligned} \tag{11}$$

The Baker Map has one internal partition boundary at h = 1/2. The union of the zero'th through the n'th iterates of 1/2 under BM^{-1} consists of the points $\{k/2^n\}$ for $k = 1 \ldots 2^{n-1}$. Therefore any point in H can be arbitrarily well approximated by a finite iterate of BM^{-1} and the future of 1/2 is dense in H. Thus, by the Lemma, there are uncountably many initial states h, for which $L(BM_h)$ is not generable by a Turing Machine. A similar argument applies to the Tent Map [4].

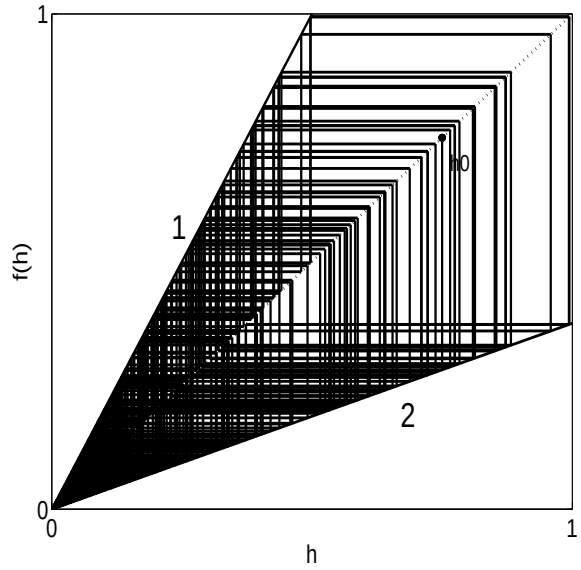


Fig. 5 A single trajectory of the system $f_1(h) = 2h$, $f_2(h) = \frac{1}{2\sqrt{2}}h$, $h_0 = \frac{3}{4}$.

5 Parameter Space Map

The results of the preceding section justify the construction of the parameter space map shown in Figure 6. The slope of the first function is encoded along the horizontal dimension and the slope of the second is encoded along the vertical.[2] The map indicates the type of language associated with most initial states under the parameter setting. The blocks with shaded bands contain settings that generate context-free languages and settings that generate super-Turing languages. Within the blocks, when the exponent, γ, is rational, then all initial states except $h_0 = 0$ are associated with context free languages that are not finite state languages. The greyscale indicates, for rational values of γ, the number of symbols required to write a grammar of the language, with lighter shading corresponding to languages requiring fewer symbols. The super-Turing languages occur when γ is irrational, but only for a proper subset (uncountably infinite) of the initial states.

6 Conclusion

The paper has provided evidence that a rich variety of computational structures occur in even very simple dynamical computing devices. All two-function linear dynamical automata were considered, supporting the presentation of a parameter space map for this subclass.

[2] Maps with one or both slopes negative generate only finite or finite state languages and are not shown.

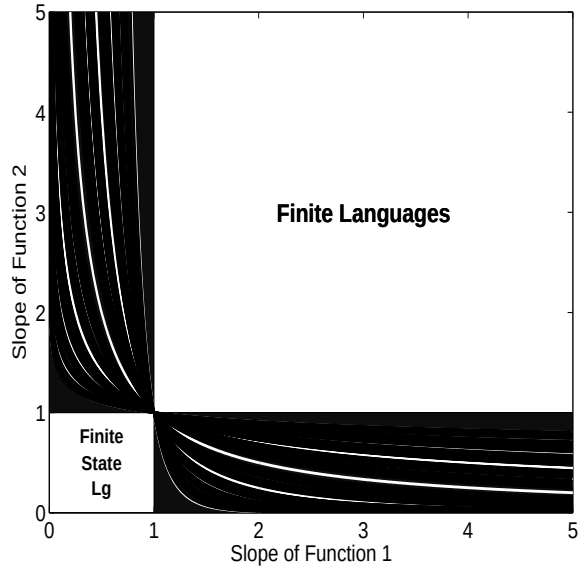


Fig. 6 Deployment of language types in the parameter space of two-function linear dynamical automata.

I return now to the issues raised in the introductory argument about the compatibility of connectionist and classical symbolic computation, arguing that the present, super-Turing perspective offers some valuable new tools for approaching challenging problems in the study of cognition.

6.1 Relation between Affine Dynamical Automata and Connectionist Models

In the Introduction, I suggested that the rich repertoire of these affine devices extends to other connectionist devices, including those that have been shown to exhibit appealing empirical properties, like induction of syntactic structure, sensitivity to quasi-regularity in morphology, and transient overgeneralization. The plausibility of this claim is suggested by related work which shows that a gradient-based learning model with affine recurrent processors at its core exhibits learning behaviors similar to that of other connectionist models [49]. Nevertheless, there is a need for further formal development to clarify this relationship. One important goal is to extend the approach to higher-dimensional networks. Another is to consider nonlinear/nonaffine functions. I leave these as goals for future work.

6.2 Turing machine computation as a formalization of "computation" in the classical sense

The argument of the Introduction was also based on the assumption that the Turing machine framework is a good formalization of the "computation part" of the symbolic paradigm of cognitive science. This assumption is worth reviewing here in order to lay some groundwork for assessing the merits of the super-Turing framework. Fodor & Pylyshyn (1988) argue that the core computational feature of symbolic theories is "combinatorial symbol combination". This property is plausibly a foundational element in four empirically defensible properties that Fodor and Pylyshyn cite as evidence for the symbolic approach: productivity (unbounded combination ability), systematicity (understanding "John loves Mary" implies understanding "Mary loves John"), compositionality (the parts–e.g. words—make essentially the same contribution to the meaning of the whole in all contexts), and inferential coherence (there is an isomorphism between human logic and actual logic).

Something like a context free grammar seems to have the right properties to support a model of at least most of these phenomena. If a context free grammar has recursive rules, it exhibits unbounded combination ability (productivity); if it is combined with a Montagovian semantics [23], in which syntactic combination rules are parallel to semantic combination rules, and the categorical equivalence of "John" and "Mary" as syntactic elements can be justified, then the context freeness implies the interchangeability of their roles (systematicity). Likewise, the context freeness implies a context independent interpretation of all constituents (compositionality). Although the grammatical nature of actual logic is an open question, the evidence that Fodor & Pylyshyn bring to bear in favor of inferential coherence is based on the compositional structure of certain particular logical relationships: it would be bizarre, they say, to have a logic in which $P \wedge Q \wedge R \rightarrow P$ but $P \wedge Q \not\rightarrow P$. Again, a context free grammar is suitable for generating the well-formed formulas of the non-bizarre logical subsystem alluded to here. All of this suggests that some kind of Turing mechanism, perhaps a context free grammar or something akin to it, is a good mechanism for specifying the computations of thought, as conceived under the symbolic paradigm.

6.3 Benefits of the Super-Turing Perspective

What benefit, then, does the super-Turing framework offer cognitive science? I suggested at the beginning, that the empirical credentials of connectionist networks are strong enough that pursuit of a formal understanding of their computational properties is warranted. Now that the formal analysis is in view, a few other observations can be made in support of the usefulness of the super-Turing perspective.

(i) Gradual metamorphosis is a hallmark property of grammar change [19] and grammar development in children [20,45]. I have argued elsewhere that, under the classical paradigm, there is no satisfactory way to model the phenomenon of gradual grammar change [46,47]. The essential problem

is that grammatical structures seem to come and go from languages via gradual accretion/loss of similar structures. But the classical paradigm offers no systematic way of measuring similarity relationships among grammars. The super-Turing framework offers a natural insight into this problem by revealing real-valued metric relations among grammars in neurally motivated computing mechanisms.

(ii) One may note that to distinguish between Turing and super-Turing mechanisms among the linear dynamical automata, one must refer to infinite-precision encodings of the parameter values. Since the real world is noisy, infinite precision is not realistic, so perhaps there is no need to consider the super-Turing devices (considering only Turing devices will get us "close enough" to everything we need to know). This argument is reminiscent of the argument that there is no need to consider infinite state computation in the study of language because one can only ever observe finite-length sentences. This argument can be countered by noting that consideration of infinite state computing devices allows us to discover principles of linguistic and logical organization like compositionality which would be out of reach if only finite state computation or finite computation were employed. This provides at least a cautionary note: we should not reject formal frameworks out of hand just because they refer to ideal systems.

However, there is also a reason that we may, from a practical standpoint, want to include super-Turing computation in the purview of cognitive science. There appear to be regions of the affine parameter space where the systems exhibit a behavior akin to robust chaos. In particular, when both a_1 and a_2 are greater than 1, the system vigorously explores the structure in the initial state, producing exponentially diverging trajectories for an uncountable number of initial states. This results in traditional chaos (e.g., Devaney, 1989) [4] in the deterministic Baker Map and Tent Map mentioned above. A natural extension of the traditional chaos to nondeterministic affine dynamical automata [50] suggests that the nondeterministic cases may have similar properties. There appear to be regions of positive measure in the parameter space of two function affine dynamical automata where this generalized chaos is pervasive; in this sense the chaos is robust. We know from studies in other sciences that chaos is a practically relevant phenomenon that can be detected empirically [1]. Therefore, these chaotic regions of the parameter space may be relevant in a measurable way to the science of mind—perhaps, for example, they will offer a way of modeling ill-formed behavior, an area about which classical cognitive theory, with its emphasis on well-formedness, has relatively little to say [51]. Now, the question is: What is the relationship between chaos and super-Turing behaviors? Although the two are not coextensive, it is at least clear that super-Turing behaviors are pervasive in regions where there is chaos. Moreover, the mathematics of chaos requires infinite precision. Therefore it seems wise to adopt the more general computational framework.

(iii) The super-Turing framework suggests a way of understanding neural learning from a new perspective. In addition to considering generic local properties of a cost function as is done in the derivation of low-level learning procedures [35] or the stability structure of the cost-function dynamics

[53], we may consider the computational landscape that the learning process traverses via parameter-space maps like Figure 6. This approach may shed light on the structural stages of the learning processes and the relationship between the emergentist perspective of many connectionist modelers and the structural perspective of linguistic theory.

6.4 Return to "Compatibility"

Returning to the compatibility issue raised in the Introduction, the current line of argument suggests that dynamical models with real parameters, like recurrent connectionist networks, are, indeed, incompatible with classical symbolic approaches in that their behaviors form a strict superset of the behaviors exhibited by Turing devices.[3] On the other hand, it is noteworthy that the Turing devices among linear dynamical automata are dense in the parameter space (see Figure 6), so, in this subregion of the affine domain, the behavior of any super-Turing computation can well approximated by a Turing mechanism. Perhaps this is why it has seemed sufficient, from the standpoint of classical (structuralist) approaches to cognition, to use Turing computation models: they can approximate observed structures quite well. But the present work suggests that we ought to adopt a super Turing framework for the same reason that one does well to consider the real continuum when formalizing an understanding of concepts like differentiation and integration: the theory of processes is much simplified by working in complete metric spaces—viz. spaces which don't have any points "missing".

Finally, it is also interesting to note that the non-finite state behaviors of the linear dynamical automata come about through a complementarity between the two parameters: they arise when one is expansive and the other is contractive. Furthermore, if we take the complexity of a positive rational number, γ, to be the sum of its numerator and denominator when it is expressed in lowest terms, then, for a linear dynamical automaton satisfying $a_1 = a_2^{-\gamma}$, the complexity of γ gives essentially the number of rules required to write a context free grammar of the automaton. One can thus say that the simplest exponent corresponds to the simplest context free grammar. Whereas the Turing computation framework does not give us any particular insight into why context free computation, among all Turing machine types, seems to be a kind of backbone of cognitive computation, the neurally grounded super-Turing framework considered here suggests an insight: it stems from a fundamental symmetry (multiplicative inversion) of real numbers.

In sum, the super-Turing framework is appealing because it is more general than the Turing machine framework and it allows us to consider relationships between devices that have been invisible in prior work on the structure of cognitive computation.

[3] While the case at hand only shows partial overlap between the two classes of mechanisms, the results of [24] suggest that sufficiently large parameterizable connectionist devices include all Turing computation as a proper subset.

7 Appendix A1

Thm. 1 Let $DA = ([0,1], \{f_1(h) = a_1 h, f_2(h) = a_2 h\}, h_0 > 0)$ be a linear dynamical automaton. If

$$a_1 = a_2^{-\frac{\alpha}{\beta}} \tag{12}$$

for α, β positive integers and $a_1, a_2 \neq 1$. Then $L(DA)$ is a context free language that is not a finite state language.
Proof: Without loss of generality, we can assume $a_1 > 1$ and $a_2 < 1$. Choose two real numbers, m and n satisfying

$$m\beta + n\alpha = 1 \tag{13}$$

For example, we could take $m = 0$ and $n = \frac{1}{\alpha}$. Let

$$u = a_1^n a_2^m \tag{14}$$

Therefore

$$\begin{aligned} a_1 &= a_1^{m\beta - n\alpha} \\ &= a_1^{-n\alpha} a_1^{m\beta} \\ &= a_1^{-n\alpha} (a_1^{-\beta})^{-m} \\ &= (a_1^n a_2^m)^{-\alpha} \\ &= u^{-\alpha} \end{aligned} \tag{15}$$

Similarly

$$a_2 = u^{\beta} \tag{16}$$

Note that $u = a_1^{-1/\alpha}$ where α is a positive integer and $a_1 > 1$, so $u < 1$.
Let PDA be a pushdown automaton with a one-symbol stack alphabet. Whenever PDA reads a 2, it pushes β symbols onto its stack. Whenever PDA reads a 1, it pops α symbols off its stack. For $x \in [0,1]$, let

$$q(x) = \lceil log_u(x) \rceil \tag{17}$$

where $\lceil y \rceil$ denotes the least integer greater than or equal to y. At the start of processing, the stack of PDA has $q(h_0)$ symbols on it. For $\sigma \in \Sigma^{\infty}$, if PDA makes a possible move at every symbol of σ in sequence (i.e., it never encounters a 1 with fewer than α symbols on its stack), then PDA is said to recognize σ. Let $L(PDA)$ be the set of strings recognized by PDA.

Let $\sigma \in \Sigma^{\infty}$ be a sentence of $L(DA)$. Consider $\sigma[n]$ the length n prefix of σ for $n \in N$. Suppose that, upon processing the prefix $\sigma[n]$, PDA has j symbols on its stack and M is at a point $x \in [0,1]$ where $q(x) = j$. Then, if the next symbol is a 2, PDA will add β symbols to its stack. Similarly, the new state of M will be $x'_2 = f_2(x) = a_2 x = u^{\beta} x$. Therefore, $q(x'_2) = \lceil log_u u^{\beta} x \rceil = \beta + \lceil log_u(x) \rceil = j + \beta$. Likewise, if the next symbol is a 1, then PDA will remove α symbols from its stack. Similarly, the new state of M will be $x'_1 = f_1(x) = a_1 x = u^{-\alpha} x$. Therefore, $q(x'_1) = \lceil log_u u^{-\alpha} x \rceil = -\alpha + \lceil log_u(x) \rceil = j - \alpha$. Therefore, since $q(h_0)$ was the number of symbols on the stack at the start of processing, $q(x)$ equals the number of symbols on the stack at every step, provided that every legal move of M is a legal

move of PDA and vice versa. In other words, it only remains to be shown that PDA and DA always generate/accept the same next symbol.

When the stack has α or more symbols on it, both 1 and 2 are possible next inputs. But when the stack has fewer than α symbols on it, only 2 is a possible next input. Likewise, if $q(x) \geq \alpha$ then the state of M is in the domains of both f_1 and f_2, but when $q(x) < \alpha$ then

$$\begin{aligned} \lceil log_u(x) \rceil &< \alpha \\ log_u(x) &< \alpha \\ x &> u^{\alpha} \end{aligned} \tag{18}$$

since $u < 1$ and $\alpha > 0$. Thus

$$\begin{aligned} x^{-1} &< u^{-\alpha} \\ x^{-1} &< a_1 \\ x &> a_1^{-1} \end{aligned} \tag{19}$$

Since a_1^{-1} is the upper bound of the domain of f_1, only f_2 can be applied. Thus $L(M) = L(PDA)$. By a well-known theorem [18], $L(PDA)$ is a context free language. Since PDA can have an unbounded number of symbols on its stack during the processing of legal strings from its language, $L(PDA)$ is not a finite state language. Thus $L(M)$ is a context free language that is not a finite state language. $\diamond$

8 Appendix A2

Lemma Let DA be a set of invertible affine dynamical automata with inverses DA^{-1}. Consider an internal partition boundary, b of DA. If some subset of DA_b^{-1} is dense in an uncountable set in H, then for uncountably many initial states h_0, DA_{h_0} generates a super Turing language.
Proof: Consider the future, F_b, of the internal partition boundary, $b = 1/a_1$ under DA^{-1}. Suppose that some part of this future is dense in an uncountable set. There must be an uncountable number of points that are pairwise separated from one another by points in F_b. Consider any pair of such points, x and y and the separating point, $B \in F_b$. Consider iterating DA, the forward map, simultaneously from the initial points x, y, and B. Since B lies strictly between x and y and the functions of DA are linear, if the same symbol is chosen for each of the three trajectories at every step, then each iterate of the future of B will always lie strictly between each iterate of x and y. Therefore, foor some σ, $\Phi_\sigma(B) = b$. Thus the futures of x and y eventually lie on opposite sides of the internal partition boundary. Therefore the futures of x and y are nonidentical. Consequently $L(DA_x) \neq L(DA_y)$. Since this is true of uncountably many pairs of points, there must be an uncountable number of distinct languages generated by DA. Since the Turing processes are countable, uncountably many of these languages must be Super Turing languages. $\diamond$
Thm. 2 Let $DA = (H = [0,1],\ \{f_1(h) = a_1 h, f_2(h) = a_2 h\})$ be a linear dynamical automaton. If

$$a_1 = a_2^{-\gamma} \tag{20}$$

where γ is a positive irrational number, then there are states $h \in H$ for which $L(DA_h)$ is not generable/recognizable by any Turing device.

Proof: Without loss of generality, assume $a_1 > 1$. By the Lemma, it will suffice to show that the future, F, of the point, $1/a_1$, under DA^{-1} is dense in an interval of positive length. In fact, F is dense in H itself. Let $\alpha_1 = 1/a_1$ and $\alpha_2 = 1/a_2$. Note that $\alpha_1 = \alpha_2^{-\gamma}$. Consider $y = \alpha_1^j \alpha_2^k (1/a_1)$, for j and k nonnegative integers and $k \leq \gamma(j+1)$, a point in the future of $1/a_1$ under DA^{-1}. I will show that for every $x \in H$ and every ϵ, there exists y satisfying the conditions just mentioned, with $|y - x| < \epsilon$. Note that

$$\begin{aligned} log_{\alpha_2} y &= log_{\alpha_2} \alpha_1^j \alpha_2^k (1/a_1) \\ &= log_{\alpha_2} (\alpha_2^{-\gamma})^j \alpha_2^k \alpha_1 \\ &= (log_{\alpha_2} \alpha_1) - \gamma j + k \end{aligned} \tag{21}$$

$f(z) = z - \gamma \ (mod\ 1)$ is an irrational rotation on [0, 1]. Therefore the set $\{z - \gamma j \ (mod\ 1) : j = 0, 1, 2, \ldots\}$ is dense in [0, 1]. Thus there is a nonnegative integer j such that $log_{\alpha_2}\alpha_1 - \gamma j \ (mod\ 1)$ is within ϵ of $log_{\alpha_2} x \ (mod\ 1)$. It follows that there is a nonnegative integer k ($k \leq \gamma(j+1)$) such that $(log_{\alpha_2}\alpha_1) - \gamma j + k$ is within ϵ of $log_{\alpha_2} x$. Since log_{α_2} is expansive on (0, 1), it also follows that $y = \alpha_1^j \alpha_2^k (1/a_1)$ is within ϵ of x. Thus F is dense in [0, 1]. $\diamond$

The foundation for this work was accomplished in collaboration with Dalia Terhesiu. Helpful feedback was provided by the audience of the workshop on Dynamical Systems in Language held at the University of Reading, England, September 8-9, 2008, by the audience of a talk given at the University of Connecticut Logic Group on April 23, 2009, and by two anonymous reviewers.

References

1. Abarbanel, H.D.I., Gilpin, M.E., Rotenberg, M.: Analysis of Observed Chaotic Data. Springer-Verlag, New York (1996)
2. beim Graben, P.: Incompatible implementations of physical symbol systems. Mind and Matter **2**(2), 29–51 (2004)
3. Chomsky, N.: Three models for the description of language. IRE Transactions on Information Theory **2**(3), 113–124 (1956). A corrected version appears in Luce, Bush, and Galanter, eds., 1965 *Readings in Mathematical Psychology, Vol. 2*
4. Devaney, R.L.: An Introduction to Chaotic Dynamical Systems, 2nd ed. Addison-Wesley, Redwood City, CA (1989)
5. Elman, J.L.: Finding structure in time. Cognitive Science **14**, 179–211 (1990)
6. Elman, J.L.: Distributed representations, simple recurrent networks, and grammatical structure. Machine Learning **7**, 195–225 (1991)
7. Elman, J.L., Bates, E.A., Johnson, M.H., Karmiloff-Smith, A., Parisi, D., Plunkett, K.: Rethinking Innateness: A Connectionist Perspective on Development. MIT Press, Cambridge, MA (1996)
8. Fodor, J.A., McLaughlin, B.P.: Connectionism and the problem of constituency: Why smolenksy's solution doesn't work. In: C. MacDonald, G. MacDonald (eds.) Connectionism. Debates on Psychological Explanation, pp. 199–222. Blackwell, Oxford (1995)

9. Fodor, J.A., Pylyshyn, Z.W.: Connectionistm and cognitive architecture: A critical analysis. Cognition **28**, 3–71 (1988)
10. Gazdar, G.: On syntactic categories. Philosophical Transactions (Series B) of the Royal Society **295**, 267–83 (1981)
11. Gerth, S., beim Graben, P.: Unifying syntactic theory and sentence processing difficulty through a connectionist minimalist parser. Cognitive Neurodynamics (2009). Submitted paper
12. beim Graben, P.: Language processing by dynamical systems. International Journal of Bifurcation and Chaos **14**(2), 599–621 (2004)
13. beim Graben, P., Atmanspracher, H.: Complementarity in classical dynamical systems. Foundations of Physics **36**(2) (2006)
14. beim Graben, P., Gerth, S., Vasishth, S.: Towards dynamical systems models of language-related brain potentials. Cognitive Neurodynamics **2**, 229–255 (2008)
15. beim Graben, P., Pinotsis, D., Saddy, D., Potthast, R.: Language processing with dynamical fields. Cognitive Neurodynamics **2**, 79–88 (2008)
16. beim Graben, P., Potthast, R.: Inverse problems in dynamical cognitive modeling. Chaos **19**, 015,103 (2009)
17. Harm, M.W., Seidenberg, M.S.: Computing the meanings of words in reading: cooperative division of labor between visual and phonological processes. Psychological Review **111**(3), 662–720 (2004)
18. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Menlo Park, California (1979)
19. Hopper, P.J., Traugott, E.C.: Grammaticalization, 2nd Edition. Cambridge University Press, Cambridge, England (2003)
20. Hyams, N.: Nondiscreteness and variation in child language: Implications for principle and parameter models of language development. In: Y. Levy (ed.) Other Children, Other Languages, pp. 11–40. Lawrence Erlbaum (1995)
21. Joshi, A.K., Schabes, Y.: Tree-adjoining grammars. In: G. Rozenberg, A. Salomaa (eds.) Handbook of Formal Languages, vol. 3, pp. 69–123. Springer-Verlag, New York (1996)
22. McClelland, J.L., Patterson, K.: Rules or connections in past tense inflection: What does the evidence rule out? Trends in Cognitive Science **6**(11), 465–472 (2002)
23. Montague, R.: English as a formal language. In: B.V. et al. (ed.) Linguaggi nella e nella Tecnica, pp. 189–224. Edizioni di Comunità (1970). Reprinted in *Formal Philosophy: Selected Papers of Richard Montague*, pp. 108–221, ed. by R. H. Thomason, New Haven: Yale University Press, 1974.
24. Moore, C.: Dynamical recognizers: Real-time language recognition by analog computers. Theoretical Computer Science **201**, 99–136 (1998)
25. Pinker, S., Ullman, M.T.: Combination and structure, not gradedness, is the issue. Trends in Cognitive Science **6**(11), 472–474 (2002)
26. Plaut, D.C., McClelland, J.L., Seidenberg, M.S., Patterson, K.E.: Understanding normal and impaired word reading: Computational principles in quasi-regular domains. Psychological Review **103**, 56–115 (1996)
27. Plunkett, K., Marchman, V.: U-shaped learning and frequency effects in a multi-layer perceptron: Implications for child language acquisition. Cognition **38**, 43–102 (1991)
28. Pollack, J.: On connectionist models of natural language processing (1987). Unpublished doctoral dissertation, University of Illinois.
29. Pollack, J.B.: The induction of dynamical recognizers. Machine Learning **7**, 227–252 (1991)
30. Rodriguez, P.: Representing the structure of a simple context-free language in a recurrent neural network: A dynamical systems approach (1995). On-line Newsletter of the Center for Research on Language, University of California, San Diego.
31. Rodriguez, P.: Simple recurrent networks learn context-free and context-sensitive languages by counting. Neural Computation **13**(9) (2001)
32. Rodriguez, P., Wiles, J., Elman, J.: A recurrent neural network that learns to count. Connection Science **11**(1), 5–40 (1999)

33. Rohde, D.: A connectionist model of sentence comprehension and production (2002). Unpublished PhD Dissertation, Carnegie Mellon University
34. Rohde, D., Plaut, D.: Language acquisition in the absence of explicit negative evidence: How important is starting small? Journal of Memory and Language **72**, 67–109 (1999)
35. Rumelhart, D., Durbin, R., Golden, R., Chauvin, Y.: Backpropagation: The basic theory. In: Backpropagation: Theory, Architectures, and Applications. Lawrence Erlbaum Associates (1995)
36. Rumelhart, D.E., McClelland, J.L.: On learning the past tenses of english verbs. In: D.E. Rumelhart, J.L. McClelland (eds.) Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1. MIT Press, Cambridge, MA (1986)
37. Savitch, W.J. (ed.): The Formal Complexity of Natural Language. Kluwer, Norwell, MA (1987)
38. Seidenberg, M.S., Gonnerman, L.M.: Explaining derivational morphology as the convergence of codes. Trends in Cognitive Sciences **4**(9), 353–361 (2000)
39. Seidenberg, M.S., McClelland, J.L.: A distributed, developmental model of word recognition and naming. Psychological Review **96**, 447–452 (1989)
40. Siegelmann, H.T.: Neural Networks and Analog Computation: Beyond the Turing Limit. Birkhäuser, Boston (1999)
41. Smolensky, P.: On the proper treatment of connectionism. Behavioral and Brain Sciences **11**(1), 1–74 (1988)
42. Smolensky, P.: Tensor product variable binding and the representation of symbolic structures in connectionist systems. Artificial Intelligence **46** (1990). Special issue on Connectionist symbol processing edited by G. E. Hinton
43. Smolensky, P.: Connectionism, constituency, and the language of thought. In: C. MacDonald, G. MacDonald (eds.) Connectionism. Debates on Psychological Explanation. Blackwell, Oxford (1995)
44. Smolensky, P.: Reply: Constituent structure and explanation in an integrated connectionist/symbolic architecture. In: C. MacDonald, G. MacDonald (eds.) Connectionism. Debates on Psychological Explanation. Blackwell, Oxford (1995)
45. Snyder, W.: Child Language: The Parametric Approach. Oxford University Press, London (2007)
46. Tabor, W.: Syntactic innovation: A connectionist model (1994). Ph.D. dissertation, Stanford University
47. Tabor, W.: Lexical change as nonlinear interpolation. In: J.D. Moore, J.F. Lehman (eds.) Proceedings of the 17th Annual Cognitive Science Conference. Lawrence Erlbaum Associates (1995)
48. Tabor, W.: Fractal encoding of context-free grammars in connectionist networks. Expert Systems: The International Journal of Knowledge Engineering and Neural Networks **17**(1), 41–56 (2000)
49. Tabor, W.: Learning exponential state growth languages by hill climbing. IEEE Transactions on Neural Networks **14**(2), 444–446 (2003)
50. Tabor, W., Terhesiu, D.: On the relationship between symbolic and neural computation (2004). AAAI Technical Report FS-04-03. ISBN 1-57735-214-9
51. Thomas, M., Karmiloff-Smith, A.: Can developmental disorders reveal the component parts of the human language faculty? Language Learning and Development **1**(1), 65–92 (2005)
52. Turing, A.: Systems of logic based on ordinals. Proceedings of the London Mathematical Society, Series 2 **45**, 161–228 (1939)
53. Wei, H., Zhang, J., Cousseau, F., Ozeki, T., Amari, S.: Dynamics of learning near singularities in layered networks. Neural Computation **20**, 813–843 (2008)
54. Wiles, J., Elman, J.: Landscapes in recurrent networks. In: J.D. Moore, J.F. Lehman (eds.) Proceedings of the 17th Annual Cognitive Science Conference. Lawrence Erlbaum Associates (1995)