Chapter 11
Universal Neural Field Computation

Peter beim Graben and Roland Potthast

Abstract Turing machines and Gödel numbers are important pillars of the theory of computation. Thus, any computational architecture needs to show how it could relate to Turing machines and how stable implementations of Turing computation are possible. In this chapter, we implement universal Turing computation in a neural field environment. To this end, we employ the canonical symbologram representation of a Turing machine obtained from a Gödel encoding of its symbolic repertoire and generalized shifts. The resulting nonlinear dynamical automaton (NDA) is a piecewise affine-linear map acting on the unit square that is partitioned into rectangular domains. Instead of looking at point dynamics in phase space, we then consider functional dynamics of probability distribution functions (p.d.f.s) over phase space. This is generally described by a Frobenius-Perron integral transformation that can be regarded as a neural field equation over the unit square as feature space of a Dynamic Field Theory (DFT). Solving the Frobenius-Perron equation yields that uniform p.d.f.s with rectangular support are mapped onto uniform p.d.f.s with rectangular support, again. We call the resulting representation *dynamic field automaton*.

P. beim Graben (✉)
Department of German Studies and Linguistics, Humboldt-Universität zu Berlin, Berlin, Germany

Bernstein Center for Computational Neuroscience Berlin, Humboldt-Universität zu Berlin, Berlin, Germany
e-mail: peter.beim.graben@hu-berlin.de

R. Potthast
Department of Mathematics and Statistics, University of Reading, Reading, Berkshire, UK

Deutscher Wetterdienst, Offenbach, Germany

11.1 Introduction

Studying the computational capabilities of neurodynamical systems has commenced with the groundbreaking 1943 article of McCulloch and Pitts [27] on networks of idealized two-state neurons that essentially behave as logic gates. Because nowadays computers are nothing else than large-scale networks of logic gates, it is clear that computers can in principle be build up by neural networks of McCulloch-Pitts units. This has also been demonstrated by a number of theoretical studies reviewed in [46]. However, even the most powerful modern workstation is, from a mathematical point of view, only a finite state machine due to its rather huge, though limited memory, while a universal computer, formally codified as a Turing machine [20,51], possesses an unbounded memory tape.

Using continuous-state units with a sigmoidal activation function, Siegelmann and Sontag [43] were able to prove that a universal Turing machine can be implemented by a recurrent neural network of about 900 units, most of them describing the machine's control states, while the tape is essentially represented by a plane spanned by the activations of just two units. The same construction, employing a Gödel code [17,19] for the tape symbols, has been previously used by Moore [29,30] for proving the equivalence of nonlinear dynamical automata and Turing machines. Along a different vain, deploying sequential cascaded networks, Pollack [36] and later Moore [31] and Tabor [48, 49] introduced and further generalized dynamical automata as nonautonomous dynamical systems. An even further generalization of dynamical automata, where the tape space becomes represented by a function space, led Moore and Crutchfield [32] to the concept of a quantum automaton (see [6] for a review and some unified treatment of these different approaches).

Quite remarkably, another paper from McCulloch and Pitts published in 1947 [34] already set up the groundwork for such functional representations in continuous neural systems. Here, those pioneers investigated distributed neural activation over cortical or subcortical maps representing visual or auditory feature spaces. These neural fields are copied onto many layers, each transforming the field according to a particular member of a symmetry group. For these, a number of field functionals is applied to yield a group invariant that serves for subsequent pattern detection. As early as in this publication, we already find all necessary ingredients for a *Dynamic Field Architecture*: a layered system of neural fields defined over appropriate feature spaces [14, 42] (see also Chaps. 12 and 13 in this volume).

We begin this chapter with a general exposition of dynamic field architectures in Sect. 11.2 where we illustrate how variables and structured data types on the one hand and algorithms and sequential processes on the other hand can be implemented in such environments. In Sect. 11.3 we review known facts about nonlinear dynamical automata and introduce dynamic field automata from a different perspective. The chapter is concluded with a short discussion about universal computation in neural fields.

11.2 Principles of Universal Computation

As already suggested by McCulloch and Pitts [34] in 1947, a neural, or likewise, *dynamic field architecture* is a layered system of Dynamic Neural Fields $u_i(x,t) \in \mathbb{R}$ where $1 \le i \le n$ $(i, n \in \mathbb{N})$ indicates the layer, $x \in D$ denotes spatial position in a suitable d-dimensional *feature space* $D \subset \mathbb{R}^d$ and $t \in \mathbb{R}_0^+$ time. Usually, the fields obey the Amari neural field equation [2]

$$\tau_i \frac{\partial u_i(x,t)}{\partial t} = -u_i(x,t) + h(x) + \sum_{j=1}^{n} \int_D w_{ij}(x,y) f(u_j(y,t)) \, \mathrm{d}y + p_i(x,t), \quad (11.1)$$

where τ_i is a characteristic time scale of the i-th layer, $h(x)$ the unique resting activity, $w_{ij}(x,y)$ the synaptic weight kernel for a connection to site x in layer i from site y in layer j,

$$f(u) = \frac{1}{1 + \mathrm{e}^{-\beta(u-\theta)}} \quad (11.2)$$

is a sigmoidal activation function with gain β and threshold θ, and $p_i(x,t)$ external input delivered to site x in layer i at time t. Note, that a two-layered architecture could be conveniently described by a one-layered complex neural field $z(x,t) = u_1(x,t) + \mathrm{i} u_2(x,t)$ as used in [6, 7, 11].

Commonly, Eq. (11.1) is often simplified in the literature by assuming one universal time constant τ, by setting $h = 0$ and by replacing p_i through appropriate initial, $u_i(x,0)$, and boundary conditions, $u_i(\partial D, t)$. With these simplifications, we have to solve the Amari equation

$$\tau \frac{\partial u_i(x,t)}{\partial t} = -u_i(x,t) + \sum_{j=1}^{n} \int_D w_{ij}(x,y) f(u_j(y,t)) \, \mathrm{d}y \quad (11.3)$$

for initial condition $u_i(x,0)$, stating a computational task. Solving that task is achieved through a transient dynamics of Eq. (11.3) that eventually settles down either in an attractor state or in a distinguished terminal state $U_i(x,T)$, after elapsed time T. Mapping one state into another, which again leads to a transition to a third state and so on, we will see how the field dynamics can be interpreted as a kind of universal computation, carried out by a program encoded in the particular kernels $w_{ij}(x,y)$, which are in general heterogeneous, i.e. they are not pure convolution kernels: $w_{ij}(x,y) \neq w_{ij}(||x-y||)$ [5, 22].

11.2.1 Variables and Data Types

How can *variables* be realized in a neural field environment? At the hardware-level of conventional digital computers, variables are sequences of bytes stored in random access memory (RAM). Since a byte is a word of eight bits and since nowadays RAM chips have about 2–8 GB, the computer's memory appears as an approximately $8 \times 4 \cdot 10^9$ binary matrix, similar to an image of black-white pixels. It seems plausible to regard this RAM image as a discretized neural field, such that the value of $u(x,t)$ at $x \in D$ could be interpreted as a particular instantiation of a variable. However, this is not tenable for at least two reasons. First, such variables would be highly volatile as bits might change after every processing cycle. Second, the required function space would be a 'mathematical monster' containing highly discontinuous functions that are not admitted for the dynamical law (11.3). Therefore, variables have to be differently introduced into neural field computers by assuring temporal stability and spatial smoothness.

We first discuss the second point. Possible solutions of the neural field equation (11.3) must belong to appropriately chosen function spaces that allow the storage and retrieval of variables through binding and unbinding operations. A variable is stored in the neural field by binding its value to an address and its value is retrieved by the corresponding unbinding procedure. These operations have been described in the framework of *Vector Symbolic Architectures* [16, 44] and applied to Dynamic Neural Fields by beim Graben and Potthast [6] through a three-tier top-down approach, called *Dynamic Cognitive Modeling*, where variables are regarded as instantiations of data types of arbitrary complexity, ranging from primitive data types such as characters, integers, or floating numbers, over arrays (strings, vectors and matrices) of those primitives, up to structures and objects that allow the representation of lists, frames or trees. These data types are in a first step decomposed into *filler/role bindings* [44] which are sets of ordered pairs of sets of ordered pairs etc., of so-called fillers and roles. Simple fillers are primitives whereas roles address the appearance of a primitive in a complex data type. These addresses could be, e.g., array indices or tree positions. Such filler/role bindings can recursively serve as complex fillers bound to other roles. In a second step, fillers and roles are identified with particular basis functions over suitable feature spaces while the binding is realized through functional tensor products with subsequent compression (e.g. by means of convolution products) [35, 45].

Since the complete variable allocation of a conventional digital computer can be viewed as an instantiation of only one complex data type, namely an array containing every variable at a particular address, it is possible to map a total variable allocation onto a compressed tensor product in function space of a dynamic field architecture. Assuming that the field u encodes such an allocation, a new variable φ in its functional tensor product representation is stored by binding it first to a new address ψ, yielding $\varphi \otimes \psi$ and second by superimposing it with the current allocation, i.e. $u + \varphi \otimes \psi$. Accordingly, the value of φ is retrieved through an unbinding $\langle \psi^+, u \rangle$ where ψ^+ is the adjoint of the address ψ where φ is bound to. These operations require further underlying structure of the employed function spaces that are therefore chosen as Banach or Hilbert spaces where either adjoint or bi-orthogonal basis functions are available (see [4, 6, 7, 11, 38] for examples).

The first problem was the volatility of neural fields. This has been resolved using attractor neural networks [18, 21] where variables are stabilized as asymptotically stable fixed points (as in Chap. 3). Since a fixed point is defined through $\dot{u}_i(x,t) = 0$, the field obeys the equation

$$u_i(x,t) = \sum_{j=1}^{n} \int_D w_{ij}(x,y) f(u_j(y,t)) \, \mathrm{d}y \,. \tag{11.4}$$

This is often achieved by means of a particularly chosen kernel $w_{ii}(||x - y||)$ with local excitation and global inhibition, often called lateral inhibition kernels [14, 42].

11.2.2 Algorithms and Sequential Processes

Conventional computers run programs that dynamically change variables. Programs perform algorithms that are sequences of instructions, including operations upon variables, decisions, loops, etc. From a mathematical point of view, an algorithm is an element of an abstract algebra that has a representation as an operator on the space of variable allocations, which is well-known as *denotational semantics* in computer science [50]. The algebra product is the concatenation of instructions being preserved in the representation which is thereby an algebra homomorphism [4, 6]. Concatenating instructions or composing operators takes place step-by-step in discrete time. Neural field dynamics, as governed by Eq. (11.3), however requires continuous time. How can sequential algorithms be incorporated into the continuum of temporal evolution?

Looking first at conventional digital computers again suggests a possible solution: computers are clocked. Variables remain stable during a clock cycle and gating enables instructions to access variable space. A similar approach has recently been introduced to dynamic field architectures by Sandamirskaya and Schöner [40, 41]. Here a sequence of neural field activities is stored in a stack of layers, each stabilized by a lateral inhibition kernel. One state is destabilized by a gating signal provided by a condition-of-satisfaction mechanism playing the role of the 'clock' in this account. Afterwards, the decaying pattern in one layer, excites the next relevant field in a subsequent layer.

Another solution, already outlined in our Dynamic Cognitive Modeling framework [6], identifies the intermediate results of a computation with saddle fields that are connected along their respective stable and unstable manifolds to form stable heteroclinic sequences [1, 5, 39]. We have utilized this approach in [7] for a dynamic field model of syntactic language processing. Moreover, the chosen model of winnerless competition among neural populations [15] allowed us to explicitly construct the synaptic weight kernel from the filler/role binding of syntactic phrase structure trees [7].

11.3 Dynamic Field Automata

In this section we elaborate our recent proposal on *dynamic field automata* [8] by crucially restricting function spaces to spaces with Haar bases which are piecewise constant fields $u(x,t)$ for $x \in D$, i.e.

$$u(x,t) = \begin{cases} \alpha(t) & : \quad x \in A(t) \\ 0 & : \quad x \notin A(t) \end{cases} \tag{11.5}$$

with some time-dependent amplitude $\alpha(t)$ and a possibly time-dependent domain $A(t) \subset D$. Note, that we consider only one-layered neural fields in the sequel for the sake of simplicity.

For such a choice, we first observe that the application of the nonlinear activation function f yields another piecewise constant function over D:

$$f(u(x,t)) = \begin{cases} f(\alpha(t)) & : \quad x \in A(t) \\ f(0) & : \quad x \notin A(t)\,, \end{cases} \tag{11.6}$$

which can be significantly simplified by the choice $f(0) = 0$, that holds, e.g., for the linear identity $f = \mathrm{id}$, for the Heaviside step function $f = \Theta$ or for the hyperbolic tangens, $f = \tanh$.

With this simplification, the input integral of the neural field becomes

$$\int_D w(x,y) f(u(y,t))\,\mathrm{d}y = \int_{A(t)} w(x,y) f(\alpha(t))\,\mathrm{d}y = f(\alpha(t)) \int_{A(t)} w(x,y)\,\mathrm{d}y\,. \tag{11.7}$$

When we additionally restrict ourselves to piecewise constant kernels as well, the last integral becomes

$$\int_{A(t)} w(x,y)\,\mathrm{d}y = w|A(t)| \tag{11.8}$$

with w as constant kernel value and $|A(t)|$ the measure (i.e. the volume) of the domain $A(t)$. Inserting (11.7) and (11.8) into the fixed point equation (11.4) yields

$$u_0 = |A(t)| \cdot w \cdot f(u_0) \tag{11.9}$$

for the fixed point u_0. Next, we carry out a linear stability analysis

$$\begin{aligned} \dot{u} &= -u + |A(t)| w f(u) \qquad (11.10) \\ &= -(u_0 + (u - u_0)) + |A(t)| w \Big(f(u_0) + f'(u_0) \cdot (u - u_0) \Big) + O(|u - u_0|^2) \\ &= \Big(-1 + |A(t)| w f'(u_0) \Big) \cdot (u - u_0) + O(|u - u_0|^2)\,. \end{aligned}$$

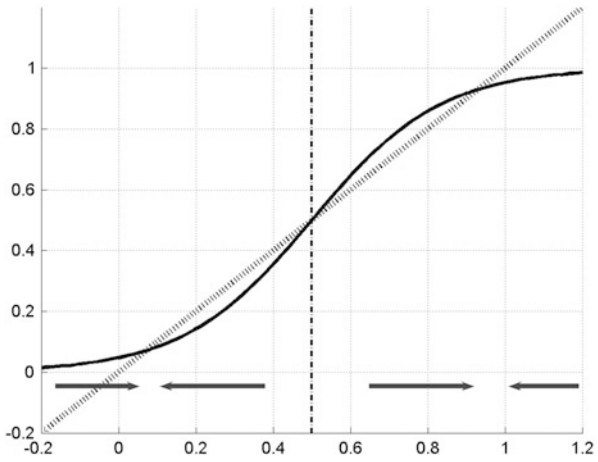

Fig. 11.1 Stability of piecewise constant neural field $u_0(x,t)$ over a domain $A \subset D$. Shown are the sigmoidal activation function $f(u)$ (*solid*) and u (*dotted*) for comparison. The axes here are given in terms of absolute numbers without unit as employed in Eqs. (11.2) or (11.3)

Thus, we conclude that if $|A(t)|wf'(u_0) < 1$, then $\dot{u} < 0$ for $u > u_0$ and conversely, $\dot{u} > 0$ for $u < u_0$ in a neighborhood of u_0, such that u_0 is an asymptotically stable fixed point of the neural field equation.

Of course, linear stability analysis is a standard tool to investigate the behavior of dynamic fields around fixed points [5]. For our particular situation it is visualized in Fig. 11.1. When the solid curve displaying $|A(t)|wf(u)$ is above u (the dotted curve), then the dynamics (11.10) leads to an increase of u, indicated by the arrows pointing to the right. In the case where $|A(t)|wf(u) < u$, a decrease of u is obtained from (11.10). This is indicated by the arrows pointing to the left. When we have three points where the curves coincide, Fig. 11.1 shows that the setting leads to two stable fixed-points of the dynamics. When the activity field $u(x)$ reaches any value close to these fixed points, the dynamics leads them to the fixed-point values u_0.

11.3.1 Turing Machines

For the construction of dynamic field automata through neural fields we consider discrete time that might be supplied by some clock mechanism. This requires the stabilization of the fields (11.5) within one clock cycle which can be achieved by self-excitation with a nonlinear activation function f as described in (11.10), leading to stable excitations as long as we do not include inhibitive elements, where a subsequent state would inhibit those states which were previously excited.

Next we briefly summarize some concepts from theoretical computer science [6, 20, 51]. A *Turing machine* is formally defined as a 7-tuple $M_{TM} = (Q, \mathbf{N}, \mathbf{T}, \delta, q_0, b, F)$, where Q is a finite set of machine control states, $\mathbf{N}$ is another finite set of tape symbols, containing a distinguished 'blank' symbol b, $\mathbf{T} \subset \mathbf{N} \setminus \{b\}$ is input alphabet, and

$$\delta : Q \times \mathbf{N} \to Q \times \mathbf{N} \times \{L, R\} \tag{11.11}$$

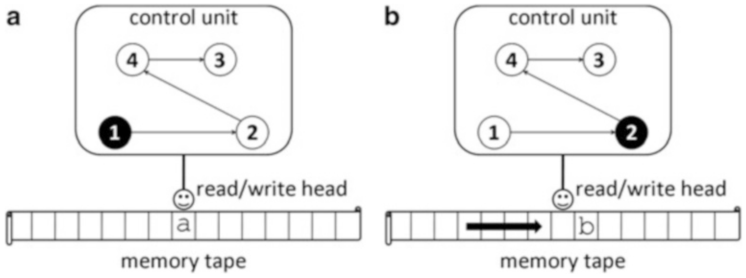

Fig. 11.2 Example state transition from (**a**) to (**b**) of a Turing machine with $\delta(1, \texttt{a}) = (2, \texttt{b}, R)$

is a partial state transition function, the so-called 'machine table', determining the action of the machine when $q \in Q$ is the current state at time t and $a \in \mathbf{N}$ is the current symbol at the memory tape under the read/write head. The machine moves then into another state $q' \in Q$ at time $t + 1$ replacing the symbol a by another symbol $a' \in \mathbf{N}$ and shifting the tape either one place to the left ('L') or to the right ('R'). Figure 11.2 illustrates such a state transition. Finally, $q_0 \in Q$ is a distinguished initial state and $F \subset Q$ is a set of 'halting states' that are assumed when a computation terminates [20].

A Turing machine becomes a time- and state-discrete dynamical system by introducing *state descriptions*, which are triples

$$s = (\alpha, q, \beta) \tag{11.12}$$

where $\alpha, \beta \in \mathbf{N}^*$ are strings of tape symbols to the left and to the right from the head, respectively. $\mathbf{N}^*$ contains all strings of tape symbols from $\mathbf{N}$ of arbitrary, yet finite, length, delimited by blank symbols b. Then, the transition function can be extended to state descriptions by

$$\delta^* : S \to S \,, \tag{11.13}$$

where $S = \mathbf{N}^* \times Q \times \mathbf{N}^*$ now plays the role of a phase space of a discrete dynamical system. The set of tape symbols and machine states then becomes a larger alphabet $\mathbf{A} = \mathbf{N} \cup Q$.

Moreover, state descriptions can be conveniently expressed by means of bi-infinite 'dotted sequences'

$$s = \dots a_{i_{-3}} a_{i_{-2}} a_{i_{-1}} . a_{i_0} a_{i_1} a_{i_2} \dots \tag{11.14}$$

with symbols $a_{i_k} \in \mathbf{A}$. In Eq. (11.14) the dot denotes the observation time $t = 0$ such that the symbol left to the dot, $a_{i_{-1}}$, displays the current state, dissecting the string s into two one-sided infinite strings $s = (\alpha', \beta)$ with $\alpha' = a_{i_{-1}} a_{i_{-2}} a_{i_{-3}} \dots$ as the left-hand part in reversed order and $\beta = a_{i_0} a_{i_1} a_{i_2} \dots$

In symbolic dynamics, a cylinder set [28] is a subset of the space $\mathbf{A}^{\mathbb{Z}}$ of bi-infinite sequences from an alphabet $\mathbf{A}$ that agree in a particular building block of length $n \in \mathbb{N}$ from a particular instance of time $t \in \mathbb{Z}$, i.e.

$$C(n,t) = [a_{i_1}, \dots, a_{i_n}] = \{s \in \mathbf{A}^{\mathbb{Z}} \,|\, s_{t+k-1} = a_{i_k}, \quad k = 1, \dots, n\} \qquad (11.15)$$

is called n-cylinder at time $t \in \mathbb{Z}$. When now $t < 0, n > |t| + 1$ the cylinder contains the dotted word $w = s_{-1}.s_0$ and can therefore be decomposed into a pair of cylinders $(C'(|t|,t), C(|t| + n - 1, 0))$ where C' denotes reversed order of the defining strings again.

A *generalized shift* [29,30] emulating a Turing machine is a pair $M_{GS} = (\mathbf{A}^{\mathbb{Z}}, \Psi)$ where $\mathbf{A}^{\mathbb{Z}}$ is the space of dotted sequences with $s \in \mathbf{A}^{\mathbb{Z}}$ and $\Psi : \mathbf{A}^{\mathbb{Z}} \to \mathbf{A}^{\mathbb{Z}}$ is given as

$$\Psi(s) = \sigma^{F(s)}(s \oplus G(s)) \qquad (11.16)$$

with

$$F : \mathbf{A}^{\mathbb{Z}} \to \mathbb{Z} \qquad (11.17)$$

$$G : \mathbf{A}^{\mathbb{Z}} \to \mathbf{A}^{e} \,, \qquad (11.18)$$

where $\sigma : \mathbf{A}^{\mathbb{Z}} \to \mathbf{A}^{\mathbb{Z}}$ is the left-shift known from symbolic dynamics [26], $F(s) = l$ dictates a number of shifts to the right ($l < 0$), to the left ($l > 0$) or no shift at all ($l = 0$), $G(s)$ is a word w' of length $e \in \mathbb{N}$ in the domain of effect (DoE) replacing the content $w \in \mathbf{A}^{d}$, which is a word of length $d \in \mathbb{N}$, in the domain of dependence (DoD) of s, and $s \oplus G(s)$ denotes this replacement function.

A generalized shift becomes a Turing machine by interpreting $a_{i_{-1}}$ as the current control state q and a_{i_0} as the tape symbol currently underneath the head. Then the remainder of α is the tape left to the head and the remainder of β is the tape right to the head. The DoD is the word $w = a_{i_{-1}}.a_{i_0}$ of length $d = 2$.

As an instructive example we consider a toy model of syntactic language processing. In order to process a sentence such as "the dog chased the cat", linguists often derive a *context-free grammar* (CFG) from a phrase structure tree (see [10] for a more detailed example). In our case such a CFG could consist of *rewriting rules*

$$\texttt{S} \to \texttt{NP VP} \qquad (11.19)$$

$$\texttt{VP} \to \texttt{V NP} \qquad (11.20)$$

$$\texttt{NP} \to \texttt{the dog} \qquad (11.21)$$

$$\texttt{V} \to \texttt{chased} \qquad (11.22)$$

$$\texttt{NP} \to \texttt{the cat} \qquad (11.23)$$

where the left-hand side always presents a nonterminal symbol to be expanded into a string of nonterminal and terminal symbols at the right-hand side. Omitting the

Table 11.1 Sequence of state transitions of the generalized shift processing the well-formed string "the dog chased the cat" (`NP V NP`). The operations are indicated as follows: "predict (X)" means prediction according to rule (X) of the context-free grammar; attach means cancelation of successfully predicted terminals both from stack and input; and "accept" means acceptance of the string as being well-formed

Time	State	Operation
0	`S . NP V NP`	predict (11.19)
1	`VP NP . NP V NP`	attach
2	`VP . V NP`	predict (11.20)
3	`NP V . V NP`	attach
4	`NP . NP`	attach
5	ϵ . ϵ	accept

lexical rules (11.21–11.23), we regard the symbols `NP`, `V`, denoting 'noun phrase' and 'verb', respectively, as terminals and the symbols `S` ('sentence') and `VP` ('verbal phrase') as nonterminals.

A generalized shift processing this grammar is then prescribed by the mappings

$$\begin{aligned} \mathtt{S}.a &\mapsto \mathtt{VP}\;\mathtt{NP}.a \\ \mathtt{VP}.a &\mapsto \mathtt{NP}\;\mathtt{V}.a \\ a.a &\mapsto \epsilon.\epsilon \end{aligned} \tag{11.24}$$

where the left-hand side of the tape is now called 'stack' and the right-hand side 'input'. In (11.24) $a \in \mathbf{T}$ stands for an arbitrary input symbol. The empty word is indicated by ϵ. Note the reversed order for the stack left of the dot. The first two operations in (11.24) are *predictions* according to a rule of the CFG while the last one is an attachment of input material with already predicted material, to be understood as a matching step.

With this machine table, a *parse* of the sentence "the dog chased the cat" (`NP V NP`) is then obtained in Table 11.1.

11.3.2 Nonlinear Dynamical Automata

Applying a Gödel encoding [6, 17, 19]

$$\begin{aligned} x = \psi(\alpha') &:= \sum_{k=1}^{\infty} \psi(a_{i_{-k}}) b_L^{-k} \\ y = \psi(\beta) &:= \sum_{k=0}^{\infty} \psi(a_{i_k}) b_R^{-k-1} \end{aligned} \tag{11.25}$$

to the pair $s = (\alpha', \beta)$ from the Turing machine state description (11.14) where $\psi(a_j) \in \mathbb{N}_0$ is an integer Gödel number for symbol $a_j \in \mathbf{A}$ and $b_L, b_R \in \mathbb{N}$ are the numbers of symbols that could appear either in α' or in β, respectively, yields the so-called symbol plane or *symbologram representation* $\mathbf{x} = (x, y)^T$ of s in the unit square X [13, 23].

The symbologram representation of a generalized shift is a *nonlinear dynamical automaton* (NDA) [6, 9, 10] which is a triple $M_{NDA} = (X, \mathscr{P}, \Phi)$ where (X, Φ) is a time-discrete dynamical system with phase space $X = [0,1]^2 \subset \mathbb{R}^2$, the unit square, and flow $\Phi : X \to X$. $\mathscr{P} = \{D_\nu | \nu = (i, j), 1 \leq i \leq m, 1 \leq j \leq n, m, n \in \mathbb{N}\}$ is a rectangular partition of X into pairwise disjoint sets, $D_\nu \cap D_\mu = \emptyset$ for $\nu \neq \mu$, covering the whole phase space $X = \bigcup_\nu D_\nu$, such that $D_\nu = I_i \times J_j$ with real intervals $I_i, J_j \subset [0,1]$ for each bi-index $\nu = (i, j)$. The cells D_ν are the domains of the branches of Φ which is a piecewise affine-linear map

$$\Phi(\mathbf{x}) = \begin{pmatrix} a_x^\nu \\ a_y^\nu \end{pmatrix} + \begin{pmatrix} \lambda_x^\nu & 0 \\ 0 & \lambda_y^\nu \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} , \tag{11.26}$$

when $\mathbf{x} = (x, y)^T \in D_\nu$. The vectors $(a_x^\nu, a_y^\nu)^T \in \mathbb{R}^2$ characterize parallel translations, while the matrix coefficients $\lambda_x^\nu, \lambda_y^\nu \in \mathbb{R}_0^+$ mediate either stretchings ($\lambda > 1$), squeezings ($\lambda < 1$), or identities ($\lambda = 1$) along the x- and y-axes, respectively.

Hence, the NDA's dynamics, obtained by iterating an orbit $\{\mathbf{x}_t \in X | t \in \mathbb{N}_0\}$ from initial condition $\mathbf{x}_0$ through

$$\mathbf{x}_{t+1} = \Phi(\mathbf{x}_t) \tag{11.27}$$

describes a symbolic computation by means of a generalized shift [29, 30] when subjected to the coarse-graining $\mathscr{P}$.

The domains of dependence and effect (DoD and DoE) of an NDA, respectively, are obtained as images of cylinder sets under the Gödel encoding (11.25). Each cylinder possesses a lower and an upper bound, given by the Gödel numbers 0 and $b_L - 1$ or $b_R - 1$, respectively. Thus,

$$\begin{aligned}
\inf(\psi(C'(|t|, t))) &= \psi(a_{i_{|t|}}, \ldots, a_{i_1}) \\
\sup(\psi(C'(|t|, t))) &= \psi(a_{i_{|t|}}, \ldots, a_{i_1}) + b_L^{-|t|} \\
\inf(\psi(C(|t| + n - 1, 0))) &= \psi(a_{i_{|t|+1}}, \ldots, a_{i_n}) \\
\sup(\psi(C(|t| + n - 1, 0))) &= \psi(a_{i_{|t|+1}}, \ldots, a_{i_n}) + b_R^{-|t|-n+1} ,
\end{aligned}$$

where the suprema have been evaluated by means of geometric series [9]. Thereby, each part cylinder C is mapped onto a real interval $[\inf(C), \sup(C)] \subset [0,1]$ and the complete cylinder $C(n, t)$ onto the Cartesian product of intervals $R = I \times J \subset [0,1]^2$, i.e. onto a rectangle in unit square. In particular, the empty cylinder, corresponding to the empty tape $\epsilon.\epsilon$ is represented by the complete phase space $X = [0,1]^2$.

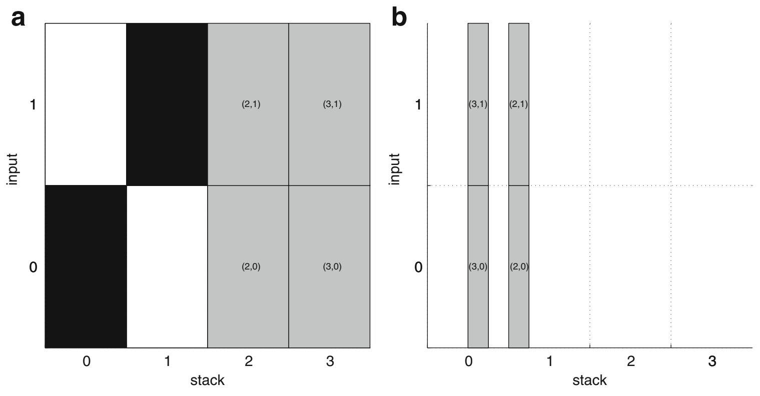

Fig. 11.3 Symbologram of the NDA processing the string "the dog chased the cat" (`NP V NP`). (**a**) Domains of dependence (DoD) of actions: identity (*white*), predict (*gray*), and attach (*black*). (**b**) Domains of effect (DoE): images of prediction (*gray*), *black rectangles* from (**a**) are mapped onto the whole unit square during attachment

Fixing the prefixes of both part cylinders and allowing for random symbolic continuation beyond the defining building blocks, results in a cloud of randomly scattered points across a rectangle R in the symbologram [10]. These rectangles are consistent with the symbol processing dynamics of the NDA, while individual points $\mathbf{x} \in [0,1]^2$ no longer have an immediate symbolic interpretation. Therefore, we refer to arbitrary rectangles $R \in [0,1]^2$ as to NDA macrostates, distinguishing them from NDA microstates $\mathbf{x}$ of the underlying dynamical system.

Coming back to our language example, we create an NDA from an arbitrary Gödel encoding. Choosing

$$\psi(\mathtt{NP}) = 0 \tag{11.28}$$

$$\psi(\mathtt{V}) = 1 \tag{11.29}$$

$$\psi(\mathtt{VP}) = 2 \tag{11.30}$$

$$\psi(\mathtt{S}) = 3 \tag{11.31}$$

we have $b_L = 4$ stack symbols and $b_R = 2$ input symbols. Thus, the symbologram is partitioned into eight rectangles. Figure 11.3 displays the resulting (a) DoD and (b) DoE.

11.3.3 Neural Field Computation

Next we replace the NDA point dynamics in phase space by functional dynamics in Banach space. Instead of iterating clouds of randomly prepared initial conditions according to a deterministic dynamics, we consider the deterministic dynamics of probability measures over phase space. This higher level of description that goes back to Koopman et al. [24,25] has recently been revitalized for dynamical systems theory [12].

The starting point for this approach is the conservation of probability as expressed by the Frobenius-Perron equation [33]

$$\rho(\mathbf{x}, t) = \int_X \delta(\mathbf{x} - \Phi^{t-t'}(\mathbf{x}'))\rho(\mathbf{x}', t')\mathrm{d}\mathbf{x}' , \tag{11.32}$$

where $\rho(\mathbf{x}, t)$ denotes a probability density function over the phase space X at time t of a dynamical system, $\Phi^t : X \to X$ refers to either a continuous-time ($t \in \mathbb{R}_0^+$) or discrete-time ($t \in \mathbb{N}_0$) flow and the integral over the delta function expresses the probability summation of alternative trajectories all leading into the same state $\mathbf{x}$ at time t.

In the case of an NDA, the flow is discrete and piecewise affine-linear on the domains D_ν as given by Eq. (11.26). As initial probability distribution densities $\rho(\mathbf{x}, 0)$ we consider uniform distributions with rectangular support $R_0 \subset X$, corresponding to an initial NDA macrostate,

$$u(\mathbf{x}, 0) = \frac{1}{|R_0|}\chi_{R_0}(\mathbf{x}) , \tag{11.33}$$

where

$$\chi_A(\mathbf{x}) = \begin{cases} 0 & : \quad \mathbf{x} \notin A \\ 1 & : \quad \mathbf{x} \in A \end{cases} \tag{11.34}$$

is the characteristic function for a set $A \subset X$. A crucial requirement for these distributions is that they must be consistent with the partition $\mathscr{P}$ of the NDA, i.e. there must be a bi-index $\nu = (i, j)$ such that the support $R_0 \subset D_\nu$.

Inserting (11.33) into the Frobenius-Perron equation (11.32) yields for one iteration

$$u(\mathbf{x}, t + 1) = \int_X \delta(\mathbf{x} - \Phi(\mathbf{x}'))u(\mathbf{x}', t)\mathrm{d}\mathbf{x}' . \tag{11.35}$$

In order to evaluate (11.35), we first use the product decomposition of the involved functions:

$$u(\mathbf{x}, 0) = u_x(x, 0)u_y(y, 0) \tag{11.36}$$

with

$$u_x(x, 0) = \frac{1}{|I_0|}\chi_{I_0}(x) \tag{11.37}$$

$$u_y(y,0) = \frac{1}{|J_0|}\chi_{J_0}(y) \tag{11.38}$$

and

$$\delta(\mathbf{x} - \Phi(\mathbf{x}')) = \delta(x - \Phi_x(\mathbf{x}'))\delta(y - \Phi_y(\mathbf{x}')) , \tag{11.39}$$

where the intervals I_0, J_0 are the projections of R_0 onto x- and y-axes, respectively. Correspondingly, Φ_x and Φ_y are the projections of Φ onto x- and y-axes, respectively. These are obtained from (11.26) as

$$\Phi_x(\mathbf{x}') = a_x^\nu + \lambda_x^\nu x' \tag{11.40}$$

$$\Phi_y(\mathbf{x}') = a_y^\nu + \lambda_y^\nu y' . \tag{11.41}$$

Using this factorization, the Frobenius-Perron equation (11.35) separates into

$$u_x(x,t+1) = \int_{[0,1]} \delta(x - a_x^\nu - \lambda_x^\nu x')u_x(x',t)\mathrm{d}x' \tag{11.42}$$

$$u_y(y,t+1) = \int_{[0,1]} \delta(y - a_y^\nu - \lambda_y^\nu y')u_y(y',t)\mathrm{d}y' \tag{11.43}$$

Next, we evaluate the delta functions according to the well-known lemma

$$\delta(f(x)) = \sum_{l:\text{simple zeros}} |f'(x_l)|^{-1}\delta(x - x_l) , \tag{11.44}$$

where $f'(x_l)$ indicates the first derivative of f in x_l. Equation (11.44) yields for the x-axis

$$x_\nu = \frac{x - a_x^\nu}{\lambda_x^\nu} , \tag{11.45}$$

i.e. one zero for each ν-branch, and hence

$$|f'(x'_\nu)| = \lambda_x^\nu . \tag{11.46}$$

Inserting (11.44), (11.45) and (11.46) into (11.42), gives

$$\begin{aligned} u_x(x,t+1) &= \sum_\nu \int_{[0,1]} \frac{1}{\lambda_x^\nu}\delta\left(x' - \frac{x - a_x^\nu}{\lambda_x^\nu}\right)u_x(x',t)\mathrm{d}x' \\ &= \sum_\nu \frac{1}{\lambda_x^\nu}u_x\left(\frac{x - a_x^\nu}{\lambda_x^\nu}, t\right) \end{aligned}$$

Next, we take into account that the distributions must be consistent with the NDA's partition. Therefore, for given $\mathbf{x} \in D_\nu$ there is only one branch of Φ contributing a simple zero to the sum above. Hence,

$$u_x(x, t+1) = \frac{1}{\lambda_x^\nu} u_x \left(\frac{x - a_x^\nu}{\lambda_x^\nu}, t \right) . \tag{11.47}$$

Our main finding is now that the evolution of uniform p.d.f.s with rectangular support according to the NDA dynamics Eq. (11.35) is governed by

$$u(\mathbf{x}, t) = \frac{1}{|\Phi^t(R_0)|} \chi_{\Phi^t(R_0)}(\mathbf{x}) , \tag{11.48}$$

i.e. uniform distributions with rectangular support are mapped onto uniform distributions with rectangular support [8].

For the proof we first insert the initial uniform density distribution (11.33) for $t = 0$ into Eq. (11.47), to obtain by virtue of (11.37)

$$u_x(x, 1) = \frac{1}{\lambda_x^\nu} u_x \left(\frac{x - a_x^\nu}{\lambda_x^\nu}, 0 \right) = \frac{1}{\lambda_x^\nu} \frac{1}{|I_0|} \chi_{I_0} \left(\frac{x - a_x^\nu}{\lambda_x^\nu} \right) .$$

Deploying (11.34) yields

$$\chi_{I_0} \left(\frac{x - a_x^\nu}{\lambda_x^\nu} \right) = \begin{cases} 0 & : \frac{x - a_x^\nu}{\lambda_x^\nu} \notin I_0 \\ 1 & : \frac{x - a_x^\nu}{\lambda_x^\nu} \in I_0 . \end{cases}$$

Let now $I_0 = [p_0, q_0] \subset [0, 1]$ we get

$$\begin{aligned} & \frac{x - a_x^\nu}{\lambda_x^\nu} \in I_0 \\ \Longleftrightarrow\ & p_0 \le \frac{x - a_x^\nu}{\lambda_x^\nu} \le q_0 \\ \Longleftrightarrow\ & \lambda_x^\nu p_0 \le x - a_x^\nu \le \lambda_x^\nu q_0 \\ \Longleftrightarrow\ & a_x^\nu + \lambda_x^\nu p_0 \le x \le a_x^\nu + \lambda_x^\nu q_0 \\ \Longleftrightarrow\ & \Phi_x(p_0) \le x \le \Phi_x(q_0) \\ \Longleftrightarrow\ & x \in \Phi_x(I_0) , \end{aligned}$$

where we made use of (11.40). Moreover, we have

$$\lambda_x^\nu |I_0| = \lambda_x^\nu (q_0 - p_0) = q_1 - p_1 = |I_1|$$

with $I_1 = [p_1, q_1] = \Phi_x(I_0)$. Therefore,

$$u_x(x,1) = \frac{1}{|I_1|}\chi_{I_1}(x)\,.$$

The same argumentation applies to the y-axis, such that we eventually obtain

$$u(\mathbf{x},1) = \frac{1}{|R_1|}\chi_{R_1}(\mathbf{x})\,, \tag{11.49}$$

with $R_1 = \Phi(R_0)$ the image of the initial rectangle $R_0 \subset X$. Thus, the image of a uniform density function with rectangular support is a uniform density function with rectangular support again.

Next, assume (11.48) is valid for some $t \in \mathbb{N}$. Then it is obvious that (11.48) also holds for $t+1$ by inserting the x-projection of (11.48) into (11.47) using (11.37), again. Then, the same calculation as above applies when every occurrence of 0 is replaced by t and every occurrence of 1 is replaced by $t+1$. By means of this inductive proof we have implemented an NDA by a dynamically evolving field. Therefore, we call this representation dynamic field automaton (DFA).

The Frobenius-Perron equation (11.35) can be regarded as a time-discretized Amari dynamic neural field equation (11.3). Discretizing time according to Euler's rule with increment $\Delta t = \tau$ where τ is the time constant of the Amari equation (11.3) yields

$$\begin{aligned}\tau\frac{u(\mathbf{x},t+\tau)-u(\mathbf{x},t)}{\tau} + u(\mathbf{x},t) &= \int_D w(\mathbf{x},\mathbf{y})f(u(\mathbf{y},t))\,\mathrm{d}\mathbf{y}\\ u(\mathbf{x},t+\tau) &= \int_D w(\mathbf{x},\mathbf{y})f(u(\mathbf{y},t))\,\mathrm{d}\mathbf{y}\,.\end{aligned}$$

For $\tau = 1$ and $f(u) = u$ the Amari equation becomes the Frobenius-Perron equation (11.35) when we set

$$w(\mathbf{x},\mathbf{y}) = \delta(\mathbf{x} - \Phi(\mathbf{y})) \tag{11.50}$$

where Φ is the NDA mapping from Eq. (11.27). This is the general solution of the kernel construction problem [6, 38]. Note that Φ is not injective, i.e. for fixed $\mathbf{x}$ the kernel is a sum of delta functions encoding the influence from different parts of the space $X = [0,1]^2$.

Finally we carry out the whole construction for our language example. This yields the field dynamics depicted in Fig. 11.4.

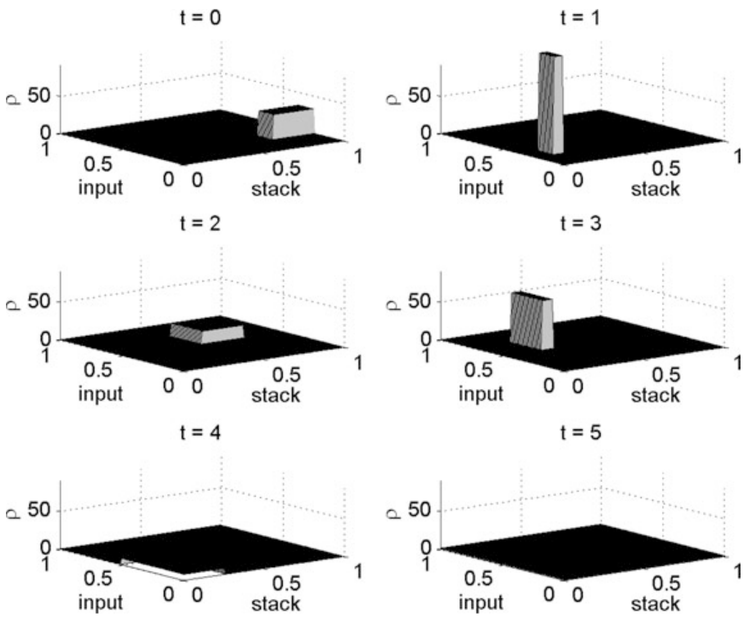

Fig. 11.4 Dynamic field automaton for processing the string "the dog chased the cat" (`NP V NP`) according to Table 11.1. The NDA states become rectangular supports of uniform distributions which are mapped onto uniform distributions with rectangular supports during discrete temporal evolution

11.4 Discussion

Turing machines and Gödel numbers are important pillars of the theory of computation [20,47]. Thus, any computational architecture needs to show how it could relate to Turing machines and in what way stable implementations of Turing computation is possible. In this chapter, we addressed the question how universal Turing computation could be implemented in a neural field environment as described by the Amari field equation (11.1). To this end, we employed the canonical symbologram representation [13,23] of the machine tape as the unit square, resulting from a Gödel encoding of sequences of states.

The action of the Turing machine on a state description is given by a state flow on the unit square which led to a Frobenius-Perron equation (11.32) for the evolution of uniform probability densities. We have implemented this equation in the neural field space by a piecewise affine-linear kernel geometry on the unit square which can be expressed naturally within a neural field framework. We also showed that stability of states and dynamics both in time as well as its encoding for finite programs is achieved by the approach.

However, our construction essentially relied upon discretized time that could be provided by some clock mechanism. The crucial problem of stabilizing states within every clock cycle could be principally solved by established methods from dynamic field architectures. In such a time-continuous extension, an excited state, represented by a rectangle in one layer, will only excite a subsequent state, represented by another rectangle in another layer when a condition-of-satisfaction is met [40, 41]. Otherwise rectangular states would remain stabilized as described by Eq. (11.10). All these problems provide promising prospects for future research.

Acknowledgements We thank Slawomir Nasuto and Serafim Rodrigues for helpful comments improving this chapter. This research was supported by a DFG Heisenberg fellowship awarded to PbG (GR 3711/1-2).

References

1. Afraimovich, V.S., Zhigulin, V.P., Rabinovich, M.I.: On the origin of reproducible sequential activity in neural circuits. Chaos **14**(4), 1123–1129 (2004)
2. Amari, S.I.: Dynamics of pattern formation in lateral-inhibition type neural fields. Biol. Cybern. **27**, 77–87 (1977)
3. Arbib, M.A. (ed.): The Handbook of Brain Theory and Neural Networks, 1st edn. MIT, Cambridge (1995)
4. beim Graben, P., Gerth, S.: Geometric representations for minimalist grammars. J. Log. Lang. Inf. **21**(4), 393–432 (2012)
5. beim Graben, P., Hutt, A.: Attractor and saddle node dynamics in heterogeneous neural fields. EPJ Nonlinear Biomed. Phys. **2**, 4 (2014). doi:10.1140/epjnbp17
6. beim Graben, P., Potthast, R.: Inverse problems in dynamic cognitive modeling. Chaos **19**(1), 015103 (2009)
7. beim Graben, P., Potthast, R.: A dynamic field account to language-related brain potentials. In: Rabinovich, M., Friston, K., Varona, P. (eds.) Principles of Brain Dynamics: Global State Interactions, chap. 5, pp. 93–112. MIT, Cambridge (2012)
8. beim Graben, P., Potthast, R.: Implementing Turing machines in dynamic field architectures. In: Bishop, M., Erden, Y.J. (eds.) Proceedings of AISB12 World Congress 2012 – Alan Turing 2012, vol. 5th AISB Symposium on Computing and Philosophy: Computing, Philosophy and the Question of Bio-Machine Hybrids, Birmingham, pp. 36–40 (2012). http://arxiv.org/abs/1204.5462
9. beim Graben, P., Jurish, B., Saddy, D., Frisch, S.: Language processing by dynamical systems. Int. J. Bifurc. Chaos **14**(2), 599–621 (2004)
10. beim Graben, P., Gerth, S., Vasishth, S.: Towards dynamical system models of language-related brain potentials. Cogn. Neurodynamics **2**(3), 229–255 (2008)
11. beim Graben, P., Pinotsis, D., Saddy, D., Potthast, R.: Language processing with dynamic fields. Cogn. Neurodynamics **2**(2), 79–88 (2008)
12. Budišić, M., Mohr, R., Mezić, I.: Applied Koopmanism. Chaos **22**(4), 047510 (2012)
13. Cvitanović, P., Gunaratne, G.H., Procaccia, I.: Topological and metric properties of Hénon-type strange attractors. Phys. Rev. A **38**(3), 1503–1520 (1988)
14. Erlhagen, W., Schöner, G.: Dynamic field theory of movement preparation. Psychol. Rev. **109**(3), 545–572 (2002)
15. Fukai, T., Tanaka, S.: A simple neural network exhibiting selective activation of neuronal ensembles: from winner-take-all to winners-share-all. Neural Comput. **9**(1), 77–97 (1997)
16. Gayler, R.W.: Vector symbolic architectures are a viable alternative for Jackendoff's challenges. Behav. Brain Sci. **29**, 78–79 (2006)

17. Gödel, K.: Über formal unentscheidbare Sätze der *principia mathematica* und verwandter Systeme I. Monatshefte für Mathematik und Physik **38**, 173–198 (1931)
18. Hertz, J.: Computing with attractors. In: Arbib, M.A. (ed.): The Handbook of Brain Theory and Neural Networks, 1st edn, pp. 230–234. MIT, Cambridge (1995)
19. Hofstadter, D.R.: Gödel, Escher, Bach: An Eternal Golden Braid. Basic Books, New York (1979)
20. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison–Wesley, Menlo Park (1979)
21. Hopfield, J.J.: Neurons with graded response have collective computational properties like those of two-state neurons. Proc. Natl. Acad. Sci. USA **81**(10), 3088–3092 (1984)
22. Jirsa, V.K., Kelso, J.A.S.: Spatiotemporal pattern formation in neural systems with heterogeneous connection toplogies. Phys. Rev. E **62**(6), 8462–8465 (2000)
23. Kennel, M.B., Buhl, M.: Estimating good discrete partitions from observed data: symbolic false nearest neighbors. Phys. Rev. Lett. **91**(8), 084,102 (2003)
24. Koopman, B.O.: Hamiltonian systems and transformations in Hilbert space. Proc. Natl. Acad. Sci. USA **17**, 315–318 (1931)
25. Koopman, B.O., von Neumann, J.: Dynamical systems of continuous spectra. Proc. Natl. Acad. Sci. USA **18**, 255–262 (1932)
26. Lind, D., Marcus, B.: An Introduction to Symbolic Dynamics and Coding. Cambridge University Press, Cambridge (1995)
27. McCulloch, W.S., Pitts, W.: A logical calculus of ideas immanent in nervous activity. Bull. Math. Biophys. **5**, 115–133 (1943)
28. McMillan, B.: The basic theorems of information theory. Ann. Math. Stat. **24**, 196–219 (1953)
29. Moore, C.: Unpredictability and undecidability in dynamical systems. Phys. Rev. Lett. **64**(20), 2354–2357 (1990)
30. Moore, C.: Generalized shifts: unpredictability and undecidability in dynamical systems. Nonlinearity **4**, 199–230 (1991)
31. Moore, C.: Dynamical recognizers: real-time language recognition by analog computers. Theor. Comput. Sci. **201**, 99–136 (1998)
32. Moore, C., Crutchfield, J.P.: Quantum automata and quantum grammars. Theor. Comput. Sci. **237**, 275–306 (2000)
33. Ott, E.: Chaos in Dynamical Systems. Cambridge University Press, New York (1993)
34. Pitts, W., McCulloch, W.S.: How we know universals: the perception of auditory and visual forms. Bull. Math. Biophys. **9**, 127–147 (1947)
35. Plate, T.A.: Holographic reduced representations. IEEE Trans. Neural Netw. **6**(3), 623–641 (1995)
36. Pollack, J.B.: The induction of dynamical recognizers. Mach. Learn. **7**, 227–252 (1991). Also published in [37], pp. 283–312
37. Port, R.F., van Gelder, T. (eds.): Mind as Motion: Explorations in the Dynamics of Cognition. MIT, Cambridge (1995)
38. Potthast, R., beim Graben, P.: Inverse problems in neural field theory. SIAM J. Appl. Dyn. Syst. **8**(4), 1405–1433 (2009)
39. Rabinovich, M.I., Huerta, R., Varona, P., Afraimovich, V.S.: Transient cognitive dynamics, metastability, and decision making. PLoS Comput. Biol. **4**(5), e1000,072 (2008)
40. Sandamirskaya, Y., Schöner, G.: Dynamic field theory of sequential action: a model and its implementation on an embodied agent. In: Proceedings of the 7th IEEE International Conference on Development and Learning (ICDL), Monterey, pp. 133–138 (2008)
41. Sandamirskaya, Y., Schöner, G.: An embodied account of serial order: how instabilities drive sequence generation. Neural Netw. **23**(10), 1164–1179 (2010)
42. Schöner, G.: Neural systems and behavior: dynamical systems approaches. In: Smelser, N.J., Baltes, P.B. (eds.) International Encyclopedia of the Social & Behavioral Sciences, pp. 10571–10575. Pergamon, Oxford (2002)
43. Siegelmann, H.T., Sontag, E.D.: On the computational power of neural nets. J. Comput. Syst. Sci. **50**(1), 132–150 (1995)

44. Smolensky, P.: Tensor product variable binding and the representation of symbolic structures in connectionist systems. Artif. Intell. **46**(1–2), 159–216 (1990)
45. Smolensky, P.: Harmony in linguistic cognition. Cogn. Sci. **30**, 779–801 (2006)
46. Sontag, E.D.: Automata and neural networks. In: Arbib, M.A. (ed.): The Handbook of Brain Theory and Neural Networks, 1st edn, pp. 119–123. MIT, Cambridge (1995)
47. Spencer, M.C., Tanay, T., Roesch, E.B., Bishop, J.M., Nasuto, S.J.: Abstract platforms of computation. AISB 2013, Exeter (2013)
48. Tabor, W.: Fractal encoding of context-free grammars in connectionist networks. Expert Syst.: Int. J. Knowl. Eng. Neural Netw. **17**(1), 41–56 (2000)
49. Tabor, W., Cho, P.W., Szkudlarek, E.: Fractal analyis illuminates the form of connectionist structural gradualness. Top. Cogn. Sci. **5**, 634–667 (2013)
50. Tennent, R.D.: The denotational semantics of programming languages. Commun. ACM **19**(8), 437–453 (1976)
51. Turing, A.M.: On computable numbers, with an application to the *Entscheidungsproblem*. Proc. Lond. Math. Soc. **2**(42), 230–265 (1937)