Inverse problems in dynamic cognitive modeling

Peter beim Graben[1,a)] and Roland Potthast[2]
[1]*School of Psychology and Clinical Language Sciences, University of Reading, Reading, Berkshire RG6 6AH, United Kingdom*
[2]*Department of Mathematics, University of Reading, Reading, Berkshire RG6 6AH, United Kingdom*

(Received 28 November 2008; accepted 11 February 2009; published online 31 March 2009)

Inverse problems for dynamical system models of cognitive processes comprise the determination of synaptic weight matrices or kernel functions for neural networks or neural/dynamic field models, respectively. We introduce dynamic cognitive modeling as a three tier top-down approach where cognitive processes are first described as algorithms that operate on complex symbolic data structures. Second, symbolic expressions and operations are represented by states and transformations in abstract vector spaces. Third, prescribed trajectories through representation space are implemented in neurodynamical systems. We discuss the Amari equation for a neural/dynamic field theory as a special case and show that the kernel construction problem is particularly ill-posed. We suggest a Tikhonov–Hebbian learning method as regularization technique and demonstrate its validity and robustness for basic examples of cognitive computations. © *2009 American Institute of Physics*. [DOI: 10.1063/1.3097067]

Inverse problems, the determination of system parameters from observable or theoretically prescribed dynamics, are prevalent in the cognitive neurosciences. In particular, the dynamical system approach to cognition involves learning procedures for neural networks or neural/dynamic fields. We present dynamic cognitive modeling as a three tier top-down approach comprising the levels of (1) cognitive processes, (2) their state space representation, and (3) dynamical system implementations that are guided by neuroscientific principles. These levels are passed through in a top-down fashion: (1) cognitive processes are described as algorithms sequentially operating on complex symbolic data structures that are decomposed using so-called filler/role bindings; (2) data structures are mapped onto points in abstract vector spaces using tensor product representations; and (3) cognitive operations are implemented as dynamics of neural networks or neural/dynamic fields. The last step involves the solution of inverse problems, namely, training the system's parameters to reproducing prescribed trajectories of cognitive operations in representation space. We show that learning tasks for neural/dynamic field models are particularly ill-posed and propose a regularization technique for the common Hebb rule, resulting into modified Tikhonov–Hebbian learning. The methods are illustrated by means of three instructive examples for basic cognitive processing, where we show that Tikhonov–Hebbian learning is a quick and simple training algorithm, not requiring orthogonality or even linear independence of training patterns. In fact, the regularization is robust against linearly dependent patterns as they could result from oversampling.

I. INTRODUCTION

Investigating nonlinear dynamical systems is an important task in the sciences. In the ideal case, one has a theoretical model in form of differential (or integrodifferential) equations and computes their analytical solutions. However, this approach is often not tractable for complex nonlinear systems. Here, analytical techniques, such as stability, bifurcation, or synchronization analysis, provide insights into the structural properties of the flow in phase space. If these methods are applicable only up to some extent, numerical solution of the model equations, i.e., determining the system's trajectories through phase space for given initial and boundary conditions, is of great importance. This forward problem for nonlinear dynamical systems is nowadays well understood and appreciated in science.

By contrast, the peculiarities and possible pitfalls of the inverse problem of determining the system's equations (or the system's parameters for a given class of equations) from observed or prescribed trajectories are less acknowledged in nonlinear dynamical system research today. Inverse problems are typically ill-posed. This notion does not only refer to the fact that there is, in general, no unique parametrization for a prescribed solution, but moreover, that such parametrizations are highly unstable and extremely sensitive to training data.

Related to dynamical system models of cognitive processes,[1–4] inverse problems are prevalent in several training algorithms for artificial neural networks (ANNs).[5,6] It is the aim of the present study to investigate these problems, in principle, by means of continuum approximations for neural networks, which are known as neural or dynamic field models. We show that learning tasks for such fields are particularly ill-posed and suggest a regularization technique for the common Hebb rule, resulting into modified Tikhonov–Hebbian learning.

[a)]Electronic mail: p.r.beimgraben@reading.ac.uk.

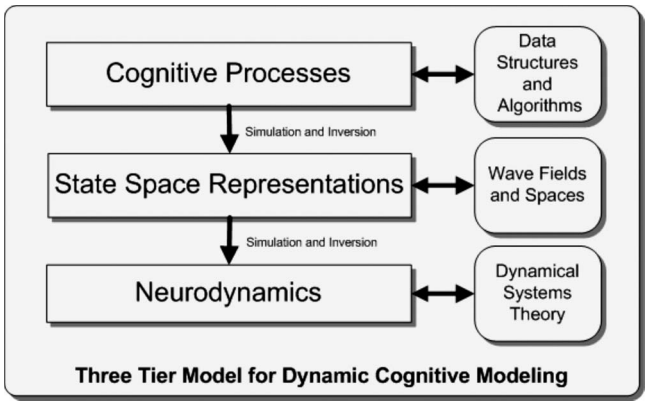

FIG. 1. Three tier top-down approach to inverse dynamic cognitive modeling.

This paper is structured as follows. Section II reviews cognitive modeling and inverse problems related to this field. Inspired by seminal work of Marr and Poggio,[7] we introduce dynamic cognitive modeling as a three tier top-down approach, as illustrated in Fig. 1. (1) At the highest level of mental states and processes, the relevant data structures and algorithms from the prospect of the computer metaphor of the mind[8] are determined in order to obtain symbolic patterns and transformation rules that are described by the so-called filler/role bindings.[9–14] (2) These symbolic structures and processes are mapped onto points and continuous trajectories[15] in abstract vector spaces, respectively, by tensor product representations.[9–14] Interestingly, the word "representation" does not only refer to models of mental representations here. It assumes a precise meaning in terms of mathematical representation theory:[16,17] cognitive operations are represented as operators in neural representation spaces. (3) Cognitive representations are implemented by nonlinear dynamical systems obeying guiding principles from the neurosciences. Here, we discuss the inverse problem for a large class of continuous neural networks (so-called neural or dynamic field models)[18–34] described by integrodifferential equations.

In Secs. III and IV we demonstrate how inverse problems for dynamical system models of cognition can be regularized by a Tikhonov–Hebbian learning rule for neural field models. Three examples are constructed in Sec. III and presented in Sec. IV. Section V gives a concluding discussion.

II. DYNAMIC COGNITIVE MODELING

This section introduces dynamic cognitive modeling as a three tier hierarchy, comprising (1) cognitive processes and symbolic structures, (2) their state space representations, and (3) their implementation by neurodynamics,[7] which is pursued from the top level (1) down to the bottom level (3).

A. Cognitive processes

According to the computer metaphor of the mind,[8] cognition is essentially symbol manipulation obeying combinatorial rules.[35,36] A paradigmatic concept for classical cognitive science is the Turing machine as a formal computer.[37]

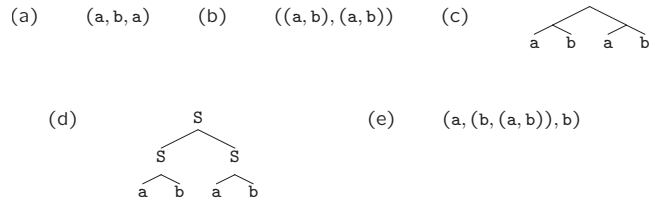

FIG. 2. Complex symbolic data structures for mental representations. (a) simple list, (b) list of simple lists, (c) the corresponding tree for (b), (d) the same tree with node labels S, (e) an even more complex frame of nested lists.

However, the Turing machine is of only marginal interest in cognitive psychology[38] and psycholinguistics[39] as it has unlimited computational resources in the form of a randomly accessible bi-infinite memory tape. Therefore, less powerful devices such as pushdown automata or nested stack automata[37,40] are of greater importance for computational models of cognitive processes. A pushdown automaton has a one-sided infinite memory tape that is accessible in "last in/first out" manner, i.e., the device has only access to the topmost symbol stored at the stack.

Turing machines, pushdown automata, or nested stack automata define particular formal languages, namely, the recursively enumerable languages, context-free languages, and indexed languages, respectively.[37] Especially the latter both are relevant in the field of computational psycholinguistics because sentences can be described by phrase structure trees and some restricted operations acting on them.[40–44]

1. Data structures and algorithms

The first step in devising a computational dynamical system model is specification of the relevant data structures and algorithms where the former instantiates a model for mental representations, while the latter defines the mental or cognitive computations. In computer science, complex symbolic data structures are e.g., lists, trees, nested lists, often called "frames," or even lists of trees, etc. Figure 2 depicts some examples for data structures.

The first example in Fig. 2(a) is a list of three items, the symbols a and b (note that symbolic expressions are printed in Roman font subsequently), where a takes the first and third position, while b occupies the second position of the list. Assuming a pushdown automaton that has only access to the first symbol a in the list, we have a simple description of a stack tape. Such an automaton can achieve symbolic operations, e.g., "push," where $\alpha' = \mathrm{push}(\mathrm{a}, \alpha)$ places a new symbol a at the top of the stack, denoted $\alpha = (a_1, a_2, \ldots, a_n)$, such that $\alpha' = (\mathrm{a}, a_1, a_2, \ldots, a_n)$. Starting with an empty list, we could, e.g., apply push three times resulting into the state transitions

$$\alpha_0 = (\,), \quad \alpha_1 = (\mathrm{a}) = \mathrm{push}(\mathrm{a}, \alpha_0),$$
$$\alpha_2 = (\mathrm{a}, \mathrm{a}) = \mathrm{push}(\mathrm{a}, \alpha_1), \quad \alpha_3 = (\mathrm{a}, \mathrm{a}, \mathrm{a}) = \mathrm{push}(\mathrm{a}, \alpha_2). \tag{1}$$

Another important interpretation of lists such as in Fig. 2(a) is related to logical inferences. Replacing, e.g., the symbol a by 0 and b by 1, yields the list (0,1,0) that can be regarded as one row of a logical truth table, where the first two items are

TABLE I. Truth table of the logical equivalence relation A equiv $B=C$.

A	B	C
0	0	1
0	1	0
1	0	0
1	1	1

the inputs and the third is the output of a logical function. In our case, the list (0, 1, 0) is the second row of the truth table of the logical equivalence relation equiv, defined in Table I.

Figure 2(b) shows a complex lists containing two simple lists. This structure can be regarded as a tree shown in Fig. 2(c). Figure 2(d) introduces the concept of a labeled tree where to each node a symbolic label is assigned. From a labeled tree one can derive a context-free grammar (CFG) of production rules by interpreting each tree branching as a rule of the form $X \rightarrow YZ$, where X denotes the mother node, and Y, and Z its immediate daughters. Therefore, example in Fig. 2(d) gives rise to the CFG

$$\mathbf{T}=\{\mathrm{a},\mathrm{b}\}, \quad \mathbf{N}=\{\mathrm{S}\}, \quad P=\left\{\begin{array}{ll}(1) & \mathrm{S}\rightarrow \mathrm{S}\ \ \mathrm{S}\\ (2) & \mathrm{S}\rightarrow \mathrm{a}\ \ \mathrm{b}\end{array}\right\}, \tag{2}$$

where $\mathbf{T}=\{\mathrm{a},\mathrm{b}\}$ is called the set of terminal symbols, $\mathbf{N}=\{\mathrm{S}\}$ that of nonterminal symbols, with the distinguished start symbol S, and the production rules expand one nonterminal at the left-hand side into a string of nonterminals or terminals at the right-hand side. Applying the rules from a CFG recursively describes a tree generation dynamics as, e.g., depicted in Fig. 10 in Sec. III D. Finally, Fig. 2(e) presents an even more complex expression of nested lists that might be regarded as a model for cognitive frames.

Coming back to the general paradigm of a Turing machine, such an automaton is formally defined as a 7-tuple

$$M_{TM}=(Q,\mathbf{N},\mathbf{T},\delta,q_0,b,F), \tag{3}$$

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, $\mathbf{T}\subset\mathbf{N}\backslash\{b\}$ is the set of admitted input symbols,

$$\delta: Q\times\mathbf{N}\rightarrow Q\times\mathbf{N}\times\{L,R\} \tag{4}$$

is a partial state transition function (the "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 being read from the memory tape. 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 3 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" assumed by the machine when a computation terminates.[37]

In order to describe the machine's behavior as deterministic dynamics in symbolic "phase space," one introduces the notion of a state description, which is a triple

$$s=(\alpha,q,\beta) \tag{5}$$

with $\alpha,\beta\in\mathbf{N}^*$ (i.e., they are lists of tape symbols from $\mathbf{N}$ of arbitrary, yet finite, length, delimited by blank symbols b).

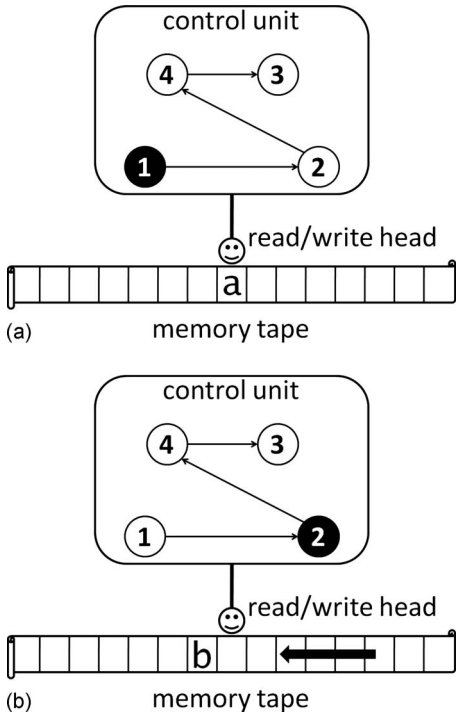

FIG. 3. Example state transition from (a) to (b) of a Turing machine with $\delta(1,\mathrm{a})=(2,\mathrm{b},L)$.

Then, the transition function can be extended to state descriptions by

$$\delta^*: S\rightarrow S, \quad s'=\delta^*(s), \tag{6}$$

where $S=\mathbf{N}^*\times Q\times\mathbf{N}^*$ now plays the role of a phase space of a discrete dynamical system. We discuss consequences of this view in the next section.

2. Filler/role bindings

However, first we introduce a general framework for formalizing arbitrary data structures and symbolic operations suggested by Smolensky and co-workers[9–12] and recently deployed by beim Graben *et al.*[13,14] This filler/role binding decomposition identifies the particular symbols occurring in complex expressions with the so-called fillers $f\in F$, where F is some finite set of cardinality N_F. Applied to the Turing machine, we can therefore choose $F=\mathbf{N}$, the set of tape symbols, including blank and input symbols. Fillers are bound to symbolic roles $r\in R$, where R is another finite (or countable) set of possible roles. Examples for such roles are (1) slots for list positions $R=\{r_i|1\leq i\leq n\}$ indicating the ith position in a list of length n; (2) slots for tree positions $R=\{r_1,r_2,r_3\}$, where

$$r_1=\text{mother}, \quad r_2=\text{left daughter}, \quad r_3=\text{right daughter}, \tag{7}$$

as indicated in Fig. 4; and (3) slots in arbitrary cognitive frames as in Fig. 2(e).

Considering the example from Fig. 2(a) yields fillers $F=\{\mathrm{a},\mathrm{b}\}$ and roles $R=\{r_1,r_2,r_3\}$ for the three list positions. A filler/role binding is now a set of pairs (f_i,r_j) when filler f_i occurs at (is "bound" to) role r_j. Thus, the filler/role decomposition for example Fig. 2(a) is given as

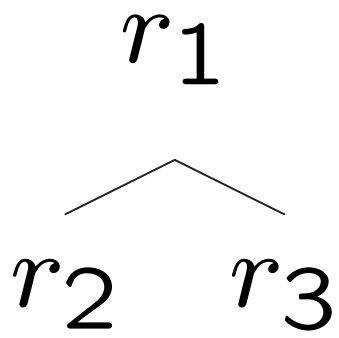

FIG. 4. Elementary role positions of a labeled binary tree.

$$f_\alpha = \{(\mathrm{a}, r_1), (\mathrm{b}, r_2), (\mathrm{a}, r_3)\}. \tag{8}$$

Such a relation can be regarded as a complex filler that could be recursively bound to roles again. Therefore, we obtain the filler/role decomposition for example Fig. 2(b) as a set of pairs of sets of pairs

$$f_T = \{(\{(\mathrm{a}, r_1), (\mathrm{b}, r_2)\}, r_1), (\{(\mathrm{a}, r_1), (\mathrm{b}, r_2)\}, r_2)\}, \tag{9}$$

where the complex filler, namely, the list $\{(\mathrm{a}, r_1), (\mathrm{b}, r_2)\}$ is bound to both roles r_1 and r_2 recursively. This is also the correct decomposition of the tree from Fig. 2(c).

For describing the tree from Fig. 2(d), the node labels have to be taken into account. As fillers we choose the labels $F=\{\mathrm{S}, \mathrm{a}, \mathrm{b}\}$, whereas the roles $R=\{r_1, r_2, r_3\}$ are given through Eq. (7). In a bottom-up manner, we first decompose the leftmost subtree L by assigning filler S to the root node r_1, filler a to the left daughter node r_2 and filler b to the right daughter node r_3, obtaining the complex filler

$$f_L = \{(\mathrm{S}, r_1), (\mathrm{a}, r_2), (\mathrm{b}, r_3)\}. \tag{10}$$

This is also the correct decomposition of the right subtree f_R, such that $f_R = f_L$. Then, f_L is bound to the left daughter r_2 of the next tree level, f_R is bound to its right daughter r_3 and its root node r_1 is occupied by the simple filler S again, such that

$$f_T = \{(\mathrm{S}, r_1), (f_L, r_2), (f_R, r_3)\}. \tag{11}$$

The complete filler/role binding of the tree Fig. 2(d) is hence

$$f_T = \{(\mathrm{S}, r_1), (\{(\mathrm{S}, r_1), (\mathrm{a}, r_2), (\mathrm{b}, r_3)\}, r_2), (\{(\mathrm{S}, r_1), (\mathrm{a}, r_2), (\mathrm{b}, r_3)\}, r_3)\}. \tag{12}$$

Application of filler/role binding to state descriptions of Turing machines is possible in several ways. Most straightforwardly, one assigns three role positions $R_S=\{r_1, r_2, r_3\}$ to the three slots in the state description [Eq. (5)]. Then, the strings $\alpha, \beta \in \mathbf{N}^*$ are regarded as complex fillers, namely, lists of tape symbols $F=\mathbf{N}$ with a countable number of slots $R_L = \{s_i | i \in \mathbb{N}\}$. Additionally, the machine states are represented by another set of fillers $F_Q = Q$ that only bind to role r_2. Thereby, one possible Turing machine filler/role decomposition is

$$f_s = \{(\{(a_1, s_1), (a_2, s_2), \ldots, (a_n, s_n)\}, r_1), (q, r_2), (\{(b_1, s_1), (b_2, s_2), \ldots, (b_m, s_m)\}, r_3)\}, \tag{13}$$

with $a_i, b_j \in F$ and $n, m \in \mathbb{N}$.

However, another decomposition is more obvious. Here, the state description is regarded as a concatenation product

$$\gamma = \alpha' \cdot q \cdot \beta \tag{14}$$

of the strings α, β with the state q in the given order, where α' is the string α in reverted order, $\alpha' = a_n a_{n-1} \cdots a_1$.[13] Introducing the notion of "dotted sequences,"[45] yields a two-sided list of tape and control state symbols from $F=\mathbf{N} \cup Q$, where the dot "." indicates the position of the control state $q \in Q$ at the tape,

$$\gamma = a_n a_{n-1} \cdots a_1 q \,.\, b_1 b_2 \cdots b_m. \tag{15}$$

Moore[46,47] proved that this description of a Turing machine leads to generalized shifts investigated in symbolic dynamics.[45–48] Then, the filler/role decomposition of a Turing machine is that of a simple list

$$f_\gamma = \{(a_n, s_{n-1}), (a_{n-1}, s_{n-2}), \ldots, (a_1, s_{-1}), (q, s_0), (b_1, s_1), (b_2, s_2), \ldots, (b_m, s_m)\}, \tag{16}$$

where we have introduced integer list positions as roles $R = \{s_i | i \in \mathbb{Z}\}$. A more axiomatic framework for the filler/role binding was presented by beim Graben *et al.*[14]

B. State space representations

The filler/role decomposition of complex symbolic data structures is a first step toward their state space representation. This is achieved by the tensor product representation independently invented by Smolensky and co-workers[9–12] and Mizraji.[49,50] The tensor product calculus is a universal framework to describe different state space representations for dynamic cognitive modeling. We first review its general algebraic framework and discuss particular representations in the subsequent subsections.

In order to employ the tensor product representation, the respective fillers and roles, $f_i \in F$ and $r_j \in R$, are mapped onto vectors from two vector spaces $\mathcal{V}_F$ and $\mathcal{V}_R$ by a function

$$\rho: F_\infty \to \mathcal{V}_F \cup \mathcal{V}_R, \quad \rho(f^*) = \boldsymbol{f}^*, \tag{17}$$

where F_∞ contains the roles and the simple fillers $(F \cup R \subset F_\infty)$ but also all complex fillers f^* (for the exact definition of F_∞, see Ref. 14) such that $\boldsymbol{f}_i = \rho(f_i) \in \mathcal{V}_F$ represents a filler and $\boldsymbol{r}_j = \rho(r_j) \in \mathcal{V}_R$ represents a role.

A filler/role binding is then represented by the direct sum of tensor products

$$\rho((f_i, r_j)) = \rho(f_i) \otimes \rho(r_j), \tag{18}$$

$$\rho(\{(f_{i_1}, r_{j_1}), (f_{i_2}, r_{j_2})\}) = \rho(f_{i_1}) \otimes \rho(r_{j_1}) \oplus \rho(f_{i_2}) \otimes \rho(r_{j_2}). \tag{19}$$

As a consequence, the image $\mathcal{F} = \rho(F_\infty)$ is isomorphic to the Fock space

$$\mathcal{F} = \bigoplus_{n=1}^{\infty} \mathcal{V}_F \otimes \bigotimes_{k=1}^{n} \mathcal{V}_R$$

of many particle quantum systems.[11,17]

Applying this mapping to the examples from Fig. 2 yields the following representations. The simple list [Eq. (8)] from Fig. 2(a) is represented by a vector

$$\rho(\alpha) = \boldsymbol{a} \otimes \boldsymbol{r}_1 \oplus \boldsymbol{b} \otimes \boldsymbol{r}_2 \oplus \boldsymbol{a} \otimes \boldsymbol{r}_3 \tag{20}$$

with filler vectors $\boldsymbol{a}=\rho(\mathrm{a})$ and $\boldsymbol{b}=\rho(\mathrm{b})$. [For the sake of clarity, we write $\rho(\alpha)$ instead of $\rho(f_\alpha)$ in the sequel which would be the precise notation as ρ is applied to the filler/role binding f_α and not to the symbolic expression α itself.] Correspondingly, the nested list from Fig. 2(b) and the tree from Fig. 2(c) [Eq. (9)] are represented by tensor products of higher rank

$$\begin{aligned}\rho(T) &= (\boldsymbol{a} \otimes \boldsymbol{r}_1 \oplus \boldsymbol{b} \otimes \boldsymbol{r}_2) \otimes \boldsymbol{r}_1 \oplus (\boldsymbol{a} \otimes \boldsymbol{r}_1 \oplus \boldsymbol{b} \otimes \boldsymbol{r}_2) \otimes \boldsymbol{r}_2 \\ &= \boldsymbol{a} \otimes \boldsymbol{r}_1 \otimes \boldsymbol{r}_1 \oplus \boldsymbol{b} \otimes \boldsymbol{r}_2 \otimes \boldsymbol{r}_1 \oplus \boldsymbol{a} \otimes \boldsymbol{r}_1 \otimes \boldsymbol{r}_2 \oplus \boldsymbol{b} \otimes \boldsymbol{r}_2 \\ &\quad \otimes \boldsymbol{r}_2.\end{aligned}$$

In the same way, we obtain the tensor product representation of the tree from Fig. 2(d), derived in Eq. (12) as

$$\begin{aligned}\rho(T) &= \boldsymbol{S} \otimes \boldsymbol{r}_1 \oplus (\boldsymbol{S} \otimes \boldsymbol{r}_1 \oplus \boldsymbol{a} \otimes \boldsymbol{r}_2 \oplus \boldsymbol{b} \otimes \boldsymbol{r}_3) \otimes \boldsymbol{r}_2 \\ &\quad \oplus (\boldsymbol{S} \otimes \boldsymbol{r}_1 \oplus \boldsymbol{a} \otimes \boldsymbol{r}_2 \oplus \boldsymbol{b} \otimes \boldsymbol{r}_3) \otimes \boldsymbol{r}_3.\end{aligned} \tag{21}$$

Also the tensor product representations for the different filler/role decompositions of Turing machine state descriptions are constructed similarly.

1. Arithmetic representations

Dolan and Smolensky,[9] Smolensky,[10] Smolensky and Legendre,[11] Smolensky,[12] Mizraji,[49,50] and more recently, beim Graben *et al.*[13] used finite-dimensional arithmetic vector spaces $\mathcal{V}_F=\mathbb{R}^p$, $\mathcal{V}_R=\mathbb{R}^q$ $(p,q \in \mathbb{N})$, and their Kronecker tensor products to obtain arithmetic vector space representations that can be regarded as activation states of neural networks or connectionist architectures. Setting, e.g.,

$$\boldsymbol{a} = \begin{pmatrix}1\\0\end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix}0\\1\end{pmatrix}, \quad \boldsymbol{r}_1 = \begin{pmatrix}1\\0\\0\end{pmatrix}, \quad \boldsymbol{r}_2 = \begin{pmatrix}0\\1\\0\end{pmatrix}, \quad \boldsymbol{r}_3 = \begin{pmatrix}0\\0\\1\end{pmatrix}$$

yields the tensor product representation of the simple list [Eq. (20)] from example in Fig. 2(a)

$$\begin{aligned}\rho(\alpha) &= \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\\0\end{pmatrix} \oplus \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\\0\end{pmatrix} \oplus \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\0\\1\end{pmatrix} \\ &= \begin{pmatrix}1\\0\\0\\0\\0\\0\end{pmatrix} \oplus \begin{pmatrix}0\\0\\0\\0\\1\\0\end{pmatrix} \oplus \begin{pmatrix}1\\0\\0\\0\\0\\0\end{pmatrix} = \begin{pmatrix}2\\0\\0\\0\\1\\0\end{pmatrix}.\end{aligned}$$

Obviously, the dimensionality of this representation is exponentially increasing with increasing recursion, thus leading to an almost meaningless vacuum, where symbolic states are sparsely scattered around (see, however, Refs. 11 and 12 for possible solutions).

2. Fractal representations

In order to avoid sparse high-dimensional state space representations, combinations of vectorial with purely numerical encodings, the so-called fractal encodings were used by Siegelmann and Sontag[51] and Tabor.[52,53]

Consider again the list example [Eq. (20)] from Fig. 2(a) with two fillers $F=\{\mathrm{a},\mathrm{b}\}$ and a countable number of list position roles $R=\{r_j | j \in \mathbb{N}\}$. Then the assignments $\rho(\mathrm{a})=0$, $\rho(\mathrm{b})=2$, and $\rho(r_j)=3^{-j}$ yield a numerical representation of any list α of n symbols from F by triadic numbers

$$\rho(\alpha) = \sum_{j=1}^{n} a_j 3^{-j} \quad \text{with} \quad a_j \in \{0,2\}, \tag{22}$$

which constitutes exactly the Cantor set as representation space.

Siegelmann and Sontag[51] used such fractal encoding for the Turing machine tape sequences α, β in the state description Eq. (5), yielding

$$x = \rho(\alpha) = \sum_{j=1}^{n} a_j g^{-j}, \quad y = \rho(\beta) = \sum_{j=1}^{m} b_j g^{-j} \tag{23}$$

with an appropriate base number $g \in \mathbb{N}$. As role vectors for the state description, they chose

$$\boldsymbol{r}_1 = \begin{pmatrix}1\\0\end{pmatrix}, \quad \boldsymbol{r}_2 = 1, \quad \boldsymbol{r}_3 = \begin{pmatrix}0\\1\end{pmatrix},$$

where the role of $\boldsymbol{r}_2$ for the control states $q \in Q$ was simply taken as the scalar constant one. Moreover, the p control states were represented in a local way by p canonical basis vectors $\rho(q)=\boldsymbol{e}_k$ of $\mathbb{R}^p$. The complete tensor product representation of a Turing machine state s is thus

$$\boldsymbol{s} = \rho(s) = (x,0,\ldots,1,\ldots,0,y)^T \tag{24}$$

with 1 in the $k+1$th position encoding the kth control state q.

Other higher-dimensional fractal representations are obtained for arbitrary filler symbols. Assigning to each filler $f_i \in F$ a different vector $\rho(f_i)=\boldsymbol{f}_i \in \mathbb{R}^p$ and using the one-dimensional role representation $\rho(r_j)=2^{-j}$ entails a p-dimensional Sierpinski sponge

$$S = \left\{ \boldsymbol{x} \in [-0.5,0.5]^p | \boldsymbol{x} = \sum_{j=1}^{n} 2^{-j} \boldsymbol{f}_j, \ n \in \mathbb{N} \right\} \tag{25}$$

as representation space.[52,53] Tabor,[52–54] e.g., represented three symbols {a,b,c} by planar vectors

$$\boldsymbol{a} = 2^{-2}\begin{pmatrix}1\\-1\end{pmatrix}, \quad \boldsymbol{b} = 2^{-2}\begin{pmatrix}-1\\-1\end{pmatrix}, \quad \boldsymbol{c} = 2^{-2}\begin{pmatrix}-1\\1\end{pmatrix}. \tag{26}$$

The state space resulting from this assignment in combination with Eq. (25) is the Sierpinski gasket shown in Fig. 5.

Since the Sierpinski gasket can be generated by an iterated function system, Tabor[52–54] proposed a general class of computational dynamical systems, called dynamical automata. Formally, a dynamical automaton is defined as an 8-tuple

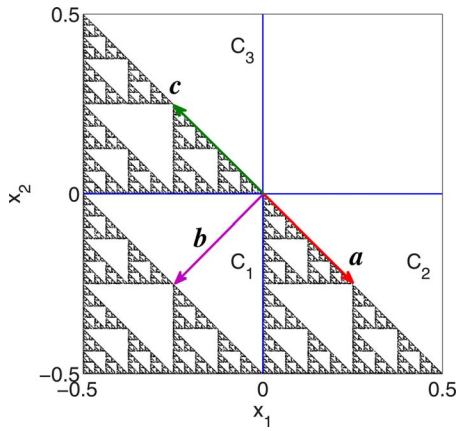

FIG. 5. (Color online) PDDA with stack tape represented by the Sierpinski gasket in the fractal encoding Eq. (25). The vectors $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ are defined in Eq. (26). The blue lines demarcate the partition compartments in Eq. (30).

$$M_{DA} = (X, \mathcal{F}, \mathcal{P}, \mathbf{T}, \iota, \boldsymbol{x}_0, A), \tag{27}$$

where X is a metric space (the automaton's phase space), $\mathcal{F}$ is a finite set of functions, $f_k: X \to X$, $1 \leq k \leq K$, $\mathcal{P}$ is a partition of X into M pairwise disjoint compartments, $C_i \in \mathcal{P}$, that cover the whole phase space X, $\mathbf{T}$ is a finite input alphabet, $\boldsymbol{x}_0 \in X$ is the initial state, and

$$\iota: \mathcal{P} \times \mathbf{T} \times \mathcal{F} \to \{0,1\} \tag{28}$$

is the input mapping specifying for each compartment $C_i \in \mathcal{P}$ and each input symbol $a_j \in \mathbf{T}$ whether a function $f_k \in \mathcal{F}$ is applicable $(\iota(C_i, a_j, f_k) = 1)$ or not $(\iota(C_i, a_j, f_k) = 0)$ when the current state $\boldsymbol{x} \in C_i$. Finally, $A \subset X$ is a region of accepting states of the dynamical automaton. (We refer to the more general definition in Refs. 52 and 54 here.)

Using this description, Fig. 5 illustrates a particular subclass, a pushdown dynamical automaton (PDDA) recognizing the context-free grammar

$$\begin{aligned} &\mathbf{T} = \{\mathrm{a,b,c,d}\}, \quad \mathbf{N} = \{\mathrm{S,A,B,C,D}\}, \\ &P = \{\mathrm{S} \to \mathrm{ABCD}, \mathrm{S} \to \epsilon, \mathrm{A} \to \mathrm{aA}, \mathrm{A} \to \mathrm{a}, \mathrm{B} \to \mathrm{bB}, \mathrm{B} \to \mathrm{b}, \\ &\quad \mathrm{C} \to \mathrm{cC}, \mathrm{C} \to \mathrm{c}, \mathrm{C} \to \mathrm{aS}, \mathrm{C} \to \mathrm{a}, \mathrm{D} \to \mathrm{dS}, \mathrm{D} \to \mathrm{d}\}, \end{aligned} \tag{29}$$

by setting

$$\begin{aligned} &X = [-0.5, 0.5]^2 \setminus [0, 0.5]^2, \\ &\mathcal{F} = \left\{ f_1(\boldsymbol{x}) = \boldsymbol{x} + \begin{pmatrix} 0 \\ 2 \end{pmatrix}, f_2(\boldsymbol{x}) = 2\boldsymbol{x} + \begin{pmatrix} 2 \\ -2 \end{pmatrix}, \right. \\ &\qquad \left. f_3(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x} - \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\}, \\ &\mathcal{P} = \{C_1 = [-0.5, 0] \times [-0.5, 0], C_2 = [0, 0.5] \times [-0.5, 0], \\ &\qquad C_3 = [-0.5, 0] \times [0, 0.5]\}, \\ &\mathbf{T} = \{\mathrm{a,b,c}\} \end{aligned} \tag{30}$$

for ι see Table II,

TABLE II. Admissible input mappings $(\iota(C_i, a_j, f_k) = 1)$ for PDDA defined in Eq. (30) and displayed in Fig. 5.

Compartment(s)	Input	State transition
C_1	b	f_1
C_3	c	f_2
Any	a	f_3

$$\boldsymbol{x}_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad A = \{\boldsymbol{x}_0\}.$$

Table II presents the admitted input mappings where $\iota(C_i, a_j, f_k) = 1$.

Dynamical automata cover a broad range of computational dynamical systems. If, e.g., the number of functions K in $\mathcal{F}$ equals the number of input symbols N in $\mathbf{T}$ and $\iota(C_i, a_j, f_j) = 1$ for all compartments $C_i \in \mathcal{P}$, function f_j is uniquely associated with symbol a_j. In this case, $\mathcal{F}$ is an iterated function system and the dynamical automaton is called dynamical recognizer.[55–57] If, on the other hand, the phase space is the unit square $X = [0,1]^2$, the partition $\mathcal{P}$ is rectangularly generated by Cartesian products of intervals of the x- and y-axes containing the same number of cells as the function set $\mathcal{F}$, and if, furthermore, these functions f_k are piecewise affine linear and if eventually $\iota(C_i, a_j, f_i) = 1$ for all input symbols $a_j \in \mathbf{T}$, then the functions f_k are piecewise affine linear branches of one unique nonlinear map $f: X \to X$ and the symbolic dynamics of f is a generalized shift, as discussed in Sec. II A 1.[45–47] The resulting dynamical automaton does not longer process input symbols directly but rather according to an autonomous nonlinear dynamics given by f. These systems have been called nonlinear dynamical automata.[4,13,58] Finally, if the number of functions $K = 2M$, where M is the number of compartments of the partition, and if the criteria for dynamical recognizers and for nonlinear dynamical automata are both satisfied, one can chose the f_k in such a way to obtain an interacting nonlinear dynamical automaton.[13,59]

3. Gödel representations

Nonlinear dynamical automata as a subclass of dynamical automata are explicitly constructed by two-dimensional tensor product representations. These Gödel encodings are obtained from scalar representations of fillers as integer numbers and fractal roles, respectively.

Let us again consider the list example from Fig. 2(a) with two fillers $F = \{\mathrm{a,b}\}$ and a countable number of list position roles $R = \{r_j | j \in \mathbb{N}\}$. Setting $\rho(\mathrm{a}) = 0$, $\rho(\mathrm{b}) = 1$ and $\rho(r_j) = 2^{-j}$ entails then a representation of a list α with symbols from F as binary numbers

$$\rho(\alpha) = \sum_{j=1}^{n} a_j 2^{-j} \quad \text{with} \quad a_j \in \{0,1\}. \tag{31}$$

More generally, a set of N fillers is mapped by the Gödel encoding onto $N-1$ positive integers. A string or list α of length n of these symbols is then represented by an N-adic rational number

$$\rho(\alpha) = \sum_{j=1}^{n} a_j N^{-j}. \tag{32}$$

The main advantage of Gödel codes is that they could be naturally extended to lists of infinite length, such that $\rho(f_\alpha) = \Sigma_{j=1}^{\infty} a_j N^{-j}$ is then a real number in the unit interval [0, 1]. Moore[46,47] used a Gödel encoding for the state description [Eq. (15)] of a Turing machine by a generalized shift. Decomposing a dotted bi-infinite sequence

$$\gamma = \cdots a_2 a_1 q \,.\, b_1 b_2 \cdots$$

into two one-sided infinite sequences

$$\gamma_L = q a_1 a_2 \cdots, \quad \gamma_R = b_1 b_2 b_3 \cdots \tag{33}$$

and computing their Gödel numbers

$$x = \rho(\gamma_L) = \rho(q) N^{-1} + \sum_{j=1}^{n} \rho(a_j) N^{-j-1},$$
$$y = \rho(\gamma_R) = \sum_{j=1}^{n} \rho(b_j) N^{-j} \tag{34}$$

yields the so-called symbologram representation of the symbol sequence γ and thus of the state description of a Turing machine in the unit square.[60,61] The generalized shift is thereby represented by a piecewise affine linear map whose branches are defined at the domains of dependence of the shift, thus entailing a nonlinear dynamical automaton. Gödel representations were used by beim Graben and co-workers[4,13,58,59] in the field of computational psycholinguistics.

4. Functional representations

The high dimensionality and sparsity of arithmetic tensor product representations should be avoided in dynamic cognitive modeling when recursion or parallelism are involved.[13,14] In such cases, a further generalization of dynamical automata, where the metric space X is an infinite-dimensional Banach or Hilbert space, appears to be appropriate. Such functional representations were suggested for quantum automata by Moore and Crutchfield.[57] Related approaches represent compositional semantics through Hilbert space oscillations,[62–65] or linguistic phrase structure trees through spherical harmonics.[14]

In order to construct a functional representation for our simple example from Fig. 2(a), we assign particular basis functions

$$\rho(\mathrm{a}) = f_{\mathrm{a}}(x) = 1, \quad \rho(\mathrm{b}) = f_{\mathrm{b}}(x) = x \tag{35}$$

and

$$\rho(r_1) = g_1(y) = \sin y, \tag{36}$$

$$\rho(r_2) = g_2(y) = \sin 2y, \quad \rho(r_3) = g_3(y) = \sin 3y \tag{37}$$

to fillers and roles, respectively. Then, the tensor product in function space leads to functions of several variables, e.g., given as

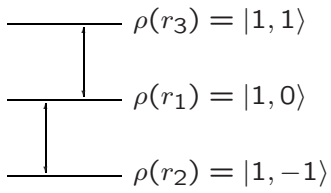

FIG. 6. Tree roles in a spin-one term schema.

$$\rho(\mathrm{b}) \otimes \rho(r_2) = (f_{\mathrm{b}} \otimes g_2)(x,y) = f_{\mathrm{b}}(x) g_2(y) = x \sin 2y. \tag{38}$$

Accordingly, the functional tensor product representation of the list $\alpha = (\mathrm{a},\mathrm{b},\mathrm{a})$ turns out to be

$$\rho(\alpha) = f_{\mathrm{a}}(x) g_1(y) + f_{\mathrm{b}}(x) g_2(y) + f_{\mathrm{a}}(x) g_3(y)$$
$$= \sin y + x \sin 2y + \sin 3y. \tag{39}$$

As another example, we give a functional representation of the logical equivalence relation discussed in Sec. II A 1. Here, we encode the inputs A and B to the truth Table I as fillers $A, B \in \{0,1\}$. The first two input positions are represented by one-dimensional Gauss functions centered around sites $y_1, y_2 \in \mathbb{R}$. In addition, we introduce a third input $G = 1$, acting as a gating variable, bound to another Gaussian centered around a third site $y_3 \in \mathbb{R}$. All three inputs are linearly superimposed in the one-dimensional tensor product representation as

$$\rho(\alpha) = A e^{-R|y-y_1|^2} + B e^{-R|y-y_2|^2} + G e^{-R|y-y_3|^2}, \tag{40}$$

where R is a characteristic spatial scale. In Sec. III C we use a second spatial dimension $x \in \mathbb{R}$ to implement logical inference through traveling pulses with lateral inhibition in a neural field.

Choosing basis functions for functional tensor product representations naively could lead to an explosion of the number of independent variables. This can be avoided by selecting suitable basis functions with particular recursion properties. One of these systems is spherical harmonics used in the angular momentum algebra of quantum systems. Here, the Clebsch–Gordan coefficients allow an embedding of tensor products for coupled spins into the original single particle space.[66] We demonstrate this construction for our example tree T from Fig. 2(d) [Eq. (21)].

In a first step, we identify the three fillers $F = \{\mathrm{S},\mathrm{a},\mathrm{b}\}$ with three roots of unity

$$\omega_k = \frac{2\pi k}{3} \tag{41}$$

leading to the harmonic oscillations in the variable x,

$$f_k(x) = e^{i\omega_k x}. \tag{42}$$

Then, we regard the tree in Fig. 4 as a "deformed" term schema for a spin-one triplet as in Fig. 6.

Figure 6 indicates that the three role positions r_1, r_2, r_3 [Eq. (7)] in a labeled binary tree are represented by three z-projections of a spin-one particle,

$$\rho(r_2) = |1,-1\rangle, \quad \rho(r_1) = |1,0\rangle, \quad \rho(r_3) = |1,1\rangle, \tag{43}$$

which have an $L^2(S)$ representation by spherical harmonics

$$|j,m\rangle = Y_{jm}(\mathbf{y}) \tag{44}$$

with $\mathbf{y}=(\vartheta,\varphi)$ and $\vartheta\in[0,\pi]$, $\varphi\in[0,2\pi[$.

For treating complex phrase structure trees, we compute the tensor products of role vectors $\rho(r_i)\otimes\rho(r_j)$. Inserting the spin eigenvectors from Eq. (43) yields expressions such as

$$|j_1,m_1\rangle\otimes|j_2,m_2\rangle = |j_1,m_1,j_2,m_2\rangle, \tag{45}$$

well known from the angular momentum coupling in quantum mechanics.[66]

In quantum mechanics, the product states [Eq. (45)] generally belong to different multiplets, which are given by the irreducible representations of the spin algebra $sl(2)$. These are obtained by the Clebsch–Gordan coefficients in the expansions

$$|j,m,j_1,j_2\rangle = \sum_{m_1,m_2=m-m_1}\langle j_1,m_1,j_2,m_2|j,m,j_1,j_2\rangle \times|j_1,m_1,j_2,m_2\rangle, \tag{46}$$

where the total angular momentum j obeys the triangle relation

$$|j_1-j_2|\le j\le j_1+j_2. \tag{47}$$

In order to describe recursive tree generation by wave functions of only one (spherical) variable $\mathbf{y}=(\vartheta,\varphi)$, we invert Eq. (46), leading to

$$|j_1,m_1,j_2,m_2\rangle = \sum_{j=|j_1-j_2|}^{j_1+j_2}\langle j,m,j_1,j_2|j_1,m_1,j_2,m_2\rangle|j,m,j_1,j_2\rangle \tag{48}$$

with the constraint $m=m_1+m_2$.

Equation (48) has to be applied recursively for obtaining the role positions of more and more complex phrase structure trees. Finally, a single tree is represented by its filler/role bindings in the basis of spherical harmonics after contraction over $|j_1,j_2\rangle$,

$$\rho(T) = \sum_{jkm} a_{jkm} f_k(x) Y_{jm}(\mathbf{y}), \tag{49}$$

where the coefficients $a_{jkm}=0$ if filler k is not bound to pattern Y_{jm}. Otherwise, the a_{jkm} encode the Clebsch–Gordan coefficients in Eq. (48).

Now we are able to construct the functional tensor product representation for the tree in Fig. 2(d). Its representation in algebraic form was obtained in Eq. (21). Inserting the spin representation for the roles yields

$$\begin{aligned}&\rho(\mathrm{S})|1,0\rangle + \rho(\mathrm{S})|1,0,1,-1\rangle + \rho(\mathrm{a})|1,-1,1,-1\rangle\\ &\quad+\rho(\mathrm{b})|1,1,1,-1\rangle + \rho(\mathrm{S})|1,0,1,1\rangle + \rho(\mathrm{a})|1,-1,1,1\rangle\\ &\quad+\rho(\mathrm{b})|1,1,1,1\rangle.\end{aligned} \tag{50}$$

Expressing the tensor products by Eq. (48) yields first

$$\begin{aligned}|1,0\rangle|1,-1\rangle &= |1,0,1,-1\rangle\\ &= \sum_{j=0}^{2}\langle j,-1,1,1|1,0,1,-1\rangle|j,-1,1,1\rangle\\ &= \langle 0,-1,1,1|1,0,1,-1\rangle|0,-1,1,1\rangle\\ &\quad+\langle 1,-1,1,1|1,0,1,-1\rangle|1,-1,1,1\rangle\\ &\quad+\langle 2,-1,1,1|1,0,1,-1\rangle|2,-1,1,1\rangle.\end{aligned}$$

The first Clebsch–Gordan coefficient $\langle 0,-1,1,1|1,0,1,-1\rangle=0$ as a spin $j=0$ particle does not permit an $m=-1$ projection. The two other Clebsch–Gordan coefficients are $\langle 1,-1,1,1|1,0,1,-1\rangle=\langle 2,-1,1,1|1,0,1,-1\rangle=1/\sqrt{2}$.

Correspondingly, we obtain for

$$\begin{aligned}|1,-1\rangle|1,-1\rangle &= |1,-1,1,-1\rangle\\ &= \sum_{j=0}^{2}\langle j,-2,1,1|1,-1,1,-1\rangle|j,-2,1,1\rangle\\ &= \langle 0,-2,1,1|1,-1,1,-1\rangle|0,-2,1,1\rangle\\ &\quad+\langle 1,-2,1,1|1,-1,1,-1\rangle|1,-2,1,1\rangle\\ &\quad+\langle 2,-2,1,1|1,-1,1,-1\rangle|2,-2,1,1\rangle.\end{aligned}$$

Here, the first two Clebsch–Gordan coefficients vanish because spins $j=0$ and $j=1$ forbid $m=-2$. Therefore, only $\langle 2,-2,1,1|1,-1,1,-1\rangle=1$ accounts for this state.

Finally, we consider

$$\begin{aligned}|1,1\rangle|1,-1\rangle &= |1,1,1,-1\rangle\\ &= \sum_{j=0}^{2}\langle j,0,1,1|1,1,1,-1\rangle|j,0,1,1\rangle\\ &= \langle 0,0,1,1|1,1,1,-1\rangle|0,0,1,1\rangle\\ &\quad+\langle 1,0,1,1|1,1,1,-1\rangle|1,0,1,1\rangle\\ &\quad+\langle 2,0,1,1|1,1,1,-1\rangle|2,0,1,1\rangle.\end{aligned}$$

Here, $m=0$ is consistent with $j=0,1,2$ such that all three terms have to be taken into account through $\langle 0,0,1,1|1,1,1,-1\rangle=1/\sqrt{3}$, $\langle 1,0,1,1|1,1,1,-1\rangle=1/\sqrt{2}$, and $\langle 2,0,1,1|1,1,1,-1\rangle=1/\sqrt{6}$.

Thus, we have constructed the functional representation of the left subtree of Fig. 2(d). The corresponding expressions for the right subtree were derived in Ref. 14. The complete spherical wave representation of the tree in Fig. 2(d) is then

$$\begin{aligned}\rho(T) &= f_{\mathrm{S}}(x)Y_{1,0}(\mathbf{y}) + \tfrac{f_{\mathrm{S}}(x)}{\sqrt{2}}(Y_{1,-1}(\mathbf{y})+Y_{2,-1}(\mathbf{y})) + f_{\mathrm{a}}(x)Y_{2,-2}(\mathbf{y})\\ &\quad+f_{\mathrm{b}}(x)\left(\tfrac{1}{\sqrt{3}}Y_{0,0}(\mathbf{y})+\tfrac{1}{\sqrt{2}}Y_{1,0}(\mathbf{y})+\tfrac{1}{\sqrt{6}}Y_{2,0}(\mathbf{y})\right)\\ &\quad+\tfrac{f_{\mathrm{S}}(x)}{\sqrt{2}}(Y_{2,1}(\mathbf{y})-Y_{1,1}(\mathbf{y})) + f_{\mathrm{a}}(x)\left(\tfrac{1}{\sqrt{3}}Y_{0,0}(\mathbf{y})-\tfrac{1}{\sqrt{2}}Y_{1,0}(\mathbf{y})\right.\\ &\quad\left.+\tfrac{1}{\sqrt{6}}Y_{2,0}(\mathbf{y})\right)+f_{\mathrm{b}}(x)Y_{2,2}(\mathbf{y}),\end{aligned} \tag{51}$$

where the fillers are functionally represented through Eq. (42).

5. Representation theory

Up to this point, we gave on overview about different state space representations for complex symbolic data structures that could be regarded as formal models of mental or cognitive states. Cognitive processes, by contrast, were compared with algorithms according to the computer metaphor of the mind. Algorithms are generally sequences of instructions, or more specifically, of symbolic operations acting

upon data structures and symbolic states. Thus, a cognitive process $P=A_1;A_2;\ldots;A_n$ composed of n cognitive operations A_i can be seen as the concatenation product[67–69] (indicated by ";") of partial functions $A_i:F_\infty \to F_\infty$, where F_∞ denotes the set of filler/role bindings for symbolic expressions as introduced in Sec. II B. Two operations A_i and A_j can be concatenated to another operator $P=A_i;A_j$ if the image of A_j is contained in the domain of A_i. Under this restriction, the operators A_i form an algebraic semigroup since the concatenation product is associative.

Having constructed a state space representation $\mathcal{F}=\rho(F_\infty)$ of symbolic data structures, the mapping ρ can be formally extended to the operators acting on F_∞. Cognitive operations A_i and A_j become thereby represented by mappings $\rho(A_i)$ and $\rho(A_j)$ such that

$$\rho(A_i;A_j) = \rho(A_i) \circ \rho(A_j), \tag{52}$$

where "$\circ$" denotes the usual functional composition in representation space, defined as $(f\circ g)(x)=f(g(x))$. Therefore, the mapping ρ preserves the semigroup structure of cognitive operations and can be properly regarded as a semigroup representation in the sense of algebraic representation theory.[16,17]

Interestingly, Fock space representations of cognitive operations are piecewise affine linear mappings[9–13,46,47,52–54,58,59] resulting in globally nonlinear maps at representation space. We present three illustrative examples in Secs. III and IV. For cognitive operators are partial functions, their representatives could be pasted together in several ways entailing nonlinear maps. The explicit construction of such maps from the well-known linear pieces establishes the inverse problem for dynamic cognitive modeling. Moreover, since many solutions of the inverse problem are, in principle, possible, their stability against parametric perturbation is of great importance. If solutions are highly unstable, the inverse problem is ill-posed. We discuss these issues in the remainder of the paper in some detail.

However, we first address another, related problem, posed by Spivey and Dale.[15] Symbolic operations are time discrete. When a cognitive process $P=A_1;A_2;\ldots;A_n$ applies to a symbolic initial state $s_0 \in F_\infty$ at time $t=0$, A_n brings s_0 into another state $s_1=A_n(s_0)$ at time $t+1$ and so on. By contrast, brain dynamics is continuous in time. If $\rho(P)=\rho(A_1)\circ\rho(A_2)\circ\cdots\circ\rho(A_n)$ is the representation of P and $v_t=\rho(s_t) \in \mathcal{F}$ are representations of the symbolic states at time t, we have to embed this time discrete dynamics of duration L into continuous time.

A common approach is a separation ansatz

$$v(\boldsymbol{x},t) = \sum_{k=1}^{L} \lambda_k(t) v_k(\boldsymbol{x}) \tag{53}$$

for a functional representation. Here, $v_k(\boldsymbol{x})$ denotes the discrete cognitive state v_k at time k represented by a function over feature space D and $\lambda_k(t)$ its corresponding time-dependent amplitude. Equation (53) is often referred to as an order parameter ansatz.[70] The amplitudes $\lambda_k(t)$ are then governed by ordinary differential equations describing the continuous time dynamics of the cognitive states $v(\boldsymbol{x},t)$.

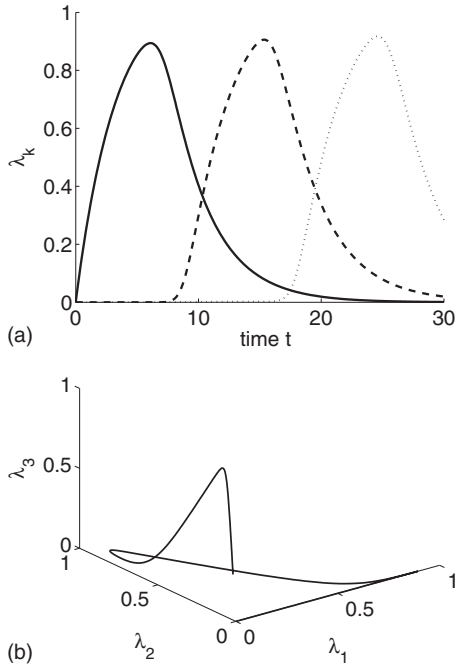

FIG. 7. Transient amplitude dynamics of three cognitive states. (a) Time course of $\lambda_1(t)$ (solid), $\lambda_2(t)$ (dashed), and $\lambda_3(t)$ (dotted), obeying Eqs. (54)–(57). (b) Three-dimensional phase portrait. Parameters as in Ref. 14.

In order to describe a transient dynamics of duration $T=3$, where one cognitive state gradually excites its successor,[14] we made the ansatz

$$\tau\frac{d\lambda_k(t)}{dt} + \lambda_k(t) = g_k(\lambda_0(t),\lambda_1(t),\ldots,\lambda_T(t)) \tag{54}$$

with delayed couplings

$$g_0(t) = w \cdot f_{\eta,\beta}\left(\frac{1.5\Delta - t}{\Delta}\right), \tag{55}$$

$$g_l(\lambda_1,\ldots,\lambda_{l-1})(t) = w \cdot f_{\eta,\beta}(\lambda_{l-1}(t-\Delta)), \quad l \geq 1 \tag{56}$$

for $t \geq 0$. Here, the sigmoidal logistic function f with threshold η and gain β is defined as

$$f_{\eta,\beta}(z) = \frac{1}{1+e^{-\beta(z-\eta)}}, \tag{57}$$

where w, η, β, and τ are real positive constants.

Figure 7 displays the dynamics of the amplitudes $\lambda_k(t)$ of three succeeding states v_1, v_2, and v_3. Figure 7(a) shows the time courses of $\lambda_k(t)$, while Fig. 7(b) depicts a three-dimensional phase portrait. Obviously, the cognitive states v_k correspond to the maxima of their respective amplitudes. In phase space, they appear as saddle points, attracting states from the direction of its precursor and repelling them toward the direction of its successor. Thus, the unstable separatrices of these saddle points are connected with each other forming a heteroclinic sequence.[71,72] Rabinovich and co-workers[71–74] suggested stable heteroclinic sequences and stable heteroclinic channels as a universal account for transient cognitive computations.

Regarding "pure" tensor product states as saddle points in continuous time dynamics implies that these symbolically

meaningful, representational states will never be reached by the system's evolution. Instead, the trajectory approaches these states from one direction and diverges into another direction. Thus, there is a trade-off between continuous time dynamics and the interpretation of the system's behavior as algorithmic symbol processing. In this sense, continuous time dynamics does not implement symbolic processing, it rather approximates it. Thus, one can call dynamics and algorithmic processing incompatible with each other.[75]

However, this kind of incompatibility could be easily resolved assuming that tensor product states are prototypes of symbolic representations in phase space. These prototypes define equivalence classes and thereby phase space partitions. Then, symbolic states can be identified with partition cells entered by the system's trajectory, instead of with single points in phase space. Consequently, a symbolic dynamics as discussed in Secs. II and III is obtained, providing an implementation of algorithmic processing through the change of the ontology. However, this implementation could be incompatible with the phase space dynamics unless the partition is generating.[4,76–78]

C. Neurodynamics

Following the top-down path of the three tier approach for dynamic cognitive modeling, the third and final steps comprise the above-mentioned implementation of a state space representation (step two) constructed from the symbolic description (step one). In the following, we use the term "implementation" neutrally as inspired by quantum theory, where a symmetry is implemented at the Fock space of quantum fields,[17] thereby disregarding the philosophical controversy about whether connectionist models are "mere implementations" of cognitive architectures.[4,11,12,36,75,79–83]

A cognitive model is implemented by equipping its representation space X with a flow $\Phi_t: X \to X$ solving dynamical equations in time t. If X is a finite-dimensional vector space $\mathbb{R}^p$, these equations are usually ordinary differential (or difference) equations for the state vector $\boldsymbol{u}(t)$. If, conversely, X is an infinite-dimensional function space resulting from a functional representation, the states are fields $u(\boldsymbol{x},t)$ obeying either partial differential equations or, more generally, integrodifferential equations. It is of crucial importance for dynamic cognitive modeling that such implementations are guided by principles from the neurosciences.

Under these guiding principles, we consider ANN models or connectionist architectures in the first case of finite-dimensional representations.[5,6,9–12,84–86] On the other hand, neural or dynamic field models are investigated in the second case of infinite-dimensional, i.e., functional, representations.[18–34]

1. Neural networks

A common choice for the dynamics of neural networks are population rate models, where the ith component $u_i(t)$ of the state vector $\boldsymbol{u}(t) \in X \subset \mathbb{R}^p$ describes the firing rate or the firing probability either of a single neuron or of a small neural population i in the network. The so-called leaky integrator models[6,84,85,87,88] are governed by equations of the form

$$\tau \frac{du_i(t)}{dt} + u_i(t) = \sum_{j=1}^{p} w_{ij} f(u_j(t)), \tag{58}$$

where $u_i(t)$ is the time-dependent membrane potential of the ith neuron in a network of p units. The activation function f describes the conversion of the membrane potential $u_i(t)$ into a spike train $r_i(t)=f(u_i(t))$. The left-hand side of Eq. (58) characterizes the intrinsic dynamics of a leaky integrator unit, i.e., an exponential decay of membrane potential with time constant $\tau>0$. The right-hand side of Eq. (58) represents the net input to unit i: the weighted sum of activity delivered by all units j that are connected with unit $i(j \to i)$. Therefore, the synaptic weight matrix $W=(w_{ij})$ comprises three different kinds of information: (1) unit j is connected with unit i if $w_{ij} \neq 0$ (connectivity, network topology), (2) the synapse $j \to i$ is excitatory $(w_{ij}>0)$, or inhibitory $(w_{ij}<0)$, and (3) the strength of the synapse is given by $|w_{ij}|$.

There are essentially two important network topologies. If the synaptic weight matrix indicates a preferred direction of propagation, the network has feed-forward topology. If, on the other hand, no preferred direction is indicated, the network is recurrent.

Activation functions f depend on particular modeling purposes. The most common ones are either linear $(f(z)=z)$ or sigmoidal as in Eq. (57), which describes the stochastic all-or-nothing law of neural spike generation.[19]

Neural network models for cognitive processes have a long tradition.[5,6,11,89–92] Siegelmann and Sontag[51] used the tensor product representation [Eq. (24)] to prove that a recurrent neural network of about 900 units implements a Turing machine. By contrast, using the Gödel representation [Eq. (34)], Moore[46,47] and Siegelmann[45] demonstrated that a Turing machine can, in fact, be implemented as a low-dimensional neural network. Pollack[55] and Tabor[52,53] used cascaded neural networks for implementing dynamical recognizers and dynamical pushdown automata [Eq. (30)] for syntactic language processing (for a local representation, see Ref. 93). These have been recently generalized by Tabor[54,94] to fractal learning neural networks. In order to model word prediction dynamics, Elman[95] suggested simple recurrent neural networks. This architecture became very popular in recent years in the domain of language processing.[96–99] Kawamoto[100] used a Hopfield net[101] with exponentially decaying activation and habituating synaptic weights to describe lexical ambiguity resolution. Similar architectures are Smolensky's harmony machines[11,12,83,102] and Haken's synergetic computers.[70,103,104] Vosse and Kempen[105] described a sentence processor through local inhibition in a unification space. Neural network applications for logic and semantic processing were proposed in Refs. 49, 50, 62–65, and 106–109. For further references, see also Refs. 39, 87, and 110.

2. Neural field models

Analyzing and simulating large neural networks with complex topology is a very hard problem[88,111–117] due to the nonlinearity of the activation function and the large number of synaptic weights. Instead of computing the sum in the right-hand side of Eq. (58), a continuum limit significantly

facilitates analytical treatment and numeric simulations. Such continuum approximations of neural networks have been proposed since the 1960s.[18–31]

Starting with the leaky integrator network [Eq. (58)], the sum over all units at the right-hand side is replaced by an integral transformation of a neural field quantity $u(\boldsymbol{x},t)$, where the continuous parameter $\boldsymbol{x}\in\mathbb{R}^p$ now indicates the position i in the network. Correspondingly, the synaptic weight matrix w_{ij} turns into a kernel function $w(\boldsymbol{x},\boldsymbol{y})$. Then, Eq. (58) assumes the form of the Amari equation[18,30] investigated in neural field theory.

$$\tau\frac{\partial u(\boldsymbol{x},t)}{\partial t}+u(\boldsymbol{x},t)=\int_D w(\boldsymbol{x},\boldsymbol{y})f(u(\boldsymbol{y},t))d\boldsymbol{y},\quad \boldsymbol{x}\in D,\quad t>0. \tag{59}$$

Interpreting the domain $D\subset\mathbb{R}^p$ of the Amari equation (59) not as a physical substrate realized by actual biological neurons, but rather as an abstract features space for computational purposes, one speaks about dynamic field theory.[32–34]

Neural and dynamic field theory, respectively, found several applications in dynamic cognitive modeling. Jirsa and Haken[25,118] and Jirsa *et al.*[119] modeled changes in magnetoencephalographic (MEG) activity during bimanual movement coordination, while Bressloff and co-workers[20,120–125] described the pattern formation and the emergence of hallucinations in primary visual cortex. Infant habituation was modeled by Schöner and colleagues[32–34] through decaying dynamic fields defined over the visual field of a children performing memory tasks (see also Ref. 3 for a review).

3. Inverse problems for neurodynamics

Up to a few examples, where neural architectures or neural fields can be explicitly designed, the top-down approach of dynamic cognitive modeling either provides discrete sequences of training patters $v_k(\boldsymbol{x})\in X$ or continuous paths $v(\boldsymbol{x},t)=\Sigma_k\lambda_k v_k(\boldsymbol{x})$ [Eq. (59)] in representation space that have to be reproduced by the state evolution $u(\boldsymbol{x},t)$. If the dynamics is given either by the leaky integrator [Eq. (58)] for a neural network or by the Amari equation (59) for a neural/dynamic field, the crucial task is the determination of the system parameters. This is again the inverse problem for dynamic cognitive modeling: given a prescribed trajectory $v(\boldsymbol{x},t)$, then find the synaptic weight matrix $\boldsymbol{W}=(w_{ij})$ or the synaptic weight kernel $w(\boldsymbol{x},\boldsymbol{y})$, respectively, such that $u(\boldsymbol{x},t)=v(\boldsymbol{x},t)$ solves the dynamical law [Eq. (58) or (59)] for all $t>0$ with initial condition $u(\boldsymbol{x},0)=v(\boldsymbol{x},0)$.

Here, we discuss the inverse problem in the framework of the Amari equation (59). We prescribe one or several complete time-dependent patterns $v_\xi(\boldsymbol{x},t)$, $\xi=1,\ldots,n$ for $x\in D$, $t\geq 0$ with some domain $D\subset\mathbb{R}^p$. For our further discussion, we assume that the nonlinear activation function $f:\mathbb{R}\rightarrow[0,1]$ is known. Then, we search for kernels $w(\boldsymbol{x},\boldsymbol{y})$ for $\boldsymbol{x},\boldsymbol{y}\in D$ such that the solutions of the Amari equation with initial conditions $u(\boldsymbol{x},0)=v_\xi(\boldsymbol{x},0)$ satisfies $u(\boldsymbol{x},t)=v_\xi(\boldsymbol{x},t)$ for $\boldsymbol{x}\in D$, $t\geq 0$, and $\xi=1,\ldots,n$.

As a first step, we transform Eq. (59) into a linear integral equation. Defining

$$\varphi_\xi(\boldsymbol{x},t)=f(v_\xi(\boldsymbol{x},t)),\quad \boldsymbol{x}\in D,\quad t\geq 0, \tag{60}$$

$$\psi_\xi(\boldsymbol{x},t)=\tau\frac{\partial v_\xi(\boldsymbol{x},t)}{\partial t}+v_\xi(\boldsymbol{x},t),\quad \boldsymbol{x}\in D,\quad t\geq 0, \tag{61}$$

and employing the integral operator

$$(W\varphi)(\boldsymbol{x})=\int_D w(\boldsymbol{x},\boldsymbol{y})\varphi(\boldsymbol{y})d\boldsymbol{y},\quad \boldsymbol{x}\in D, \tag{62}$$

leads to a reformulation of the inverse problem into the equation

$$\psi_\xi(\boldsymbol{x},t)=W\varphi_\xi(\boldsymbol{x},t),\quad \boldsymbol{x}\in D,\ t\geq 0,\quad \xi=1,\ldots,n, \tag{63}$$

where here the kernel $w(\boldsymbol{x},\boldsymbol{y})$, $\boldsymbol{x},\boldsymbol{y}\in D$ of the linear integral operator W is unknown. Equation (63) is *linear* in the kernel w. It can be rewritten as

$$\boldsymbol{\psi}=W\boldsymbol{\varphi} \tag{64}$$

with

$$\boldsymbol{\varphi}=(\varphi_1,\ldots,\varphi_n),\quad \boldsymbol{\psi}=(\psi_1,\ldots,\psi_n).$$

For every fixed $\boldsymbol{x}\in D$, we can rewrite Eq. (63) as

$$\psi_{\boldsymbol{x}}(t)=\int_D \varphi(\boldsymbol{y},t)w_{\boldsymbol{x}}(\boldsymbol{y})d\boldsymbol{y},\quad t\geq 0 \tag{65}$$

with

$$w_{\boldsymbol{x}}(\boldsymbol{y})=w(\boldsymbol{x},\boldsymbol{y}),\ \boldsymbol{x},\boldsymbol{y}\in D,\ \psi_{\boldsymbol{x}}(t)=\psi(\boldsymbol{x},t),\ \boldsymbol{x}\in D,\ t\geq 0.$$

We define

$$(Vg)(t)=\int_D k(t,\boldsymbol{y})g(\boldsymbol{y})d\boldsymbol{y},\quad t\geq 0$$

with the particular choice $k(t,\boldsymbol{y})=\varphi(\boldsymbol{y},t)$ to write Eq. (65) as

$$\psi_{\boldsymbol{x}}=Vw_{\boldsymbol{x}},\quad \boldsymbol{x}\in D. \tag{66}$$

If φ is continuous in $\boldsymbol{y}$ and t, then for fixed $\boldsymbol{x}$, Eq. (66) is a Fredholm integral equation (58) of the first kind with continuous kernel φ. This equation is known to be ill-posed,[126] i.e., we do not have (a) existence or (b) uniqueness in general and even if we have uniqueness, the solution does not depend in a (c) stable way on the right-hand side.

Fredholm integral equations of the form

$$\int_D k(t,\boldsymbol{y})g(\boldsymbol{y})d\boldsymbol{y}=f(t)$$

with continuous kernel $k(t,\boldsymbol{y})$ are well studied in mathematical literature (cf., for example, Refs. 126–128). The analysis is built on the properties of compact linear operators between Hilbert spaces. Consider a compact linear operator $A:X\rightarrow Y$ with Hilbert spaces X, Y, and its adjoint A^*. Then, the nonnegative square roots of the eigenvalues of the self-adjoint compact operator $A^*A:X\rightarrow X$ are called the singular values of the operator A. The singular value decomposition[126] leads to a representation of the operator as a multiplication operator within two orthonormal systems $\{g_n|n\in\mathbb{N}\}$ in X and $\{f_n|n\in\mathbb{N}\}$ in Y, such that

$$Ag = \sum_{n=1}^{\infty} \mu_n \langle g, g_n \rangle f_n \tag{67}$$

for $g \in X$. The representation (67) is also known as spectral representation of the operator A. For the orthonormal systems g_n and f_n we have[127]

$$Ag_n = \mu_n f_n, \quad A^* f_n = \mu_n g_n. \tag{68}$$

If A is injective, the inverse of A is given by

$$A^{-1} f = \sum_{n=1}^{\infty} \frac{1}{\mu_n} \langle f, f_n \rangle g_n. \tag{69}$$

If A is not injective, the inverse A^{-1} defined in Eq. (69) projects onto the orthogonal space $N(A)^{\perp} = \{g \,|\, \langle g, \tilde{g} \rangle = 0$ for all $\tilde{g} \in N(A)\}$.

For compact operators A, it is well known[128] that the singular values build a sequence which at most accumulates at zero. The instability of the inverse problem is then caused by behavior

$$\left| \frac{1}{\mu_n} \right| \to \infty, \quad n \to \infty,$$

amplifying small errors when the inverse is applied in a naive way. The classical approach to remedy the instability is known as regularization.[126] In terms of the spectral inverse [Eq. (69)], we can easily understand the main idea of regularization by damping of the factors $1/\mu_n$ for large n, i.e., by replacing $1/\mu_n$ by q_n which is bounded for $n \in \mathbb{N}$. This leads to a modified inverse operator

$$R_\alpha f = \sum_{n=1}^{\infty} q_n^{(\alpha)} \langle f, f_n \rangle g_n.$$

The choice

$$q_n^{(\alpha)} = \begin{cases} \dfrac{1}{\mu_n}, & n \le 1/\alpha \\ 0, & \text{otherwise} \end{cases} \tag{70}$$

is known as spectral cutoff. An alternative to the abrupt cut in Eq. (70) is the Tikhonov regularization

$$q_n^{(\alpha)} = \frac{\mu_n}{\alpha + \mu_n^2}, \quad n \in \mathbb{N}. \tag{71}$$

Here, the parameter α is known as regularization parameter. Boundedness of $q_n^{(\alpha)}$ can be easily obtained by elementary calculation. For both cases, we have

$$q_n^{(\alpha)} \to \frac{1}{\mu_n} \quad \text{for} \quad \alpha \to 0$$

for each fixed $n \in \mathbb{N}$. This can be used to prove (see Ref. 126) that

$$R_\alpha A g \to g \quad \text{for} \quad \alpha \to 0, \tag{72}$$

i.e., in the case of exact data $f = Ag$, the regularized solution converges toward the true solution when the regularization parameter α tends to zero. This can be seen by using $\langle Ag, f_n \rangle = \langle g, A^* f_n \rangle = \mu_n \langle g, g_n \rangle$ and split of the sum into

$$\begin{aligned} R_\alpha A g &= \sum_{n=1}^{\infty} q_n^{(\alpha)} \langle Ag, f_n \rangle g_n \\ &= \sum_{n=1}^{\infty} q_n^{(\alpha)} \mu_n \langle g, g_n \rangle g_n \\ &= \sum_{n=1}^{N} \cdots + \sum_{n=N+1}^{\infty} \cdots . \end{aligned} \tag{73}$$

Since $q_n^{(\alpha)} \mu_n$ is uniformly bounded for all $\alpha > 0$, using the *Cauchy–Schwarz inequality*, we have

$$\left\| \sum_{n=N+1}^{\infty} q_n^{(\alpha)} \mu_n \langle g, g_n \rangle g_n \right\|^2 \le \sum_{n=N+1}^{\infty} |q_n^{(\alpha)} \mu_n|^2 |\langle g, g_n \rangle|^2 \to 0 \quad \text{for} \quad N \to \infty. \tag{74}$$

Further, since for every fixed $n \in \mathbb{N}$

$$q_n^{(\alpha)} \to 1 \quad \text{for} \quad \alpha \to 0,$$

we obtain

$$\sum_{n=1}^{N} q_n^{(\alpha)} \langle g, g_n \rangle g_n \to \sum_{n=1}^{N} \langle g, g_n \rangle g_n \quad \text{for} \quad \alpha \to 0. \tag{75}$$

Combining Eqs. (73)–(75) we obtain Eq. (72) by arguing that we first choose N to make the tail $\Sigma_{n=N+1}^{\infty} \cdots$ sufficiently small uniformly for all α and then letting α tend to zero.

Clearly, if we do not have exact data $f = Ag$, but some right-hand side $f^{(\delta)}$ polluted by numerical or measurement error, then we cannot carry out the limit $\alpha \to 0$. In this case we can split the total error of the reconstruction into two parts

$$\|R_\alpha f^{(\delta)} - g\| = \underbrace{\|R_\alpha f^{(\delta)} - R_\alpha A g\|}_{\text{data error}} + \underbrace{\|R_\alpha A g - g\|}_{\text{reconstruction error}}.$$

If α becomes small, the reconstruction error will decrease and tend to zero, but the data error will become large. If α is large, then the data error is getting smaller, but the reconstruction error increases. Somewhere for medium size α, we obtain a minimum of the total error. Many methods for the automatic choice of the regularization parameter have been developed.[128,129]

Tikhonov regularization is a very general scheme which can be derived not only via the spectral approach but also by matrix algebra or by optimization for solving an ill-posed equation $Vg = f$. Clearly, the equation $Vg = f$ is equivalent to the minimization

$$\min_{g \in X} \|Vg - f\|, \tag{76}$$

where X denotes some appropriate Hilbert space X, for example, the space $L^2(D)$ of square integrable functions on some domain D. The normal equations (cf. Ref. 130) for the minimization problem are given by

$$V^*Vg = V^*f.$$

The operator V^*V does not have a bounded inverse. Stabilization is reached by adding a small multiple of the identity operator I,[130] i.e., by solving

$$(\alpha I + V^*V)g = V^*f, \tag{77}$$

which corresponds to adding a stabilization term $\alpha\|g\|^2$ to the minimization [Eq. (76)], leading to the third form of the Tikhonov regularization

$$\min_{g\in X}(\|Vg - f\|^2 + \alpha\|g\|^2). \tag{78}$$

The operator [Eq. (77)] is usually discretized by standard procedures and then leads to a matrix equation which can be solved either directly or by iterative methods.[130]

The Moore–Penrose pseudoinverse is given by the limit of the Tikhonov regularization for $\alpha\to 0$, i.e., it is

$$V^\dagger = (V^*V)^{-1}V^* = \lim_{\alpha\to 0}(\alpha I + V^*V)^{-1}V^*. \tag{79}$$

However, as discussed above, this limit will lead to satisfactory reconstructions only for well-posed problems. For the above-mentioned ill-posed inverse problem for dynamic cognitive modeling, we have to employ $\alpha > 0$.

These techniques were applied to neural pulse construction problems by Potthast and beim Graben.[131,132] In particular, the authors demonstrated the feasibility of regularized construction techniques for synaptic kernel construction. Properties of the solutions were investigated and the ill-posedness of the problem was proven and demonstrated by particular examples.

The construction of synaptic weight kernels w for the Amari equation (59) is a linear problem and can be solved by linear methods. In Sec. III, we generalize the well-known linear Hebb rule to neural field models described by Eq. (59). However, if the minimization problems (76) and (78), respectively, are more complex, also involving, besides the kernel w, space-dependent time constants $\tau = \tau(\boldsymbol{x})$ or driving forces $p(\boldsymbol{x},t)$ added to the right-hand side of Eq. (59), the inverse problem becomes highly nonlinear and different tools are required. Here, we briefly generalize the standard backpropagation algorithm[6,5] to neural field models.

Time-dependent recurrent backpropagation[6,87,133–137] minimizes an error function for leaky integrator networks [Eq. (58)]. Its generalization aims at minimizing the error functional

$$\min_w \int_0^T \int_D \frac{1}{2}(u(\boldsymbol{x},t) - v(\boldsymbol{x},t))^2 d\boldsymbol{x}dt \tag{80}$$

between a prescribed field $v(\boldsymbol{x},t)$ and the w-dependent solution of the Amari equation $u(\boldsymbol{x},t)[w]$. This problem can be tackled using field-theoretic variational calculus for the minimization problem (80) under the constraint (59) by introducing Lagrange multipliers $\lambda(\boldsymbol{x},t)$ and combining Eqs. (59) and (80) into the functional

$$J = \int_0^T \int_D \left[\frac{1}{2}(u(\boldsymbol{x},t) - v(\boldsymbol{x},t))^2 + \lambda(\boldsymbol{x},t)\{\tau(\boldsymbol{x})\dot{u}(\boldsymbol{x},t) + u(\boldsymbol{x}) - (Wf(u))(\boldsymbol{x},t)\}\right] d\boldsymbol{x}dt. \tag{81}$$

By partial integration of the time derivative $\dot{u}$ and incorporating an additional condition $\lambda(\boldsymbol{x},T)=0$ on λ, we transform Eq. (81) into

$$J = \int_0^T \int_D \left[\frac{1}{2}(u(\boldsymbol{x},t) - v(\boldsymbol{x},t))^2 - \dot{\lambda}(\boldsymbol{x},t)\tau(\boldsymbol{x})u(\boldsymbol{x},t) + \lambda(\boldsymbol{x},t)u(\boldsymbol{x},t) - f(u)(\boldsymbol{x},t)(W^*\lambda)(\boldsymbol{x},t)\right] d\boldsymbol{x}dt + \int_D \lambda(\boldsymbol{x},0)\tau(x)u(\boldsymbol{x},0)d\boldsymbol{x}, \tag{82}$$

where we took the adjoint of W with respect to the scalar product over D.

Since the Amari equation needs to be satisfied, the partial derivative of J with respect to λ should be zero at the minimum. Thus, we are free in the choice of λ, which we choose such that the partial derivative of J with respect to u is zero. We further remark that at time $t=0$, the variation in u should vanish, i.e., the last term in Eq. (82) will not contribute to the partial derivative of u. This leads to the adjoint equation for the backpropagated error signal λ,

$$\dot{\lambda}(\boldsymbol{x},t)\tau(\boldsymbol{x}) - \lambda(\boldsymbol{x},t) = u(\boldsymbol{x},t) - v(\boldsymbol{x},t) - f'(u)(\boldsymbol{x},t)(W^*\lambda)(\boldsymbol{x},t) \tag{83}$$

on $[0,T]$ with boundary condition $\lambda(x,T)=0$. Alternatively, Eq. (83) can also be obtained from the Euler–Lagrange equations

$$\frac{\delta\mathcal{L}}{\delta u} - \frac{d}{dt}\frac{\delta\mathcal{L}}{\delta\dot{u}} = 0$$

applied to the Lagrange density

$$\mathcal{L}(\boldsymbol{x},t) = \tfrac{1}{2}(u(\boldsymbol{x},t) - v(\boldsymbol{x},t))^2 + \lambda(\boldsymbol{x},t)\{\tau(\boldsymbol{x})\dot{u}(\boldsymbol{x},t) + u(\boldsymbol{x}) - (Wf(u))(\boldsymbol{x},t)\}$$

for $\boldsymbol{x}\in D$, $t\in[0,T]$. Equation (83) is an integrodifferential equation for which the existence theory of Potthast and beim Graben[131] can be applied, i.e., we obtain well-defined global solutions for every admissible choice of a kernel w for W^* and forcing terms u and v.

Backpropagation for neural fields is then obtained as a gradient descent algorithm in function space, where J is minimized under the constraints (59) and (83) with respect to w and τ (and possibly p).[138]

III. METHOD

In this section, we first generalize the classical Hebbian learning rule to the Amari equation using basic results from mathematical analysis. We show that the Hebb rule is imme-

diately entailed by the theory of orthonormal systems and is thus a natural way to solve the inverse problem in dynamic cognitive modeling.

In addition, we illustrate our previous findings by means of three examples for basic cognitive processes. The first example implements the push operation equation (1) in a one-dimensional neural field. The second example implements logical inference of the equivalence relation in Table I by means of traveling pulses with lateral inhibition in a layered neural field. Our third example presents a tree generator processing a simple context-free grammar.

In all three examples we construct synaptic weight kernels for the Amari equation by regularized Hebbian learning such that the neural field dynamics replicate the prescribed training patterns.

A. Tikhonov regularized Hebbian learning

Let us study the neural inverse problem in the form

$$\psi(\boldsymbol{x},t) = \int_D w(\boldsymbol{x},\boldsymbol{y})\varphi(\boldsymbol{y},t)d\boldsymbol{y}, \quad \boldsymbol{x} \in D, \quad t \geq 0, \tag{84}$$

where the auxiliary fields φ and ψ are given by Eqs. (60) and (61), and we search for the kernel w. Given the images $\psi(\boldsymbol{x},t)$ for the elements $\varphi(\boldsymbol{x},t)$ for all $\boldsymbol{x} \in D$, $t \geq 0$ leads to the problem that we know a bounded linear operator W and need to construct a kernel $w(\boldsymbol{x},\boldsymbol{y})$ such that

$$(Wg)(\boldsymbol{x}) = \int_D w(\boldsymbol{x},\boldsymbol{y})g(\boldsymbol{y})d\boldsymbol{y}, \quad \boldsymbol{x} \in D \tag{85}$$

for all $g \in X$ with some appropriate space X. Assume that we know W. Then for any orthonormal basis $\{g_n | n \in \mathbb{N}\}$, we calculate $f_n = Wg_n$. We define the kernel w as

$$w(\boldsymbol{x},\boldsymbol{y}) = \sum_{n=1}^{\infty} f_n(\boldsymbol{x})g_n(\boldsymbol{y}) \tag{86}$$

if the sum is convergent. Then we can calculate

$$\int_D w(\boldsymbol{x},\boldsymbol{y})g_\ell(\boldsymbol{y})d\boldsymbol{y} = \sum_{n=1}^{\infty} f_n(\boldsymbol{x}) \int_D g_n(\boldsymbol{y})g_\ell(\boldsymbol{y})d\boldsymbol{y}$$

$$= \sum_{n=1}^{\infty} f_n(\boldsymbol{x})\delta_{n\ell} = f_\ell(\boldsymbol{x}).$$

If we know that W is an integral operator (85) with kernel w, then we can calculate

$$f_n(\mathbf{x}) = \int_D w(\boldsymbol{x},\boldsymbol{y})g_n(\boldsymbol{y})d\boldsymbol{y}, \quad n \in \mathbb{N}.$$

Further, we can develop $w(\boldsymbol{x},\boldsymbol{y})$ into a Fourier series with respect to the variable $\mathbf{y}$ and the orthonormal basis g_n. We calculate

$$w(\boldsymbol{x},\boldsymbol{y}) = \sum_{n=1}^{\infty} \left(\int_D w(\boldsymbol{x},\boldsymbol{y})g_n(\boldsymbol{y})d\boldsymbol{y} \right) g_n(\boldsymbol{y}) = \sum_{n=1}^{\infty} f_n(\boldsymbol{x})g_n(\boldsymbol{y}). \tag{87}$$

Equations (86) and (87), respectively, provide the desired generalization of the well-known Hebbian learning rule of neural networks for neural fields. This learning rule was originally suggested by Hebb[139] as a psychological mechanism based on physiological arguments. Here, it appears as a basic conclusion from the Fourier theorem for orthonormal systems. The underlying physiological idea is that a synaptic connection between neurons at sites $\boldsymbol{y}$ and $\boldsymbol{x}$, as expressed by the kernel $w(\boldsymbol{x},\boldsymbol{y})$, is given by the cross correlation of their activations, reflecting the input-output pairing φ, ψ. Apparently, this is a natural consequence of the representation of the kernel by orthonormal input patterns φ.

However, training patterns are usually not orthonormal. For a general set of training patterns that are not orthogonal, the Hebb rule does not yield optimal kernels due to cross-talk. Nevertheless, as far as training patterns are still linearly independent, one can cope with cross-talk through biorthogonal bases, which lead, in turn, to the Moore–Penrose pseudoinverse for finite-dimensional neural networks.[6,87,104] Potthast and beim Graben[132] generalized this approach to the field-theoretic setting deploying biorthogonal bases in Hilbert spaces.

If, by contrast, training patterns are neither orthogonal nor linearly independent, the inverse problem for the Amari equation is ill-posed and solutions have to be regularized to stably construct appropriate kernels. Thus, the more general theory of Sec. II C 3 is required for this aim. We propose a modified Tikhonov–Hebb rule and apply it to calculating neural fields for different cognitive dynamics.

Consider a dynamical field defined by Eq. (54) as

$$v(\boldsymbol{x},t) = \sum_{k=1}^{L} \lambda_k(t)v_k(\boldsymbol{x}), \quad \boldsymbol{x} \in D, \quad t \in [0,T].$$

Introducing the auxiliary fields φ, ψ according to Eqs. (60) and (61), again, leads to the integral equation

$$\psi(\boldsymbol{x},t) = \int_D w(\boldsymbol{x},\boldsymbol{y})\varphi(\boldsymbol{y},t)d\boldsymbol{y}, \quad \boldsymbol{x} \in D, \quad t \in [0,T] \tag{88}$$

for the unknown kernel $w(\boldsymbol{x},\boldsymbol{y})$. We discretize the time interval $[0,T]$ by N_t points and step size

$$h_t = \frac{T}{N_t - 1},$$

expressing the Euler rule

$$\frac{du(\boldsymbol{x},t)}{dt} \approx \frac{u(\boldsymbol{x},t+h_t) - u(\boldsymbol{x},t)}{h_t},$$

i.e., the nodes of the regular time grid are given by

$$t_k = (k-1) \cdot h_t, \quad k = 1, \ldots, N_t. \tag{89}$$

On the spatial domain D we use a discretization with N_d points. For the one-dimensional case, we employ a regular grid on $[0,2\pi]$; for the sphere in the second example; we chose a regular grid in polar coordinates. In both cases we write the discretization as x_j, $j=1,\ldots,N_d$. The time-space discretization leads to matrices

$$\boldsymbol{\varphi}_{jk} = f(u(\boldsymbol{x}_j,t_k)), \quad \boldsymbol{x} \in D, \quad t \geq 0, \tag{90}$$

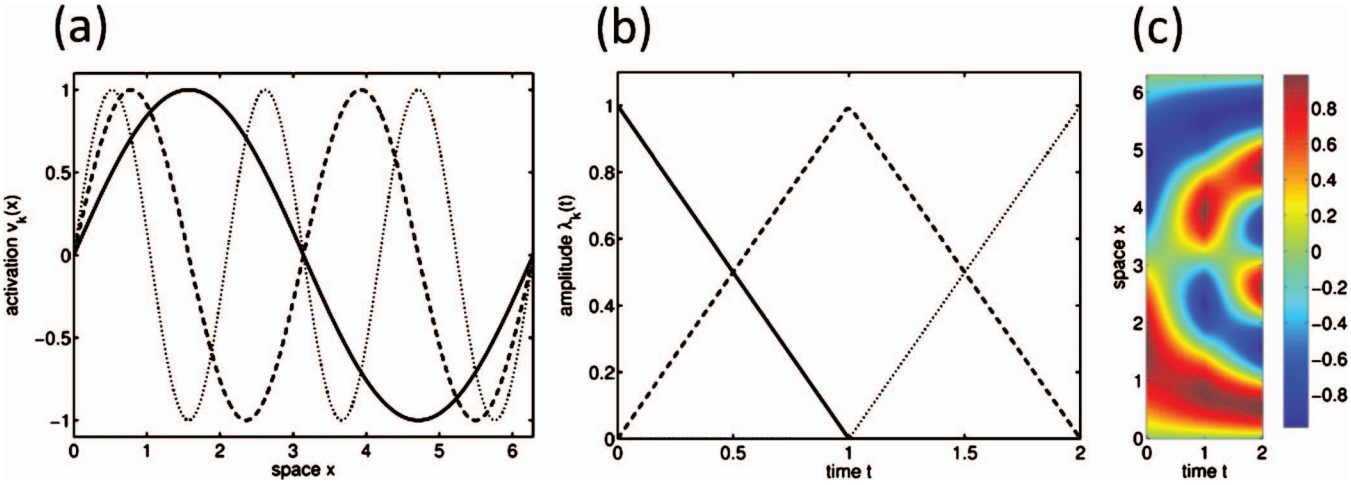

FIG. 8. (Color) Test patterns used in the one-dimensional Amari model. (a) Three "cognitive states" of a pushdown automaton (cf. Sec. II B) represented by spatial waves $v_k(x)=\sin(kx)$, with $k=1$ (solid), $k=2$ (dashed), and $k=3$ (dotted). (b) Their respective amplitudes $\lambda_k(t)$ as tent functions of time. (c) The spatiotemporal training pattern $v(x,t)=\Sigma_k\lambda_k(t)v_k(x)$ resulting from the separation ansatz (53) for solving the inverse problem for the Amari equation (59).

$$\psi_{jk}=\tau\frac{u(x_j,t_k+h_t)-u(x_j,t_k)}{h_t}+u(x_j,t_k),\quad x\in D,\quad t\geq 0. \tag{91}$$

Discretization of the kernel $w(x,y)$ is obtained by using the above spatial grid for both x and y. We will consider the surface element dy to be a part of the discretized kernel, i.e., we approximate the operator W by the matrix

$$W=(w(x_j,y_\ell))_{j,\ell=1,\ldots,N_d}.$$

Then, Eq. (88) is transformed into the matrix equation

$$\psi=W\varphi. \tag{92}$$

With the Tikhonov inverse of φ given as

$$R_\alpha=(\alpha I+\varphi^*\varphi)^{-1}\varphi^*,\quad \alpha>0, \tag{93}$$

we obtain the reconstructed synaptic weight kernel

$$W_\alpha=\psi R_\alpha=\psi(\alpha I+\varphi^*\varphi)^{-1}\varphi^* \tag{94}$$

for the Tikhonov–Hebbian learning rule of the Amari equation (59). Note that Oja's rule for unsupervised Hebbian learning[6] can be considered as a special case of the Tikhonov–Hebb rule by replacing the regularization constraint $\min_g \alpha\|g\|$ in Eq. (78) by $\min_g \alpha(1-\|g\|)$.

B. Pushdown stack

In Eq. (1) we described a stack as a list of filler symbols $F=\{a,b\}$ occupying slots $R=\{r_1,r_2,r_3\}$ for the three list positions. We use the functional representations (35) and (36), thus obtaining the cognitive states

$$\begin{aligned} v_1(x)&=\rho(\alpha_1)=\sin x,\\ v_2(x)&=\rho(\alpha_2)=\sin x+\sin 2x,\\ v_3(x)&=\rho(\alpha_3)=\sin x+\sin 2x+\sin 3x, \end{aligned} \tag{95}$$

as images of the symbolic path Eq. (1) in function space where we have deliberately replaced the variable y by x since the symbol b, represented by the function $f_b(x)=x$, does not occur.

Instead of describing the temporal evolution by the order parameter [Eq. (54)], we simply use tent maps

$$\lambda_k(t)=\begin{cases}\dfrac{t-(k-1)\mu}{\mu}+1, & -k\mu\leq t\leq -(k-1)\mu\\ -\dfrac{t-(k-1)\mu}{\mu}+1, & -(k-1)\mu\leq t\leq (2-k)\mu,\end{cases} \tag{96}$$

where μ indicates the maximum of amplitude $\lambda_k(t)$. For the first example we set $\mu=1$.

Figure 8 displays the spatial patterns $v_k(x)$ given by Eq. (95) in (a), the time course of the amplitudes $\lambda_k(t)$ [Eq. (96)] in (b), and the spatiotemporal dynamics $v(x,t)$ resulting from Eq. (53) in (c).

This prescribed trajectory in function space is trained via Tikhonov–Hebbian learning by a discretized neural field obeying the Amari equation (59) with the following parameters: number of spatial discretization points $N_d=300$, number of temporal discretization points $N_t=100$, time constant $\tau=0.5$, synaptic gain $\beta=10$, and activation threshold $\eta=0.3$, where we compare the Moore–Penrose pseudoinverse (79) (i.e., unregularized, $\alpha=0$) with Tikhonov regularization [Eq. (93), $\alpha=0.1$] for the Hebb rule.

C. Logic gate

One of the most persistent problems in early neural network research was the implementation of linearly nonseparable logical functions, such as xor or equiv (see Sec. II A 1) by means of feed-forward architectures. These problems are not solvable with one-layered networks at all and solving them with the nonlinear backpropagation algorithm for two-layered networks is computationally very expensive (cf. Ref. 6).

Here, we discuss a two-dimensional neural field model where the y-dimension serves as representation space for the

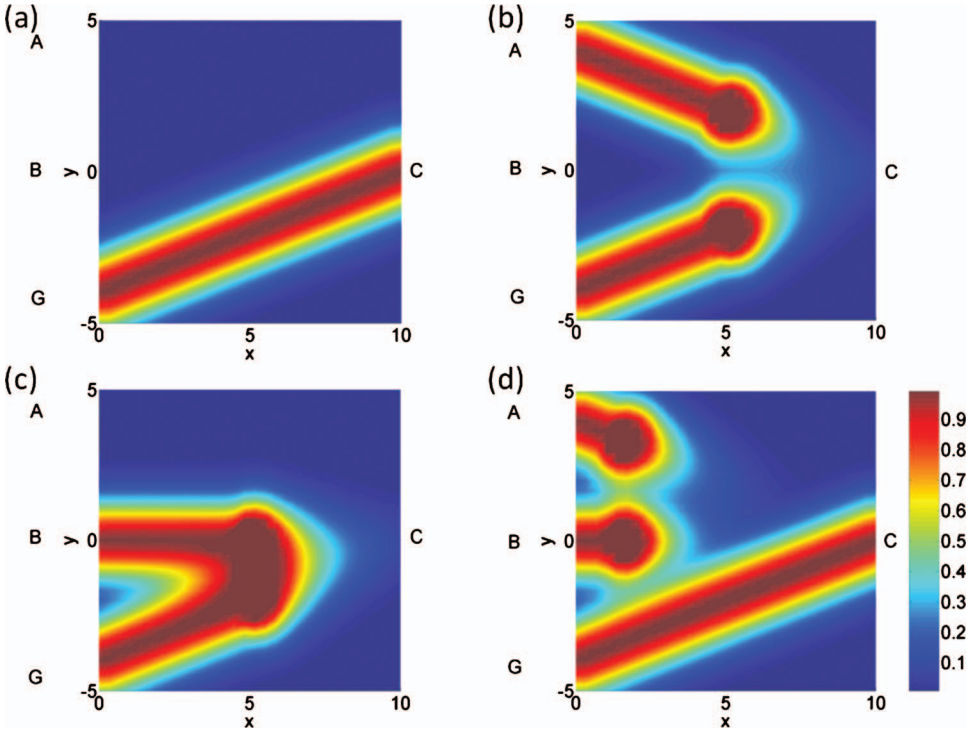

FIG. 9. (Color) Training pulses for logical equivalence function equiv (Table I). (a) $A=0$, $B=0$, and $C=1$; (b) $A=1$, $B=0$, and $C=0$; (c) $A=0$, $B=1$, and $C=0$; and (d) $A=1$, $B=1$, and $C=1$. The gating pulse $G=1$ in all cases.

tensor product representation [Eq. (40)], whereas the x-dimension provides a continuum of (uncountably) infinite many layers. The inputs A and B for the equiv gate are given by two Gaussians centered around points $x_1, x_2 \in \mathbb{R}$. Additionally, a permanent gating input $G=1$ is represented by a third Gaussian centered around $x_3 \in \mathbb{R}$. The inference dynamics is prescribed by Gaussian pulses traveling along linear paths along the x-axis through

$$v(\boldsymbol{x},t) = \lambda_1(t) A e^{-R|\boldsymbol{x}-\boldsymbol{x}_1(t)|^2} + \lambda_2(t) B e^{-R|\boldsymbol{x}-\boldsymbol{x}_2(t)|^2} + \lambda_3(t) G e^{-R|\boldsymbol{x}-\boldsymbol{x}_3(t)|^2} \tag{97}$$

with $A,B \in \{0,1\}$, $G=1$, $\boldsymbol{x}=(x,y) \in D \subset \mathbb{R}^2$, and $\boldsymbol{x}_i(t) = (0,y_i) + t\boldsymbol{d}_i$ for $t \geq 0$. Here, $\boldsymbol{d}_i$ denotes the direction of the ith neural pulse.

The amplitudes $\lambda_i(t)$ obey a suitably chosen order parameter equation (53) that reflects lateral inhibition among the three pulses. Four trajectories of neural pulses were explicitly constructed, as illustrated in Fig. 9.

When $A=B=0$ [Fig. 9(a)], there is no inhibition at all and the gating pulse G travels into its destination, yielding output $C=1$. For $A=1$, $B=0$ [Fig. 9(b)], input A interferes with G around $x=5$, leading to extinction of G. Likewise, for $A=0$, $B=1$ [Fig. 9(c)], B and G annihilate each other around $x=5$, again. Finally, if $A=B=1$ [Fig. 9(d)], lateral inhibition of A and B takes place much earlier around $x=2$ such that the unaffected gating pulse G becomes the desired output $C=1$.

The four different fields of superimposed traveling pulses are used as training patterns for Tikhonov–Hebbian learning of synaptic weight kernels, as explained above. We used a discretization of $N_d=30\times31$ points in the spatial domain D and $N_t=80$ temporal discretization points for solving the inverse problem. As regularization parameter we set $\alpha=1$. Simulations were carried out with $N_t=160$ time discretization points and $\tau=2$, $\beta=10$, and $\eta=0.5$.

D. Tree generator

Our second example is related to syntactic language processing where context-free grammars are processed by pushdown automata. For the sake of simplicity, we describe the process of tree generation as used in top-down parsing approaches.[13,14,37] Here we use the CFG

$$\mathbf{T}=\{\mathrm{a}\}, \quad \mathbf{N}=\{\mathrm{S}\}, \quad P=\left\{\begin{array}{l}(1)\ \ \mathrm{S}\rightarrow \mathrm{S}\ \ \mathrm{S}\\ (2)\ \ \mathrm{S}\rightarrow \mathrm{a}\end{array}\right\}, \tag{98}$$

where a is the only terminal and the start symbol S is the only nonterminal symbol. One particular path of a tree generator is shown in Fig. 10.

The trees depicted in Fig. 10 are functionally represented using the following mappings:

$$\rho(\mathrm{a}) = f_{\mathrm{a}}(x) = 0, \quad \rho(\mathrm{S}) = f_{\mathrm{S}}(x) = 1 \tag{99}$$

for the fillers. Note that a is treated as the "empty word" represented by the zero function here. The tree roles are represented by spherical harmonics as outlined in Sec. IV. Using the tensor product representation deploying Clebsch–Gordan coefficients, again, yields static spatial patterns

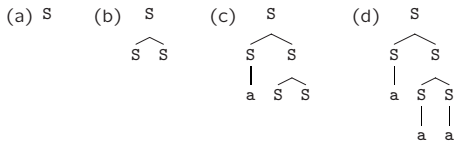

FIG. 10. Example-tree generation according to grammar (98).

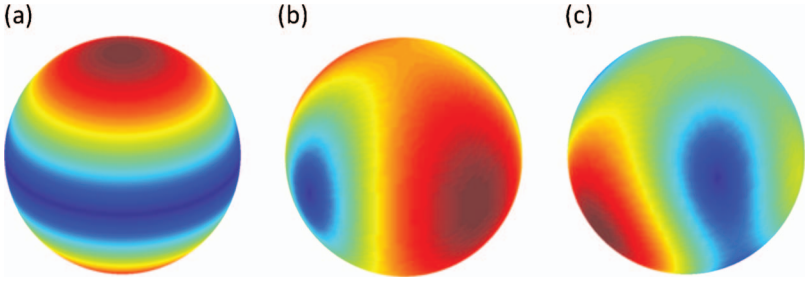

FIG. 11. (Color) Spherical harmonics representation of trees $v_k(x)$ [Eq. (100)] used in the two-dimensional Amari model (cf. Sec. II B). (a) Zero-level tree consisting only of root node S as in Fig. 10(a), (b) one-level tree as in Fig. 10(b), and (c) two-level tree as in Fig. 10(c). [Note that trees from Figs. 10(c) and 10(d) are mapped onto the same spherical representation].

$$v_1(x) = |Y_{1,0}(x)|, \quad v_2(x) = |Y_{1,0}(x) + Y_{1,-1}(x) + Y_{1,1}(x)|, \tag{100}$$

$$v_3(x) = \left| Y_{1,0}(x) + Y_{1,-1}(x) + \frac{1}{\sqrt{2}}(Y_{2,1}(x) - Y_{1,1}(x)) + \frac{1}{\sqrt{3}}Y_{0,0}(x) - \frac{1}{\sqrt{2}}Y_{1,0}(x) + \frac{1}{\sqrt{6}}Y_{2,0} + Y_{2,2}(x) \right|,$$

which are displayed in Fig. 11.

Figure 11(a) shows the representation $v_1(x)$ of the simple tree from Fig. 10(a) containing only the filler S bound to the root role. Note that this is actually a p_z orbital of the electronic wave function in the hydrogen atom. Figure 11(b) shows $v_2(x)$ as a superposition of three spherical harmonics corresponding to Fig. 10(b). Accordingly, Fig. 11(c) displays the representation of the tree in Fig. 10(c) where the right branch has been further expanded.

Finally, we chose the same tent map [Eq. (96)] (although with different time scales: $\mu=15$) as in Sec. III B for the amplitude dynamics. The spatiotemporal patterns resulting from Eq. (53) are shown in Fig. 12.

Again, a discretized neural field characterized by the Amari equation (59) is trained with the Tikhonov–Hebb rule [Eq. (93)]. Here, we use simulation parameters: number of spatial discretization points $N_d=6400$, number of temporal discretization points $N_t=30$, time constant $\tau=0.5$, synaptic gain $\beta=10$, and activation threshold $\eta=0.3$.

IV. RESULTS

We next describe the results when applying the techniques of Sec. II C 3 to construct synaptic weight kernels generating the prescribed cognitive dynamics in representation space, constructed above.

A. Pushdown stack

As shown in Fig. 8(c), the dynamics of a pushdown stack constructed in Eqs. (95) and (96), appears as a one-dimensional wave field (y-axis) over time (x-axis), realizing continuous transitions from state v_1 to state v_2 and from v_2 to state v_3.

After discretizing temporal and spatial dimensions in accordance to Eqs. (89)–(91), Fig. 13 displays the mapping of the auxiliary fields (a) φ onto (b) ψ. This mapping is achieved by the linear integral transformation [Eq. (92)] derived from the Amari equation (59).

We used Tikhonov–Hebbian inversion theory to construct a synaptic weight kernel w by Eq. (94). Without regularization we do not expect to obtain a reasonably stable kernel. This is demonstrated in Fig. 14(a), where we carried out the reconstruction with the standard Moore–Penrose pseudoinverse (79). Clearly, the classically reconstructed kernel is strongly fluctuating between -10^4 and $+10^4$, thus reflecting the ill-posedness of the inverse problem. On the other hand, the regularized kernel is visualized in Fig. 14(b). Regularization does neither require orthogonality nor even linear independence of training patterns. It is rather robust against linear dependence as resulting, e.g., from oversampling of discretized data. Note further, that the kernel depicted in Fig. 14(b) is not translation invariant. Hence we have explicitly constructed an inhomogeneous synaptic weight kernel from the prescribed cognitive training process.

In the final step we then solved the Amari equation numerically with the reconstructed kernel. Here, it is of considerable importance to use a different discretization in time in order to avoid the inverse crime: using the same sampling for training and simulation could spuriously yield coincident patterns. Therefore, we doubled the temporal sampling rate for simulation.

Simulation results are presented in Fig. 15. Image (a) shows the field generated with the Moore–Penrose pseudoinverse. The field quickly explodes and moves into high-order oscillations, which correspond to the strong amplification of high modes excited by small numerical errors [cf. Eq. (69)]. Figure 15(b) demonstrates the reconstruction of the prescribed field dynamics. This image has been generated from the Tikhonov–Hebbian regularized kernel.

The results prove that stable and quick construction of synaptic weight kernels is possible to generate prescribed dynamics constructed from representations of cognitive states.

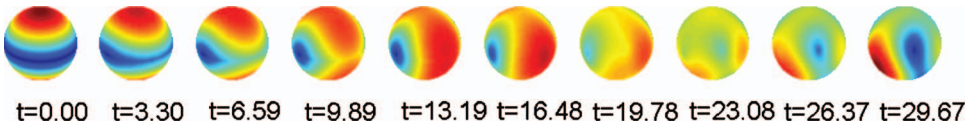

FIG. 12. (Color) Tree generation dynamics resulting from the separation ansatz (53) for solving the inverse problem for the Amari equation (59).

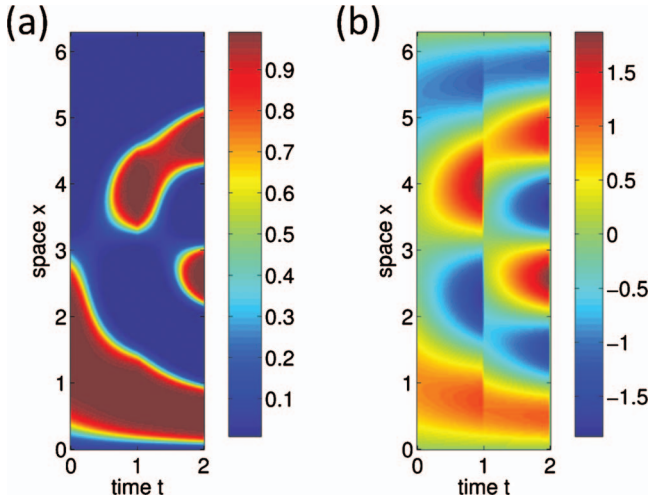

FIG. 13. (Color) Auxiliary fields (a) $\varphi(\mathbf{x},t)$ and (b) $\psi(\mathbf{x},t)$ obtained from the training pattern $v(\mathbf{x},t)$ by Eqs. (60) and (61) for parameters $\tau=0.5$, $\beta=10$, and $\eta=0.3$.

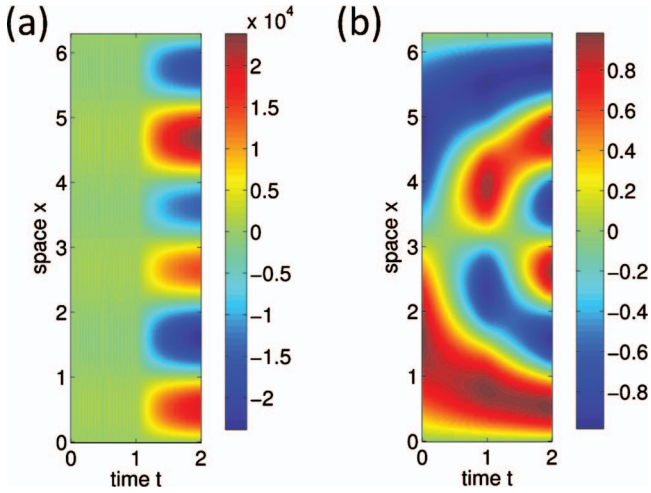

FIG. 15. (Color) Numerical solutions $u(\boldsymbol{x},t)$ of the Amari equation (59) with estimated synaptic weight kernels $w(\boldsymbol{x},\boldsymbol{y})$ as in Fig. 14. (a) For Moore–Penrose pseudoinverse (i.e., $\alpha=0$) and (b) for Tikhonov inverse (93) with regularization parameter $\alpha=0.1$ for parameters $\tau=0.5$, $\beta=10$, and $\eta=0.3$. Compare with Fig. 8.

B. Logic gate

Figure 16 shows the reconstructed traveling pulse dynamics of the logical interference neural field. Clearly, they are in good agreement with the prescribed training data shown in Fig. 9.

Our results show that a hardly tractable problem for connectionist neural networks becomes linearly solvable for neural fields.

C. Tree generator

As before, Fig. 12 visualizes the prescribed cognitive dynamics constructed in Eq. (100). It shows a selection of time slices for the spatiotemporal dynamics of the field representations, sampling a realization of a continuous transition over several states $v_1,\ldots,v_n$. Again, we discretized both the temporal and spatial dimensions following Eqs. (89)–(91).

We then used Tikhonov–Hebbian inversion theory again to construct a synaptic weight kernel w. In turn, this kernel was used to calculate the neural field $u(\boldsymbol{x},t)$ as a solution of the Amari equation (59), which is shown in Fig. 17.

The results demonstrate that stable and quick construction of synaptic weight kernels is feasible over another two-dimensional manifold in order to replicate neural field dynamics in cognitive representation spaces.

V. DISCUSSION

We described dynamic cognitive modeling as a three tier top-down approach comprising the levels of (1) cognitive processes, (2) state space representations, and (3) neurodynamical implementations. These levels are passed through in a top-down manner: (1) cognitive processes are described as algorithms sequentially operating on complex symbolic data structures that are decomposed using filler/role bindings; (2) data structures are mapped onto "points" in Fock spaces (abstract feature spaces) using tensor product representations; and (3) cognitive operations are implemented as dynamics of neural respective dynamic fields as continuum approximations of neural networks. The last step involves the solution of inverse problems, namely, training synaptic weights of the Amari equation to reproduce prescribed paths in representation space.

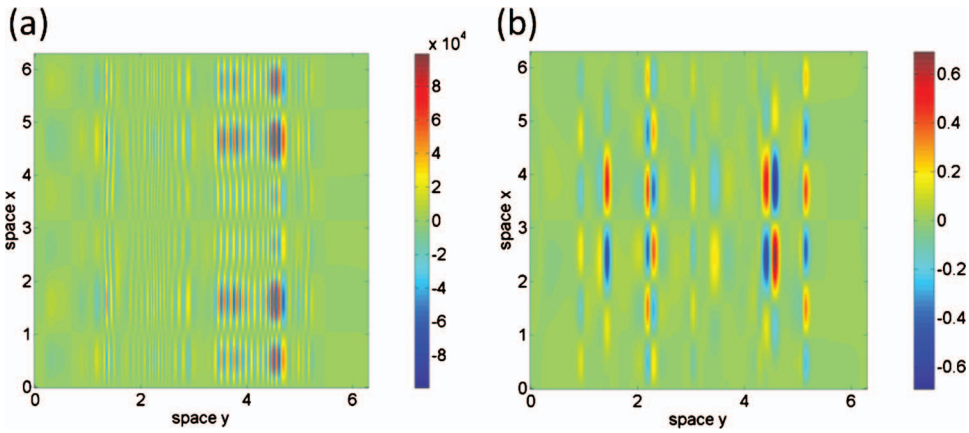

FIG. 14. (Color) Synaptic weight kernels $w(\boldsymbol{x},\boldsymbol{y})$ of the Amari equation (59). (a) For Moore–Penrose pseudoinverse (i.e., $\alpha=0$) and (b) for Tikhonov inverse (93) with regularization parameter $\alpha=0.1$ for parameters $\tau=0.5$, $\beta=10$, and $\eta=0.3$.

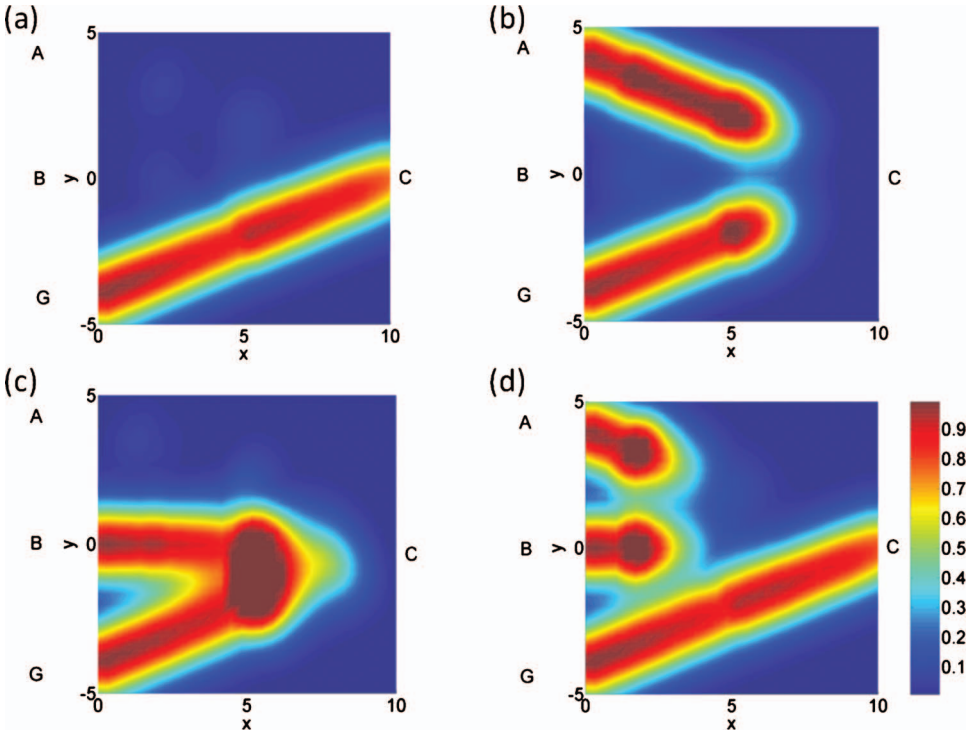

FIG. 16. (Color) Numerical solutions $u(x,t)$ of the Amari equation (59) for logical equivalence function equiv (Table I). (a) $A=0$, $B=0$, and $C=1$; (b) $A=1$, $B=0$, and $C=0$; (c) $A=0$, $B=1$, and $C=0$; and (d) $A=1$, $B=1$, and $C=1$ for parameters $\alpha=1$, $\tau=2$, $\beta=10$, and $\eta=0.5$. Compare with Fig. 9.

After recasting the Amari equation into a linear integral equation that can principally be trained by Hebbian learning, we demonstrated that these problems are ill-posed and solutions have to be regularized. We suggested a modified Tikhonov–Hebb learning rule and showed its stability for particular examples of cognitive dynamics in function spaces. Tikhonov–Hebbian learning is a quick and simple (for being linear) training algorithm, not requiring orthogonality or even linear independence of training patterns. In fact, the regularization is robust against linearly dependent patterns as they could result from oversampling. Regarding neural field models with traveling pulse dynamics as layered architectures,[132] showed that Tikhonov–Hebbian learning also works for linearly nonseparable problems such as the persistent XOR problem for which nonlinear and computationally expensive training algorithms have to be employed for connectionist models.

Moreover, Tikhonov–Hebbian learning generally leads, for given training patterns, to inhomogeneous kernels, which is a considerable progress for neural/dynamic field models of cognitive processes. Most efforts in the field so far have been done investigating either homogeneous kernels[21,24,25,28,29] or by introducing inhomogeneity explicitly.[20,31,122,140–143]

For the sake of simplicity, we represented cognitive states as static spatial patterns in one and two dimensions by encoding symbolic fillers as constants. Using functional representations for fillers as well would further increase the dimensionality of variable domains by one. By contrast, Ref. 14 suggested a separation of time scales to represent fillers as fast oscillations and cognitive processes as slow transients independently. This approach, however, is not feasible using Tikhonov–Hebbian learning for solving the inverse problem for the Amari equation because a synaptic weight kernel, trained on the fast time scale, would not be able to replicate the slow cognitive dynamics and vice versa. As the neural field is a deterministic dynamical system, the same fast cycle would be repeated after training the first one. Nevertheless, separation of time scales could still be possible using more sophisticated solutions of the inverse problem involving time-dependent input to the neural field. We outlined one possible solution of the nonlinear inverse problem by functional backpropagation for learning weight kernels, time constants and possibly input forces simultaneously.

In the present state, dynamic cognitive modeling deals with an abstract, theoretical task, namely, solving the inverse problem of finding a neurodynamical implementation for a given algorithmic symbol processor in a top-down fashion. It does currently not address another inverse problem prevalent in cognitive neuroscience, namely the bottom-up reconstruction of neurodynamics from observed physiological time series, such as electroencephalographic (EEG),[144–149] MEG,[150] optical diffusion tomographic,[151] or functional magnetic resonance tomographic[152] data. In particular, it is not related to dynamic causal modeling[153,154] or similar approaches[155,156] where the inverse problem of finding neural generators from physiological data is addressed. Certainly, dynamic causal modeling and dynamic cognitive modeling ought to be connected at the intermediate level of neurody-

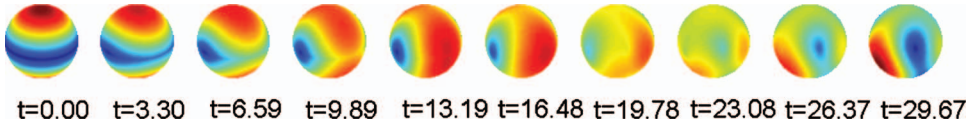

FIG. 17. (Color) Numerical solutions $u(x,t)$ of the Amari equation (59) for tree generation dynamics. Compare with Fig. 12.

namics in the long run, one coming from the bottom level of physiological data, the other one coming from the top level of cognitive architecture. We leave this ambitious project for future research.

ACKNOWLEDGMENTS

This work has been supported by an EPSCR Bridging the Gaps grant on *Cognitive Systems Sciences*. Inspiring discussions with and helpful comments from Sabrina Gerth, Whitney Tabor, Paul Smolensky, and Johannes Haack are gratefully acknowledged. We would like to thank Tito Arecchi and Jürgen Kurths for their kind invitation to contribute to this focus issue on *Nonlinear Dynamics in Cognitive and Neural Systems*.

[1] J. A. S. Kelso, *Dynamic Patterns: The Self-Organization of Brain and Behavior* (MIT Press, Cambridge, MA, 1995).
[2] T. van Gelder, Behav. Brain Sci. **21**, 615 (1998).
[3] R. D. Beer, Trends Cogn. Sci. **4**, 91 (2000).
[4] P. beim Graben, Mind and Matter **2**, 29 (2004).
[5] *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, edited by D. E. Rumelhart, J. L. McClelland, and the PDP Research Group (MIT Press, Cambridge, MA, 1986), Vol. 1.
[6] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation* (Perseus Books, Cambridge, MA, 1991).
[7] D. Marr and T. Poggio, Massachusetts Institute of Technology Technical Report No. AIM-357, 2003.
[8] Z. W. Pylyshyn, *Computation and Cognition: Toward a Foundation for Cognitive Science* (MIT Press, Cambridge, MA, 1986).
[9] C. P. Dolan and P. Smolensky, Connect. Sci. **1**, 53 (1989).
[10] P. Smolensky, Artif. Intell. **46**, 159 (1990).
[11] P. Smolensky and G. Legendre, *The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar*,(MIT Press, Cambridge, MA, 2006), Vol. 1.
[12] P. Smolensky, Cogn. Sci. **30**, 779 (2006).
[13] P. beim Graben, S. Gerth, and S. Vasishth, Cognit. Neurodynamics **2**, 229 (2008).
[14] P. beim Graben, D. Pinotsis, D. Saddy, and R. Potthast, Cognit. Neurodynamics **2**, 79 (2008).
[15] M. J. Spivey and R. Dale, Psychol. Learn. Motiv. **45**, 87 (2004).
[16] B. L. van der Waerden, *Algebra* (Springer, New York, 2003), Vol. 2.
[17] R. Haag, *Local Quantum Physics: Fields, Particles, Algebras* (Springer, Berlin, 1992).
[18] S.-I. Amari, Biol. Cybern. **27**, 77 (1977).
[19] *Lectures in Supercomputational Neuroscience: Dynamics in Complex Brain Networks*, edited by P. beim Graben, C. Zhou, M. Thiel, and J. Kurths (Springer, Berlin, 2008), pp. 3–48.
[20] P. C. Bressloff and J. D. Cowan, Physica D **173**, 226 (2002).
[21] S. Coombes, G. Lord, and M. Owen, Physica D **178**, 219 (2003).
[22] G. B. Ermentrout and J. B. McLeod, Proc. - R. Soc. Edinburgh, Sect. A: Math **123A**, 461 (1993).
[23] J. S. Griffith, Bull. Math. Biophys. **25**, 111 (1963).
[24] A. Hutt and F. M. Atay, Physica D **203**, 30 (2005).
[25] V. K. Jirsa and H. Haken, Phys. Rev. Lett. **77**, 960 (1996).
[26] P. L. Nunez, Behav. Brain Sci. **23**, 371 (2000).
[27] K. A. Richardson, S. J. Schiff, and B. J. Gluckman, Phys. Rev. Lett. **94**, 028103 (2005).
[28] P. A. Robinson, C. J. Rennie, and J. J. Wright, Phys. Rev. E **56**, 826 (1997).
[29] N. Venkov, S. Coombes, and P. Matthews, Physica D **232**, 1 (2007).
[30] H. R. Wilson and J. D. Cowan, Kybernetik **13**, 55 (1973).
[31] J. J. Wright, C. J. Rennie, G. J. Lees, P. A. Robinson, P. D. Bourke, C. L. Chapman, E. Gordon, and D. L. Rowe, Neuropsychopharmacology **28**, S80 (2003).
[32] W. Erlhagen and G. Schöner, Psychol. Rev. **109**, 545 (2002).
[33] G. Schöner and E. Thelen, Psychol. Rev. **113**, 273 (2006).
[34] E. Thelen, G. Schöner, C. Scheier, and L. B. Smith, Behav. Brain Sci. **24**, 1 (2001).
[35] A. Newell and H. A. Simon, Commun. ACM **19**, 113 (1976).
[36] J. Fodor and Z. W. Pylyshyn, Cognition **28**, 3 (1988).
[37] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation* (Addison-Wesley, Menlo Park, California, 1979).
[38] J. R. Anderson, *Cognitive Psychology and its Implications* (W. H. Freeman, New York, NY, 1995).
[39] R. L. Lewis, *Encyclopedia of Cognitive Science* (Macmi llan, London, 2003).
[40] A. K. Joshi and Y. Schabes, in *Handbook of Formal Languages and Automata*, edited by A. Salomma and G. Rosenberg (Springer, Berlin, 1997), Vol. 3, pp. 69–124.
[41] E. P. Stabler, in *Logical Aspects of Comutational Linguistics*, edited by C. Retoré (Springer, New York, 1997), pp. 68–95.
[42] J. Michaelis and C. Wartena, in *Constraints and Resources in Natural Language Syntax and Semantics*, edited by G. Bouma, G.-J. Kruijff, E. Hinrichs, and R. T. Oehrle (CSLI, Stanford, CA, 1999), pp. 263–279.
[43] E. P. Stabler, Cogn. Sci. **28**, 699 (2004).
[44] J. Michaelis, in *Logical Aspects of Computational Linguistics*, edited by M. Moortgat (Springer, Berlin, 2001), pp. 179–198.
[45] H. T. Siegelmann, Theor. Comput. Sci. **168**, 461 (1996).
[46] C. Moore, Phys. Rev. Lett. **64**, 2354 (1990).
[47] C. Moore, Nonlinearity **4**, 199 (1991).
[48] R. Badii and A. Politi, *Complexity: Hierarchical Structures and Scaling in Physics* (Cambridge University Press, Cambridge, 1997).
[49] E. Mizraji, Bull. Math. Biol. **51**, 195 (1989).
[50] E. Mizraji, Fuzzy Sets Syst. **50**, 179 (1992).
[51] H. T. Siegelmann and E. D. Sontag, J. Comput. Syst. Sci. **50**, 132 (1995).
[52] W. Tabor, Technical Report No. TR98-1694, 1998.
[53] W. Tabor, Expert Sys. **17**, 41 (2000).
[54] W. Tabor, University of Connecticut Report, 2002.
[55] J. B. Pollack, Mach. Learn. **7**, 227 (1991).
[56] C. Moore, Theor. Comput. Sci. **201**, 99 (1998).
[57] C. Moore and J. P. Crutchfield, Theor. Comput. Sci. **237**, 275 (2000).
[58] P. beim Graben, B. Jurish, D. Saddy, and S. Frisch, Int. J. Bifurcation Chaos Appl. Sci. Eng. **14**, 599 (2004).
[59] P. beim Graben, in *Advances in Cognitive Neurodynamics*, Proceedings of the International Conference on Cognitive Neurodynamics, ICCN 2007, edited by R. Wang, F. Gu, and E. Shen (Springer, Berlin, 2008), pp. 469–473.
[60] P. Cvitanović, G. H. Gunaratne, and I. Procaccia, Phys. Rev. A **38**, 1503 (1988).
[61] M. B. Kennel and M. Buhl, Phys. Rev. Lett. **91**, 084102 (2003).
[62] A. Maye and M. Werning, Neurocomputing **58–60**, 941 (2004).
[63] M. Werning, Synthese **146**, 203 (2005).
[64] A. Maye and M. Werning, Chaos Complexity Lett. **2**, 315 (2007).
[65] M. Werning and A. Maye, Chaos Complexity Lett. **2**, 435 (2007).
[66] A. R. Edmonds, *Angular Momentum in Quantum Mechanics* (Princeton University Press, New Jersey, 1957).
[67] D. Gernert, BioSystems **54**, 165 (2000).
[68] H. Atmanspacher and T. Filk, BioSystems **85**, 84 (2006).
[69] P. beim Graben, Mind and Matter **4**, 169 (2006).
[70] H. Haken, *Synergetics: An Introduction* (Springer, Berlin, 1983).
[71] V. S. Afraimovich, V. P. Zhigulin, and M. I. Rabinovich, Chaos **14**, 1123 (2004).
[72] M. I. Rabinovich, R. Huerta, P. Varona, and V. S. Afraimovich, PLOS Comput. Biol. **4**, e1000072 (2008).
[73] M. Rabinovich, A. Volkovskii, P. Lecanda, R. Huerta, H. D. I. Abarbanel, and G. Laurent, Phys. Rev. Lett. **87**, 068102 (2001).
[74] R. Huerta and M. Rabinovich, Phys. Rev. Lett. **93**, 238104 (2004).
[75] P. Smolensky, Behav. Brain Sci. **11**, 1 (1988).
[76] R. Dale and M. J. Spivey, J. Exp. Theor. Artif. Intell. **17**, 317 (2005).
[77] P. beim Graben and H. Atmanspacher, Found. Phys. **36**, 291 (2006).
[78] M. J. Spivey and S. E. Anderson, J. Exp. Theor. Artif. Intell. **20**, 239 (2008).
[79] P. Smolensky, in *Meaning in Mind: Fodor and His Critics*, edited by B. Loewer and G. Rey (Blackwell, Oxford, 1991), pp. 201–227.
[80] J. Fodor and B. P. McLaughlin, Cognition **35**, 183 (1990).
[81] D. J. Chalmers, in Proceedings of the 12th Annual Conference on Cognitive Science Society, 1990 (unpublished), pp. 340–347.
[82] J. Fodor, Cognition **62**, 109 (1997).
[83] A. Prince and P. Smolensky, Science **275**, 1604 (1997).
[84] R. B. Stein, K. V. Leung, D. Mangeron, and M. N. Oğuztöreli, Kybernetik **15**, 1 (1974).
[85] H. R. Wilson and J. D. Cowan, Biophys. J. **12**, 1 (1972).
[86] A. Arenas, A. Díaz-Guilera, J. Kurths, Y. Moreno, and C. Zhou, Phys. Rep. **469**, 93 (2008).

[87]*Lectures in Supercomputational Neuroscience: Dynamics in Complex Brain Networks*, edited by P. beim Graben, C. Zhou, M. Thiel, and J. Kurths (Springer, Berlin, 2008), pp. 195–223.
[88]P. beim Graben and J. Kurths, Neurocomputing **71**, 999 (2008).
[89]*Neurocomputing: Foundations of Research*, edited by J. A. Anderson and E. Rosenfeld (MIT Press, Cambridge, MA, 1988), Vol. 1.
[90]*Neurocomputing: Directions for Research*, edited by J. A. Anderson, A. Pellionisz, and E. Rosenfeld (MIT Press, Cambridge, MA, 1990), Vol. 2.
[91]P. S. Churchland and T. J. Sejnowski, *The Computational Brain* (MIT Press, Cambridge, MA, 1994).
[92]*The Handbook of Brain Theory and Neural Networks*, edited by M. A. Arbib (MIT Press, Cambridge, MA, 1998).
[93]C.-H. Chen and V. Honavar, IEEE Trans. Neural Netw. **10**, 1239 (1999).
[94]W. Tabor, IEEE Trans. Neural Netw. **14**, 444 (2003).
[95]J. L. Elman, in *Mind as Motion: Explorations in the Dynamics of Cognition*, edited by R. F. Port and T. van Gelder (MIT Press, Cambridge, MA, 1995), pp. 195–223.
[96]W. Tabor, C. Juliano, and M. K. Tanenhaus, Lang. Cognit. Processes **12**, 211 (1997).
[97]W. Tabor and M. K. Tanenhaus, Cogn. Sci. **23**, 491 (1999).
[98]S. Lawrence, C. L. Giles, and S. Fong, IEEE Trans. Knowl. Data Eng. **12**, 126 (2000).
[99]I. Farkas and M. W. Crocker, Neurocomputing **71**, 1172 (2008).
[100]A. H. Kawamoto, J. Mem. Lang. **32**, 474 (1993).
[101]J. J. Hopfield, Proc. Natl. Acad. Sci. U.S.A. **79**, 2554 (1982).
[102]P. Smolensky, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, edited by D. E. Rumelhart, J. L. McClelland, and the PDP Research Group (MIT Press, Cambridge, MA, 1986), Vol. 1, pp. 194–281.
[103]H. Haken, *Synergetic Computers and Cognition: A Top-Down Approach to Neural Nets* (Springer, Berlin, 1991).
[104]H. Haken, *Principles of Brain Functioning* (Springer, Berlin, 1996).
[105]T. Vosse and G. Kempen, Cognition **75**, 105 (2000).
[106]C. Balkenius and P. Gärdenfors, in *Principles of Knowledge Representation and Reasoning*, edited by J. A. Allan, R. Fikes, and E. Sandewall (Morgan Kaufmann, San Mateo, CA, 1991), pp. 32–39.
[107]P. Gärdenfors, *Foundations of Computation*, edited by J. v. Eijck and A. Visser (MIT-Press, Cambridge, MA, 1994), pp. 49–77.
[108]R. Blutner, Synthese **141** (2004).
[109]E. Mizraji and J. Lin, Int. J. Bifurcation Chaos Appl. Sci. Eng. **11**, 155 (2001).
[110]M. H. Christiansen and N. Chater, Cogn. Sci. **23**, 417 (1999).
[111]S.-I. Amari, Kybernetik **14**, 201 (1974).
[112]R. Albert and A.-L. Barabási, Rev. Mod. Phys. **74**, 47 (2002).
[113]L. Zemanová, C. Zhou, and J. Kurths, Physica D **224**, 202 (2006).
[114]C. Zhou, L. Zemanová, G. Zamora, C. C. Hilgetag, and J. Kurths, Phys. Rev. Lett. **97**, 238103 (2006).
[115]M. Kaiser and C. C. Hilgetag, Neurocomputing **58–60**, 297 (2004).
[116]O. Sporns, D. R. Chialvo, M. Kaiser, and C. C. Hilgetag, Trends Cogn. Sci. **8**, 418 (2004).
[117]W. Maass, T. Natschläger, and H. Markram, Neural Comput. **14**, 2531 (2002).
[118]V. K. Jirsa and H. Haken, Physica D **99**, 503 (1997).
[119]V. K. Jirsa, A. Fuchs, and J. A. S. Kelso, Neural Comput. **10**, 2019 (1998).
[120]P. C. Bressloff, J. D. Cowan, M. Golubitsky, P. J. Thomas, and M. C. Wiener, Philos. Trans. R. Soc. London, Ser. B **356**, 299 (2001).
[121]P. C. Bressloff, J. D. Cowan, M. Golubitsky, P. J. Thomas, and M. C. Wiener, Neural Comput. **14**, 473 (2002).
[122]P. C. Bressloff, Phys. Rev. Lett. **89**, 088101 (2002).
[123]P. C. Bressloff and J. D. Cowan, J. Physiol. (Paris) **97**, 221 (2003).
[124]P. C. Bressloff, Biol. Cybern. **93**, 256 (2005).
[125]A. M. Oster and P. C. Bressloff, Bull. Math. Biol. **68**, 73 (2006).
[126]R. Kress, *Linear Integral Equations* (Springer-Verlag, Berlin, 1989).
[127]D. Colton and R. Kress, *Inverse Acoustic and Electromagnetic Scattering Theory* (Springer-Verlag, Berlin, 1998).
[128]H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of Inverse Problems* (Kluwer Academic, Dordrecht, 1996).
[129]A. Neumaier, SIAM Rev. **40**, 636 (1998).
[130]R. Kress, *Numerical Analysis, Graduate Texts in Mathematics* (Springer-Verlag, New York, 1999).
[131]R. Potthast and P. beim Graben, "Existence and properties of solutions for neural field equations," Math. Models Meth. Appl. Sci. (to be published).
[132]R. Potthast and P. beim Graben, "Inverse problems in neural field theory," SIAM J. Appl. Dyn. Syst. (unpublished).
[133]H. Bersini, M. Saerens, and L. G. Sotelino, IEEE Trans. Neural Netw. **5**, 945 (1994).
[134]B. A. Pearlmutter, Neural Comput. **1**, 263 (1989).
[135]B. A. Pearlmutter, IEEE Trans. Neural Netw. **6**, 1212 (1995).
[136]L. G. Sotelino, M. Saerens, and H. Bersini, Neural Netw. **7**, 767 (1994).
[137]G.-Z. Sun, H.-H. Chen, and Y.-C. Le, Proceedings International Joint Conference on Neural Networks (IJCNN 91), 1991 (unpublished), Vol. 2, pp. 13–18.
[138]C. Igel, W. Erlhagen, and D. Jancke, Neurocomputing **36**, 225 (2001).
[139]D. O. Hebb, *The Organization of Behavior* (Wiley, New York, NY, 1949).
[140]M. Breakspear, J. A. Roberts, J. R. Terry, S. Rodrigues, N. Mahant, and P. A. Robinson, Cereb. Cortex **16**, 1296 (2006).
[141]Z. P. Kilpatrick, S. E. Folias, and P. C. Bressloff, SIAM J. Appl. Dyn. Syst. **7**, 161 (2008).
[142]V. K. Jirsa and J. A. S. Kelso, Phys. Rev. E **62**, 8462 (2000).
[143]C. J. Rennie, P. A. Robinson, and J. J. Wright, Biol. Cybern. **86**, 457 (2002).
[144]D. Lehmann, Electroencephalogr. Clin. Neurophysiol. **31**, 439 (1971).
[145]D. Lehmann, H. Ozaki, and I. Pal, Electroencephalogr. Clin. Neurophysiol. **67**, 271 (1987).
[146]R. Friedrich, A. Fuchs, and H. Haken, in *Rhythms in Physiological Systems*, edited by H. Haken and H. P. Koepchen (Springer, Berlin, 1991), pp. 315–338.
[147]C. Allefeld and J. Kurths, Int. J. Bifurcation Chaos Appl. Sci. Eng. **14**, 417 (2004).
[148]M. Paluš, I. Dvořák, and I. David, Physica A **185**, 433 (1992).
[149]P. beim Graben, S. Frisch, A. Fink, D. Saddy, and J. Kurths, Phys. Rev. E **72**, 051916 (2005).
[150]V. K. Jirsa, R. Friedrich, and H. Haken, Physica D **89**, 100 (1995).
[151]H. Obrig and A. Villringer, J. Cereb. Blood Flow Metab. **23**, 1 (2003).
[152]K. J. Friston, P. Fletcher, O. Josephs, A. Holmes, M. D. Rugg, and R. Turner, Neuroimage **7**, 30 (1998).
[153]O. David and K. J. Friston, Neuroimage **20**, 1743 (2003).
[154]O. David, S. J. Kiebel, L. Harrison, J. Mattout, J. Kilner, and K. J. Friston, Neuroimage **30**, 1255 (2006).
[155]A. Galka, O. Yamashita, T. Ozaki, R. Biscay, and P. Valdeś-Sosa, Neuroimage **23**, 435 (2004).
[156]A. Galka, T. Ozaki, H. Muhle, U. Stephani, and M. Siniatchkin, Cognit. Neurodynamics **2**, 101 (2008).