International Journal of Bifurcation and Chaos, Vol. 14, No. 2 (2004) 599–621

LANGUAGE PROCESSING BY DYNAMICAL SYSTEMS

PETER BEIM GRABEN*
Institute of Linguistics,
†*Center for the Dynamics of Complex Systems,*
University of Potsdam, Germany
peter@ling.uni-potsdam.de

BRYAN JURISH
Berlin-Brandenburg Academy of Sciences, Berlin, Germany

DOUGLAS SADDY† and STEFAN FRISCH
Institute of Linguistics,
University of Dynamics of Complex Systems,
University of Potsdam, Germany

Received January 5, 2002; Revised September 30, 2002

We describe a part of the stimulus sentences of a German language processing ERP experiment using a context-free grammar and represent different processing preferences by its unambiguous partitions. The processing is modeled by deterministic pushdown automata. Using a theorem proven by Moore, we map these automata onto discrete time dynamical systems acting at the unit square, where the processing preferences are represented by a control parameter. The actual states of the automata are rectangles lying in the unit square that can be interpreted as cylinder sets in the context of symbolic dynamics theory. We show that applying a wrong processing preference to a certain input string leads to an unwanted invariant set in the parsers dynamics. Then, syntactic reanalysis and repair can be modeled by a switching of the control parameter — in analogy to phase transitions observed in brain dynamics. We argue that ERP components are indicators of these bifurcations and propose an ERP-like measure of the parsing model.

Keywords: Language processing; local ambiguity; ambiguity resolution; pushdown automata; Gödel codes; symbolic dynamics; cylinder sets; entropy.

1. Introduction

Phase transitions in the human brain are well known in the processes of cognitive motor control [Kelso *et al.*, 1992; Engbert *et al.*, 1997] and they have been successfully modeled by nonlinear field theories of cortical activation [Jirsa, 2004; Wright *et al.*, 2004]. In human language processing phase transitions were observed by Raçzasek *et al.* [1999] using continuously varying prosodic parameters for the disambiguation of speech and they were modeled by Kawamoto [1993] for lexical ambiguity resolution using a Hopfield net. Due to Başar [1980], beim Graben *et al.* [2000b] and Frisch *et al.* [2004] have considered event-related brain potentials (ERPs) and event-related cylinder entropies as order parameters of the brain indicating disorder–order phase transitions when the

*Address for correspondence: Institute of Linguistics, University of Potsdam, P.O. Box 601553, 14415 Potsdam, Germany.

control parameters, i.e. the experimental conditions, assume critical values. In language processing ERP experiments the control parameters of the brain are given by the manipulations of the sentence material, e.g. preferred versus dispreferred continuations of local syntactic ambiguities. In German, as in many other languages, native speakers have been shown to prefer the grammatical function of the subject for an ambiguous nominal constituent rather than the object (see [Frisch *et al.*, 2004]). If the preferred analysis according to this so-called *subject preference strategy* is disconfirmed by following sentence material, processing problems arise and revision (a so-called *reanalysis*) must be initiated (see [Fodor & Ferreira, 1999] for an overview).

Since the manipulations in language processing ERP experiments are symbolic, they were not well suited for quantitative dynamical modeling. Therefore most efforts in modeling language processing have been undertaken in developing symbol manipulating automata [Aho & Ullman, 1972; Hopcroft & Ullmann, 1979]. According to this approach one considers formal languages which can be processed by finite control machines. A formal language is a subset of all strings obtained by the concatenation of "letters", the so-called *terminals* from a finite *alphabet*. Such a subset might be generated by a *grammar*, i.e. a finite production system of "rules" describing the substitution of strings of letters and auxiliary symbols, called *nonterminals* by strings of terminals and/or nonterminals. A *context-free grammar* is a production system where each rule must describe the substitution of one single nonterminal into a string of terminals and/or nonterminals. A context-free grammar is called *locally ambiguous* if there are two or more rules *expanding* the same nonterminal symbol. One refers to the history of rules applied to obtain a certain string of terminals as a *derivation*. A *left derivation* is a derivation where always the leftmost nonterminal symbol is expanded. A context-free grammar is called *ambiguous* if there are more than one left derivations yielding the same sequence of terminal symbols. Languages generated by unambiguous context-free grammars can be processed by *deterministic pushdown automata*. These are devices with a finite control and with a one-sided infinite memory tape, the so-called *stack*. A simple subclass of pushdown automata are *deterministic top-down recognizers* with only one internal state. Languages with local ambiguities can also be processed with deterministic pushdown automata, however they need a *look-ahead* into the terminal string to be processed. Without this look-ahead the processing of locally ambiguous languages may lead to *garden path effects* when the automaton makes wrong predictions about the structure of a string.

In the last decades, physicists continued to be interested in the physics of symbolic computation. In his compelling review, Bennett [1982] describes "ballistic" and "Brownian" computers that are computationally equivalent to Turing machines. These dynamical systems generate trajectories "isomorphic with the desired computation" from initial conditions regarded as inputs (see also [Crutchfield, 1994]). The formal equivalence of Turing machines and nonlinear dynamical systems has been proven by Moore [1990, 1991b]. He demonstrated that any symbolic system can be mapped onto a piecewise affine linear map defined at the unit square by interpreting the symbols as integers from some b-adic number system.

In this paper, we use Moore's construction to map top-down recognizers (parsers) that recognize context-free languages onto such dynamical systems. To capture the dynamics of nondeterministic parsers we decompose a locally ambiguous grammar into a set of unambiguous grammars that can be processed by deterministic top-down recognizers. After representing these parsers by nonlinear dynamical systems, we mapped their numbers onto the values of a control parameter, thus obtaining *one* bifurcating dynamical system at the unit square. This control parameter reflects the parsing strategy. Then, psycholinguistic phenomena, such as garden path effects in the processing of ambiguous structures [Frisch *et al.*, 2004] can be roughly modeled as a kind of phase transition in the dynamical system elicited by a switching of the control parameter.

We pursue three goals by this study. Firstly, we argue that cognitive computation is essentially transient dynamics in the sense of Bennett [1982] and Crutchfield [1994]. That is, a problem is translated into an initial condition of a system from which the dynamics evolves along a transient trajectory that is isomorphic to a running computer program. Finally, an attractor or some given state is reached which is considered as the solution of the problem. Secondly, we suggest an approach that bridges the cleft between dynamical models on one hand and symbolic computation on the other hand [van Gelder, 1998;

Marcus, 2001], namely by arguing that symbolic dynamics provides an interface between dynamical system theory and the theory of formal languages and automata. And thirdly, we make use of these findings in order to demonstrate that cognitive (symbolic) computation is essentially nonlinear dynamics as follows from Moore's proof [Moore, 1990, 1991b].

The organization of the paper is as follows. In Sec. 2 we discuss again the language processing ERP experiment of Frisch *et al.* [2004] and we present a toy-model of parsing by dynamical systems. Subsequently, we provide a formal theory of dynamical language processing in Sec. 3. In Sec. 4 we shall address some problems and open questions by offering possible generalizations of our approach.

2. A Toy-Model of Language Processing

In order to illustrate our modeling approach, we consider a part of the sentence material used by Frisch *et al.* [2004] in their ERP experiment on the processing of ambiguous pronouns. In this study, the following two conditions — among others — were tested:

(1) *Nachdem die Kommissarin den Detektiv getroffen hatte, sah sie*$_s$
after the cop the detective met had saw she$_s$
[*den Schmuggler*]$_o$.
[the smuggler]$_o$.
"After the cop had met the detective, she saw the smuggler."

(2) *Nachdem die Kommissarin den Detektiv getroffen hatte, sah sie*$_o$
after the cop the detective met had saw she$_o$
[*der Schmuggler*]$_s$.
[the smuggler]$_s$.
"After the cop had met the detective, the smuggler saw her."

Here, we have tagged the nominal constituents of the main clause (the second part after the comma) of the sentences (1) and (2) by their grammatical roles, where s denotes "subject" and o denotes "object". The sentences (1) and (2) of the ERP experiment are therefore mapped onto the strings *so* and *os*, respectively constituting the formal language $L = \{so, os\}$ from the alphabet $\{s, o\}$.

2.1. *Context-free grammars*

Next we ask for a grammar describing the language L. Since we are only interested in describing the main clauses of the sentences (1) and (2), we propose the following *context-free grammar* G.

$$S \to s\,o \tag{3}$$

$$S \to o\,s \tag{4}$$

where S means "sentence" (the *start-symbol* of the grammar). The rule (3) assigns subject to the pronoun (*sie*: "she") and object to the second noun phrase of the clause (*den Schmuggler*: "the smuggler"), while rule (4) does the opposite: the pronoun becomes object and the second noun phrase becomes subject (*der Schmuggler*). By applying the rules (3), (4) one obtains the *context-free language* $L = \mathcal{L}(G) = \{so, os\}$. Our toy-grammar $G = \{(3), (4)\}$ is locally ambiguous. That means that there is more than one possibility to expand the nonterminal S of the grammar into a string of terminals consisting of s or o (for a formal treatment of local ambiguity see Sec. 3.3).

Let us look for a coarse-grained model of the processing strategies. Psycholinguists have found that a subject interpretation of a constituent which is ambiguous between subject and object is preferred and that a later disambiguation towards an object interpretation is more difficult to process [Schlesewsky *et al.*, 2000; Frisch *et al.*, 2002]. Due to this *subject preference strategy* the first encountered noun phrase is interpreted to be the subject of a clause [Frisch *et al.*, 2004]. This strategy corresponds to the rule (3) of the toy-grammar because (3) creates the sequence *so* for the matrix clause. Contrarily, the rule (4) assigns the object role to the pronoun which is the dispreferred interpretation. We can thus split the locally ambiguous grammar (3),

(4) into two unambiguous ones: the subgrammar G_1 consisting only of the rule (3) models the *subject preference strategy* while the subgrammar G_2 which is formed by the single rule (4) produces the string *os* hence modeling example (2). The subgrammars G_1 and G_2 generate the formal languages $L_1 = \mathcal{L}(G_1) = \{so\}$ and $L_2 = \mathcal{L}(G_2) = \{os\}$, respectively. We show in Sec. 3.3 that any locally ambiguous context-free grammar can be partitioned into a finite set of locally unambiguous context-free subgrammars.

2.2. *Top-down parser*

So far we have modeled the sentence material and the processing preferences of the ERP experiment that has been described by Frisch *et al.* [2004]. Next we shall construct a processing model. It is well known from computer science that context-free languages are recognized by pushdown automata [Aho & Ullman, 1972; Hopcroft & Ullmann, 1979]. The languages produced by our toy-grammars G_1 and G_2 (rules (3) and (4)) can be processed by a subclass of pushdown automata, the so-called *top-down parsers*, which are the most simple of these devices (cf. Sec. 3.4). A top-down parser consists of two tapes: the *input tape*, containing the string being processed, and a *stack memory* for storing temporary information. Both tapes can be described by strings γ, w where $\gamma = \gamma_{i_0}, \ldots, \gamma_{i_{k-1}}$ denotes the concatenation of k symbols on the stack and $w = w_{j_1}, \ldots, w_{j_l}$ denotes a sequence consisting of l symbols in the input. While the input may only contain terminal symbols, the stack is able to store terminal and nonterminal symbols as well. The doublet (γ, w) of stack and input is called the *actual pair* of the automaton.[1] The top-down parser only has access to the first symbol $Z = \gamma_{i_0}$ on the stack and to the first item $a = w_{j_1}$ at the input. Depending on these "visible" symbols the parser determines its actions at input and stack.

Let us consider the language $L_1 = \mathcal{L}(G_1) = \{so\}$ for an example. A top-down parser τ_1 that recognizes L_1 can be described by a table of actual pairs [Allen, 1987] that is shown in Table 1. At the first step the parser is initialized with the start symbol $Z = S$ on the stack and the whole string $w = so$ in the input. The device "sees" the start symbol at the top of the stack (indicated by the bold font) and first recognizes that it is a nonterminal of the grammar G_1. Then it looks into the list of production rules to find a rule with the start symbol on the left-hand side in order to *predict* the right-hand side. Since G_1 is unambiguous, the parser uniquely finds the rule (3) for expanding S into *so*; that is the operation given in the fourth column of Table 1.

After the first step the actual pair is (**s***o*, **s***o*), where the visible entries are printed in bold. Now the parser recognizes that the leading symbol on the stack is the terminal *s*. In this case the parser tries to *attach* the predicted symbol with the current input. This is possible if both symbols agree. By attachment the parser cancels the coincident symbols of stack and input tape and it shifts both tapes one step to the left thus yielding state two. Now, the leading symbols of stack and input tape are again corresponding terminals, so that the parser performs a further attachment. Finally, the automaton reaches its *accepting state* where both tapes, stack and input are empty (or, technically speaking, they contain the *empty word* ε).

As for the language L_1 we do the construction of the recognizing top-down parser for the other

Table 1. Top-down parser τ_1 processing *so* according to grammar G_1. The leading symbols at stack and input tape are printed in bold fonts.

Step	Stack	Input	Operation
0.	**S**	**s***o*	predict by rule (3)
1.	**s***o*	**s***o*	attach
2.	**o**	**o**	attach
3.	ε	ε	accept

Table 2. Top-down parser τ_1 processing $os \in L_2$ according to grammar G_1. The leading symbols at stack and input tape are printed in bold fonts.

Step	Stack	Input	Operation
0.	**S**	**o***s*	predict by rule (3)
1.	**s***o*	**o***s*	**attachment fails**
2.	**s***o*	**o***s*	nonaccepting final state

[1] The term "state" has quite different meanings in computational linguistics and in dynamical system theory, respectively. However, as will become clear subsequently, the actual pairs of a top-down parser correspond to the (macro-)states of the assigned dynamical system. We shall therefore refer to actual pairs as "states" in the dynamical sense, unless stated otherwise (as in Sec. 3.3).

language L_2 in the same manner thus obtaining an automaton τ_2. What happens if one employs a wrong processing strategy? We shall discuss this issue on the example of processing the string $os \in L_2$ by the parser τ_1. Table 2 shows this process.

The parsing process fails and ends up in a nonaccepting state of the automaton. This can be regarded as a very crude model of a *garden path interpretation* in psycholinguistics [Fodor & Ferreira, 1999]. In automata theory such miss-parses must be corrected by *backtracking* [Aho & Ullman, 1972] that may serve as a model of psycholinguistic *reanalysis* [Fodor & Ferreira, 1999].

2.3. *Gödel encoding*

The parsing processes described in the last subsection are obviously reminiscent to dynamics. The Tables 1 and 2 provide dynamic computation in the sense of Bennett [1982] and Crutchfield [1994]: the problem to be solved is formulated by an initial condition, namely an actual pair of the parser; the computational process is a sequence of such states; i.e. a transient trajectory and the result of the computation corresponds to an attractor, either wanted or unwanted. This raises the question whether it is possible to translate parsing into nonlinear dynamics. Indeed, it is. As mentioned above, the actual pairs (γ, w) of the top-down parser can be considered as states belonging to some phase space. We shall discuss in this subsection how this symbolic space is mapped on a vector space (for a formal proof, see Sec. 3.4).

The idea of the construction is to map the common terminals s, o and the nonterminal S of the two languages L_1 and L_2 onto integer numbers. This mapping g is called a *Gödel encoding* [Gödel, 1931; Hofstadter, 1979]. The way in which this is achieved is completely arbitrary. We shall choose the following encoding

$$\begin{aligned} g(o) &= 0 \\ g(s) &= 1 \\ g(S) &= 2\,. \end{aligned} \tag{5}$$

From these Gödel codes of the symbols we compute the values of symbol strings simply by a b-adic number expansion of proper fractions, where we use the number of terminals and nonterminals together, $b_1 = 3$, for representing the stack and the number of terminal symbols, $b_2 = 2$, for representing the input tape. Let us consider the first two rows of Table 1 as an example. Row one contains the actual pair (S, so). The start symbol S (Gödel code = 2) is interpreted as the fraction 0.2_3 in the 3-adic number system. Hence, we obtain the decimal fraction $0.2_3 = 2 \times 3^{-1} = 0.6667_{10}$. The decimal representation of the input tape is $0.10_2 = 1 \times 2^{-1} + 0 \times 2^{-2} = 0.5_{10}$. Thus, the actual pair (S, so) of the parser is mapped onto the doublet (0.6667, 0.5) which is a point in the unit square $[0, 1] \times [0, 1]$. The same calculation maps the actual pair (so, so) of the second row of Table 1 onto the point (0.3333, 0.5). We shall refer to these b-adic fractions also as Gödel codes and we shall use the symbols g_1 and g_2 for the extension of the map (5) to the contents of the stack and the input tape, respectively.

Now we have to address one of the most essential issues of our construction. For theoretical reasons (which will be thoroughly treated in the next section) we have to interpret the actual pairs of the parser not as points in the unit square but as rectangles lying in this phase space. We therefore consider the tapes of the parser not as being empty following the stored contents γ and w, e.g. $so\,\varepsilon$ at the input tape. Instead, we allow for any arbitrary continuation that is considered to be unknown, i.e.

$$\begin{aligned} [so] = \{&so\,ooo\ldots,\ so\,oos\ldots,\ so\,oso\ldots,\ldots, \\ &so\,sss\ldots\} \end{aligned} \tag{6}$$

We will see in the next section that these sets of strings all having a common building block are nothing else than those *cylinder sets*, which have been used by beim Graben *et al.* [2000b] and by Frisch *et al.* [2004] to analyze event-related brain potentials. Let us consider the infimum and the supremum of these sets represented by their Gödel fractions. The Gödel number of the infimum is given by $g_2(so) + g_2(o) = g_2(so)$. But the Gödel number of the supremum is $g_2(so) + 2^{-2} \times g_2(sss\ldots)$. From an expansion into a geometrical series we recognize that the value of the infinite binary fraction $g_2(sss\ldots) = 0.111\ldots_2$ is one. Hence the supremum is obtained as $g_2(so) + 2^{-2}$. We generalize these results to the formulas

$$\begin{aligned} \inf(g_{1;2}([\gamma])) &= g_{1;2}(\gamma) \\ \sup(g_{1;2}([\gamma])) &= g_{1;2}(\gamma) + b_{1;2}^{-|\gamma|} \end{aligned} \tag{7}$$

for any string γ of length $|\gamma|$ consisting of $b_{1;2}$ symbols that constitutes the cylinder set $[\gamma]$ (for a proof see Sec. 3.2).

Using this representation of strings by cylinder sets we eventually map the actual pairs (γ, w) of

the parsers onto Cartesian products of intervals

$$(\gamma,\, w) \mapsto [g_1(\gamma),\, g_1(\gamma) + b_1^{-|\gamma|}] \times [g_2(w),\, g_2(w) + b_2^{-|w|}] \qquad (8)$$

which are rectangles in the unit square. For the actual pairs of the first two rows of Table 1 we therefore get the rectangles $[0.6667,\, 1.0000] \times [0.5000,\, 0.7500]$ and $[0.3333, 0.4444] \times [0.5000,\, 0.7500]$.

2.4. *Parsing dynamics*

Using the Gödel encoding of actual pairs we are able to map the complete parsing process onto a trajectory of rectangles through the unit square. Table 3 shows the trajectory for the parse presented in Table 1.

Table 3. Trajectory of the parse of $so \in L_1$ processed by the top-down parser τ_1 according to grammar G_1.

Step	Rectangle
0	$[0.6667,\ 1.0000] \times [0.5000,\ 0.7500]$
1	$[0.3333,\ 0.4444] \times [0.5000,\ 0.7500]$
2	$[0.0000,\ 0.3333] \times [0.0000,\ 0.5000]$
3	$[0.0000,\ 1.0000] \times [0.0000,\ 1.0000]$

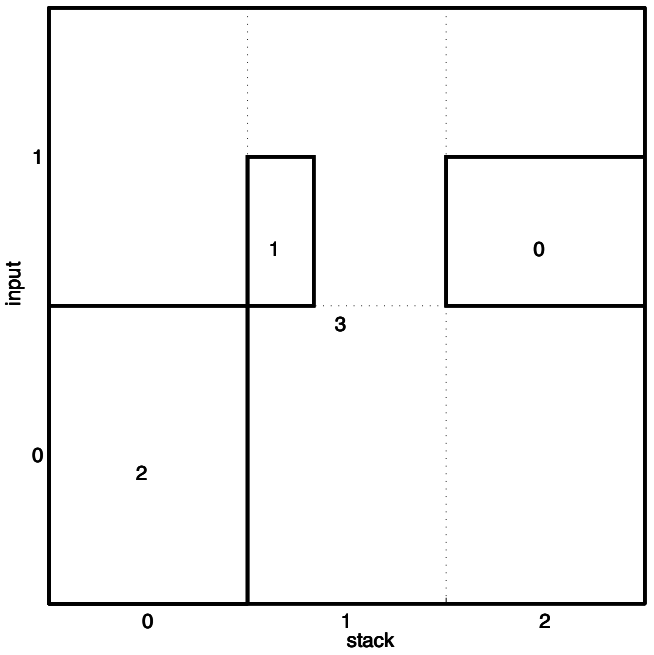


Fig. 1. Graphical representation of the trajectory of the parse of $so \in L_1$ processed by the top-down parser τ_1 according to grammar G_1. The numbers of the rectangles correspond to the steps in Table 3.

This trajectory is graphically displayed in Fig. 1, which suggests that there is a *deterministic flow* defined at the unit square which drives a rectangular set of points through the state space by stretching or squashing it. This flow actually exists and it can be constructed from the possible operations of the top-down parser. As we have seen in Sec. 2.2 the parser has only access to the topmost symbols of the stack and of the input tape. Therefore, only the actual pairs $(Z,\, a)$ completely determine the next operation of the parser. Since Z could be a terminal or a nonterminal and since a must be a terminal, these symbols provide a partition of the unit square into a grid of basic rectangles by their Gödel numbers, which are displayed in Fig. 1 by the dotted lines. At each of these dotted rectangles the parser operates in one of the following three ways:

(9) *Predict*; if Z is a nonterminal symbol, expand it into a string of terminals and/or nonterminals according to a unique rule of the grammar; the content of the input remains unchanged. By expanding Z into a string $\gamma i_1, \ldots, \gamma i_{k-1}$ it is mapped onto the symbol γ_{i_1}. This entails a shift of the rectangle along the x-axis in the Gödel-representation. Additionally, the content of the stack becomes longer, yielding a compression of the rectangle along the x-axis. From Fig. 1, we see that this happens by mapping the rectangle 0 onto the rectangle 1.

(10) *Attach*; if Z is a terminal symbol, compare it with a in the input tape. If they agree, delete both from stack and input. That means, the actual pair $(a,\, a)$ is mapped onto the actual pair $(\varepsilon,\, \varepsilon)$. According to the Gödel encoding the rectangle corresponding to $(a,\, a)$ is extended and shifted along the x- and the y-axes such that it coincides with the unit square. This operation was performed by mapping rectangle 2 onto rectangle 3 in Fig. 1. If, on the other hand, Z does not agree with a the parser arrives at a nonaccepting state.

(11) *Accept*; the parse successfully terminates when the whole unit square $[0,\, 1] \times [0,\, 1]$ is covered by the last rectangle. Otherwise any rectangle corresponding to an actual pair $(Z,\, a)$ where none of these operations applies, is an *invariant set* that is mapped onto itself.

The striking result of our construction due to Moore's proof is that the deterministic dynamics of

a top-down parser is a piecewise affine linear map defined at those cells of the partition of the unit square, which correspond to the actual pairs (Z, a) of one symbol in the stack and one symbol in the input tape, respectively. Formally, we recognize that a parser τ_p is given by the map Φ_p acting as

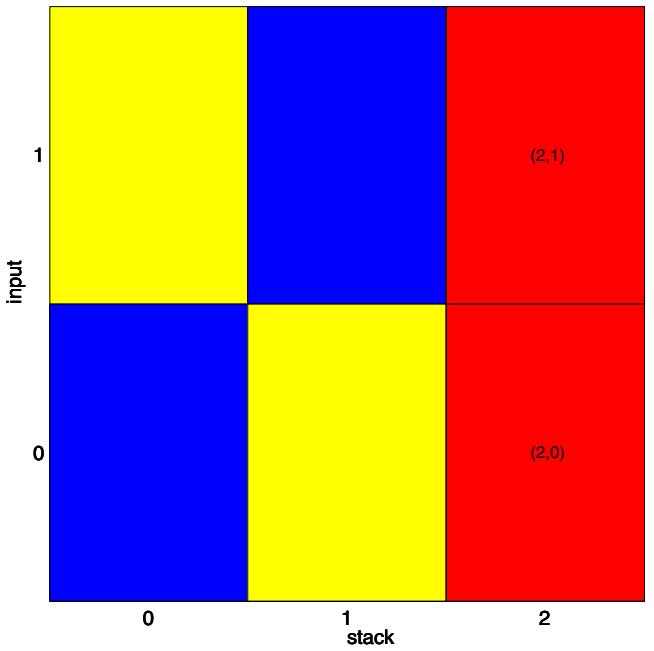
$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \Phi_p \left(\begin{bmatrix} x_t \\ y_t \end{bmatrix} \right) = \begin{bmatrix} a_x^{p,(i,j)} \\ a_y^{p,(i,j)} \end{bmatrix} + \begin{bmatrix} \lambda_x^{p,(i,j)} & 0 \\ 0 & \lambda_y^{p,(i,j)} \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix}, \tag{12}$$

where $[x_t, y_t]^T$ is a point in the unit square, $[x_{t+1}, y_{t+1}]^T$ is its iterate, and (i, j) denote the *domains of definition* that are given by the Gödel numbers $i = g(Z)$, $j = g(a)$ of the topmost symbols of stack and input tape providing the partition of the unit square. The coefficients $a_x^{p,(i,j)}$ and $a_y^{p,(i,j)}$ of the parser τ_p describe a parallel translation of a state whereas the matrix elements $\lambda_x^{p,(i,j)}$ and $\lambda_y^{p,(i,j)}$ mediate the stretching ($\lambda > 1$) or squeezing ($\lambda < 1$) of a rectangle. We present the values of these coefficients for the parser τ_1 accepting the language $L_1 = \{so\}$ in Table 4.

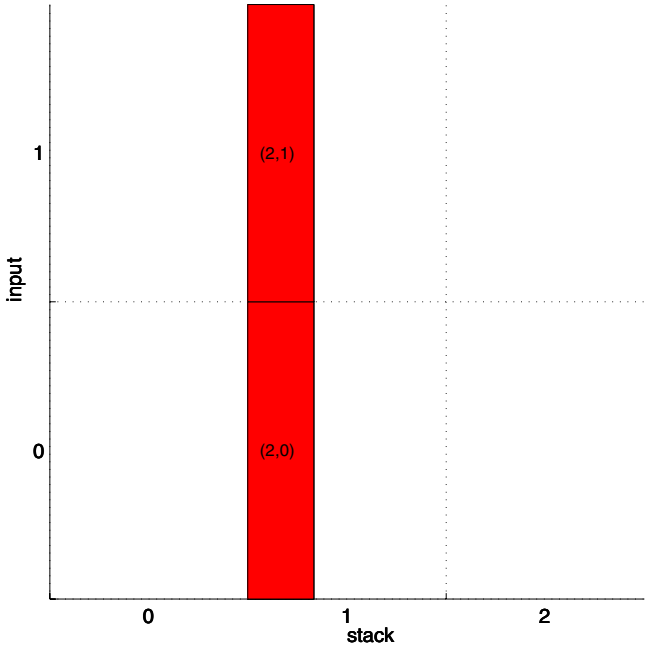
In Fig. 2 we display the domains of definition and the coefficients of the map (12) for the parser τ_1. Figure 2 encodes the possible operations of the top-down parser by colors, showing the domains of definition of the piecewise affine linear functions in (a) and their images in (b). Red rectangles which are labeled by their corresponding actual pairs in the Gödel numbering represent the *predict* operations, where the leading symbol of the stack is expanded into a string of symbols, thus entailing a translation of the rectangle in connection with its contraction along the x-axis. In (b) the red rectangles having the same labels are those images.

Table 4. Coefficients of the map (12) for the top-down parser τ_1 joined to grammar G_1.

Actual Pair	a_x	a_y	λ_x	λ_y
(0, 0)	0.0000	0.0000	3.0000	2.0000
(0, 1)	0.0000	0.0000	1.0000	1.0000
(1, 0)	0.0000	0.0000	1.0000	1.0000
(1, 1)	−1.0000	−1.0000	3.0000	2.0000
(2, 0)	0.1111	0.0000	0.3333	1.0000
(2, 1)	0.1111	0.0000	0.3333	1.0000

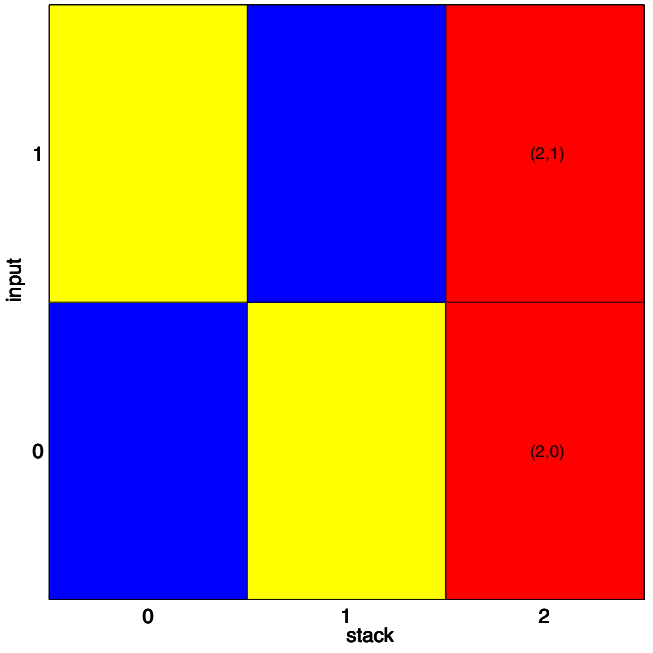


(a)

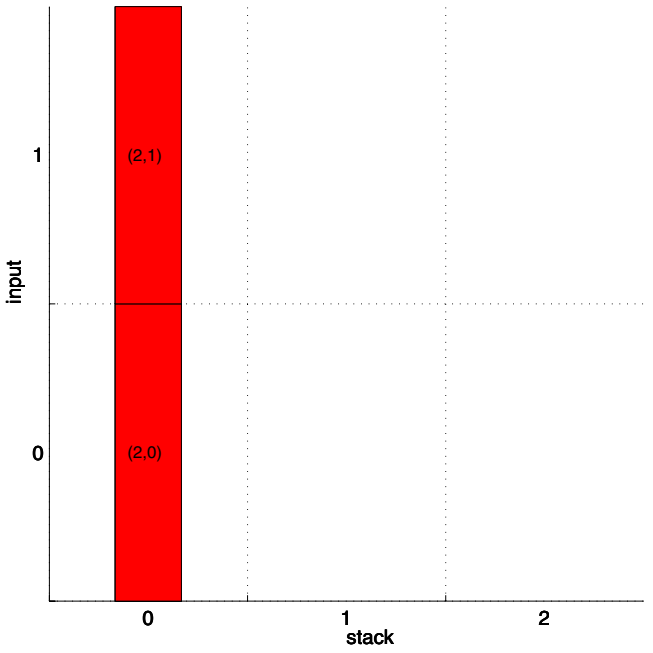


(b)

Fig. 2. (a) Domains of definition of the parsing map (12) for the top-down parser τ_1. The colors of the rectangles encode the operation of the parser at the corresponding actual pairs. Red: *predict*; the leading symbol of the stack is expanded into a string of symbols. Blue: *attach*; the leading symbol of the stack is a terminal agreeing with the leading symbol at the input tape. The blue rectangles are extended to the whole unit square. Yellow: *do not accept*; in the yellow domains the map is the identity. (b) Images of the red rectangles from (a). The red rectangles are the images of those red rectangles from (a) labeled by the same actual pair (i, j).



(a)



(b)

Fig. 3. (a) Domains of definition of the parsing map (12) for the top-down parser τ_2. (b) Images of the red rectangles from (a). For the color scheme, see Fig. 2.

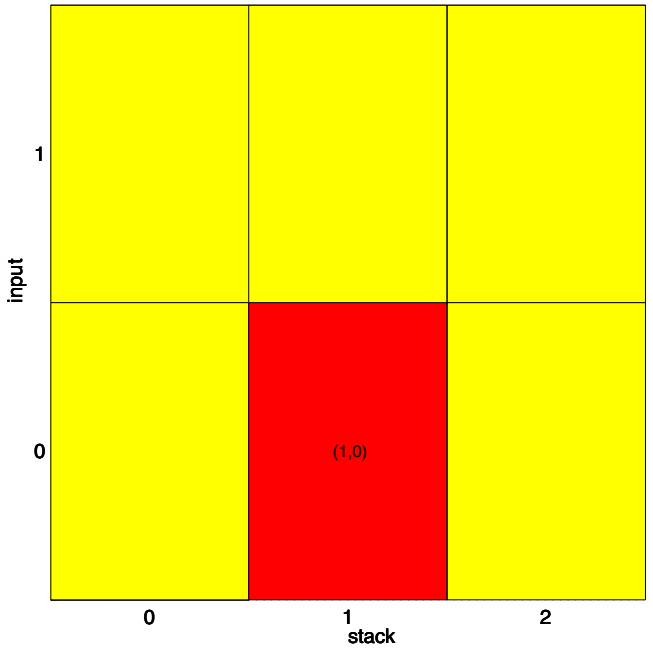
Blue rectangles in (a) represent the *attach* operation. Their images are the whole unit square. Yellow rectangles in (a) represent actual pairs where neither *predict* nor *attach* nor *accept* can be applied.

So far we have discussed the dynamics of the parser τ_1 processing the string *so* according to the *subject preference strategy*. Additionally, we present also the map of the parser τ_2 that processes the string *os* with respect to the grammar G_2 in Fig. 3. Obviously, as Figs. 2 and 3 reveal, the domains of definition are the same for all parsers maps. Only the images of the red rectangles differ due to the different grammars which are used to expand the nonterminal symbols on the stack.

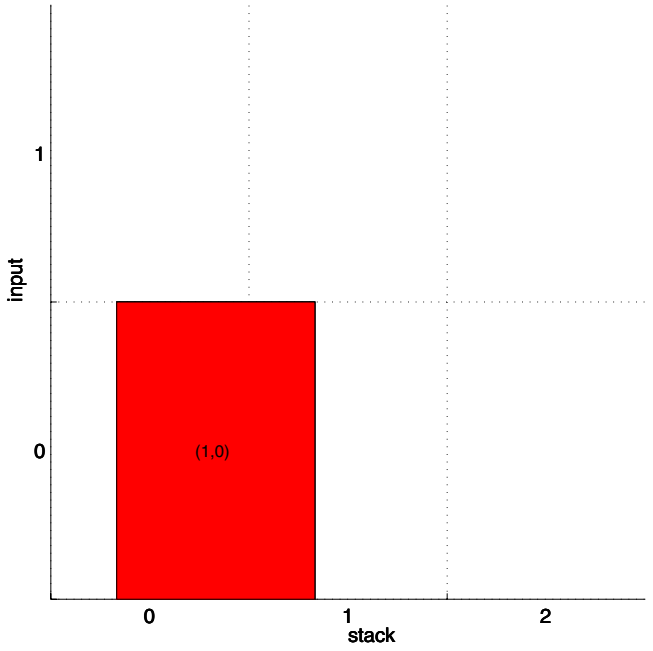
After performing this construction of a discrete time deterministic dynamical system for the two top-down parsers τ_1 and τ_2, any string of the language $L = \{so, os\}$ can be processed by each of these dynamical systems. To achieve this, the string must be translated into an interval at the y-axis by the b_2-adic Gödel encoding g_2. Moreover, one has to compute the image of the start symbol S that initializes the stack of the parsers under the Gödel encoding g_1. Thus we obtain a rectangle R_0 in the unit square as the image of the first actual pair. To carry out a simulation, we prepare an ensemble of points $\mathbf{x}_0^i = [x_0^i, y_0^i]^T$, $i = 1, 2, \ldots, M$, randomly chosen in R_0 as initial conditions of the dynamics. From each of these points the first iterate $\mathbf{x}_1^i = \Phi_p(\mathbf{x}_0^i)$ of the map (12) is computed. The envelope of this set yields the rectangle R_1 in the unit square corresponding to the second actual pair of the parse. By performing the iteration procedure recursively: $\mathbf{x}_t^i = \Phi_p(\mathbf{x}_{t-1}^i)$, we obtain an ensemble of M trajectories $\{\mathbf{x}_t^i | t \in \mathbb{N}_0, i = 1, 2, \ldots, M\}$ exploring the phase space of the parsing dynamics. The envelopes of the sets of all states at the trajectories at each time t provide a sequence of rectangles R_t that are equivalent to the actual pairs of the parser τ_p due to the Gödel encoding. In this sense we have constructed a dynamical toy-model of language processing.

2.5. *Diagnosis and repair*

In Sec. 2.2 we have seen that processing the string *os* according to the "wrong" grammar G_1 leads to the nonaccepting final state (*so*, *os*). This actual pair is situated within the yellow rectangle (1, 0) in the unit square (see Fig. 3) where all parsing maps are evidently given by the identity. The rectangle corresponding to this actual pair is therefore an *invariant set* of the map (12). Hence, we recognize that both the top-down parser τ_1 and its associated map Φ_1 lead to a garden path interpretation by processing *os*. In automata theory, such a dead-end can only



(a)



(b)

Fig. 4. (a) Domains of definition of the repair map for the control parameter $r = 1/2$. (b) Images of the red rectangles from (a). For the color scheme, see Fig. 2.

be left by backtracking [Aho & Ullman, 1972]. We shall demonstrate in this section that our dynamical system model of language processing allows to establish a dynamics managing reanalysis without any backtracking.

Up to now, we have considered the two maps Φ_1 and Φ_2 constructed from the parsers τ_1 and τ_2 as being distinct dynamical systems. However, from the physical point of view it is much more natural to consider them as only *one* dynamical system depending on an additional *control parameter*, because all these maps are living at the same state space that is partitioned in the same way. For the sake of convenience we will normalize the possible values of the control parameter r to the unit interval $[0, 1]$. We take the predecessor of the parser's number p as the control parameter, $r = p - 1$. Thus, the maps Φ_p become parameterized as Φ_r.

What happens when the wrong parsing strategy $r = 0$ ($p = 1$) is used for processing the string *os* that must be correctly parsed by the strategy $r = 1$ ($p = 2$)? As we know, the dynamics reaches an unwanted invariant set represented by the actual pair of Gödel numbers $(1, 0)$. Assuming that the map Φ_0 is iteratively invoked by some "cognitive control unit" (regarded as a "homunculus"), this higher level system eventually recognizes the nonacceptable attractor $(1, 0)$ in the parser's phase space. We may call this recognition process *diagnosis* [Fodor & Ferreira, 1999]. After diagnosing the garden path analysis the control unit will turn the control parameter of the parsing dynamics towards the right choice $r = 1$. However, this will not help at all because we know that the state $(1, 0)$ is an invariant set for every value of the control parameter. The solution is to introduce an intermediary dynamical system to *repair* the failed parse. We assign this repair dynamics to the control parameter $r = 1/2$. Figure 4 presents the domains of definition and the images of this map.

The map $\Phi_{1/2}$ is constant for almost every cell of the partition but not at $(1, 0)$. Having this intermediary map available, the control unit changes the control parameter from $r = 0$ to $r = 1/2$ for one iteration. This causes the dynamics to leave the invariant set after the next iteration. The system does not trace the trajectory in the past, as would be necessary for backtracking. Finally, the control unit processes further with $r = 1$. Figure 5 shows the complete parse with garden path interpretation ($r = 0$), diagnosis and repair ($r = 0 \mapsto r = 1/2$), and completion ($r = 1$).

In dynamical system theory, changing control parameters beyond critical values causes *qualitative changes* in the structure of the flow at the state space, i.e. *bifurcations*. Thus, we see that our language processing dynamics can be modeled

in analogy to a bifurcating dynamical system when garden path interpretations are diagnosed and repaired.[2]

2.6. *Parsing entropy*

In the language processing experiment that motivates our dynamical system modeling, Frisch *et al.* [2004] observed a P600 ERP component elicited by the revision of the *subject preference strategy*. The authors also presented an alternative analysis of ERP by means of symbolic dynamics where ERP components are usually reflected by decreases of the cylinder entropy, thus indicating disorder–order phase transitions of the brain dynamics [beim Graben *et al.*, 2000b]. In this subsection we shall present a way to measure the entropy of the parsing states of our model which is justified by the equivalence — shown in the next section — of the actual pairs of a parser with cylinder sets of a symbolic dynamics.

Entropy is a measure of uncertainty based on probability distributions. Shannon and Weaver [1949] defined the entropy of a discrete distribution $\{(i,\, p_i)|i = 1, \ldots,\, n\}$ by

$$H = -\sum_{i=1}^{n} p_i \log p_i \,. \tag{13}$$

In dynamical system theory the probability p_i for occupying the ith cell B_i of a partition of the phase space can be estimated by the relative dwelling time of typical trajectories in B_i [Ott, 1993]. We shall use a similar argument utilizing the relative area of a rectangle in the unit square with respect to a *measurement partition*. To be precise, we define

$$p_i(R) = \frac{\text{area}\ (R \cap B_i)}{\text{area}\ (R)} \tag{14}$$

where the geometrical function $\text{area}\ (R) = (x_2 - x_1) \cdot (y_2 - y_1)$ determines the area of the rectangle $R = [x_1,\, x_2] \times [y_1,\, y_2]$.

By applying Eqs. (13) and (14) to the trajectories of rectangles of our parsing dynamics using an appropriately chosen measurement partition that is

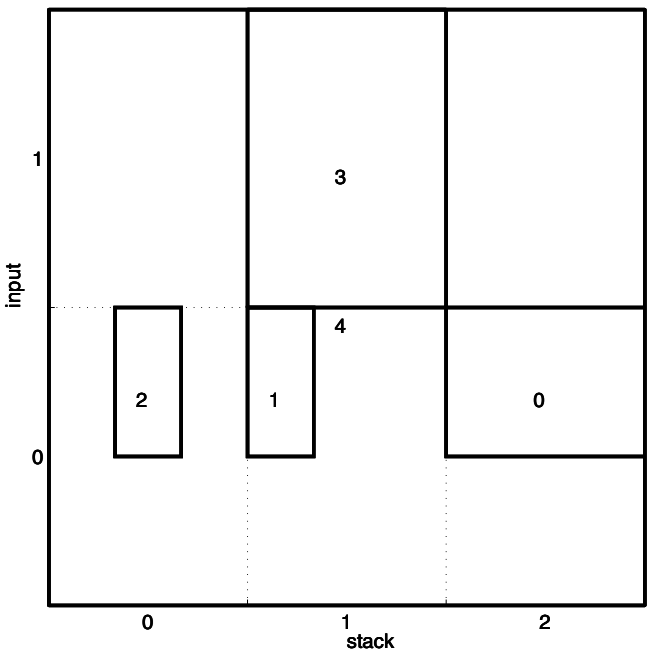


Fig. 5. Graphical representation of the diagnosis and repair trajectories. Reanalysis of the parse of $os \in L_2$ processed by the dynamics Φ_0 according to grammar G_1: steps 0 and 1; the state 1 corresponds to the unwanted garden path. This is recognized by the control unit which subsequently turns the control parameter to $r = 1/2$ (diagnosis). Step 2 yields the repaired state. Then the control unit changes the control parameter to $r = 1$ to complete the parse according to the right grammar G_2: steps 2–4.

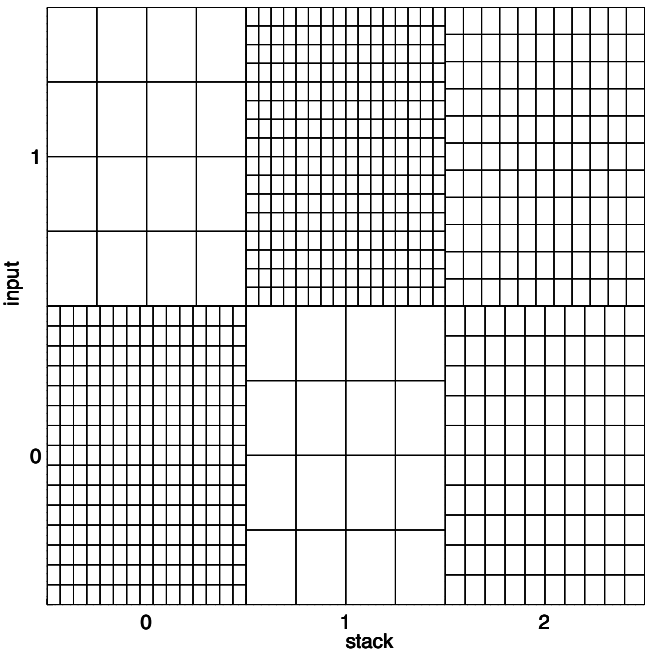


Fig. 6. Measurement partition of the unit square for determining parsing entropies.

[2]Similarly, Kawamoto [1993] has modeled the access to an alternative meaning of a lexically ambiguous word by the habituation of synaptic weights in a Hopfield neural network entailing a bifurcation of the network's energy landscape.

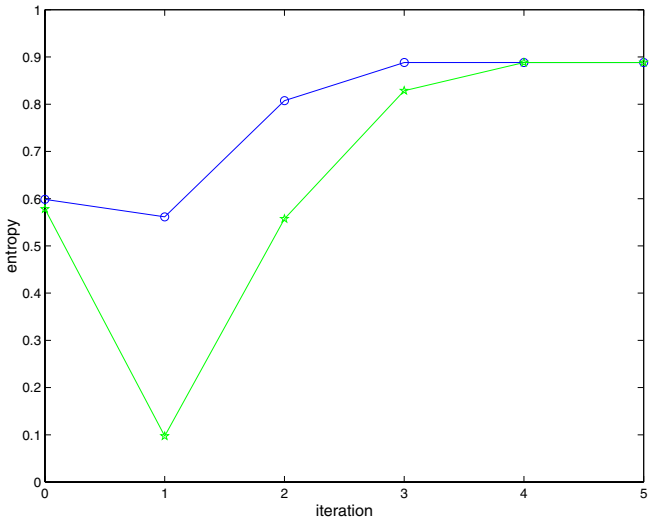


Fig. 7. Time coarse of parsing entropies due to the *subject preference strategy* from the dynamical systems according to the measurement partition shown in Fig. 6. The colors correspond to the ERP components and event-related entropies presented in [Frisch *et al.*, 2004]. Blue denotes the string *so*, green *os* where the revision of the *subject preference strategy* evokes a P600 ERP (the trajectory is shown in Fig. 5).

shown in Fig. 6, we obtain the time course of the parsing entropies presented in Fig. 7.

For a better visualization, Fig. 6 displays the mesh density of the measurement partition in a logarithmic scale. From Eqs. (13) and (14) one easily recognize that the probability $p_i = 1$ when the rectangle R representing the parsers state is completely covered by one cell B_i of the measurement partition. This is either the case when R is too small or the grid of the partition is too coarse, such that R drops through the grid, leading to zero entropy. This almost happens by construction of the measurement partition at the domain $(0, 1)$ where the garden path interpretation of our dynamical model is situated. On the other hand, when R is completely covered by N cells B_i of equal size, the overlap between one of these cells and R becomes area $(R)/N$, entailing $p_i = 1/N$ and thus maximal entropy.

As Fig. 7 reveals, the measurement partition was constructed in such a way that the detection of the garden path analysis in processing the string according to the inappropriate strategy leads to a decrease in parsing entropy as is the case in proper language processing [Frisch *et al.*, 2004]. Therefore our toy-model of language processing may offer a possible interface between dynamical and computational modeling on the one hand and psycholinguistic experiments and nonlinear data analysis on the other hand.

3. Theory of Language Processing by Dynamical Systems

This section is devoted to a formal theory of dynamical language processing that has already been outlined by beim Graben *et al.* [2000a]. After mentioning some basic concepts of dynamical system theory and symbolic dynamics, we will quote the fundamental theorem of Moore [1990, 1991b] on mapping Turing machines onto dynamical systems at the unit square. Then we shall apply Moore's proof to construct dynamical systems from pushdown automata. First, we recall the concepts of discrete time dynamical systems in one and more dimensions.

3.1. *Symbolic dynamics of one-dimensional systems*

A discrete time deterministic dynamical system is a pair (X, Φ_r), where X is the *phase space* while the *flow*[3] $\Phi_r : X \to X$ is an invertible map assigning to a state x_t at a certain time t its successor x_{t+1} at time $t + 1$, occasionally depending on a control parameter r. A given state x_{t_0} at a certain time t_0 is called *initial condition* of the dynamics. The set of states $\{x_t | x_t = \Phi_r^t(x_{t_0}), t \in \mathbb{Z}\}$ generated by the flow from the initial condition x_{t_0} is called a *trajectory* of the dynamical system. The powers of the map Φ_r^t are recursively defined by iterating the map: $\Phi_r^t = \Phi_r \circ \Phi_r^{t-1}$.

A special class of dynamical systems is provided by the unit interval $[0, 1]$ as their phase space and by a nonlinear function Φ_r mapping the unit interval on itself. These systems are known as one-dimensional dynamical systems [Collet & Eckmann, 1980]. The most simple one-dimensional dynamics are defined by piecewise (affine) linear maps at the unit interval. In the following, we shall consider the Bernoulli map [Schuster, 1989] as an intriguing example that shares many common properties with our parsing models discussed in Sec. 2.4. The Bernoulli map is defined by $\Phi(x) = 2x \mod 1$, i.e. the map is obtained by the linear function $y = f_0(x) = 2x$ at the subinterval $[0, 0.5]$ and by the affine linear function $y = f_1(x) = 2x - 1$ at the subinterval $]0.5, 1]$. Though being piecewise linear,

[3]For discrete time systems the flow is sometimes called *cascade* [Anosov & Arnol'd, 1988].

the map is nonlinear at the whole phase space $[0, 1]$ and exhibits the typical stretching and folding properties of chaotic dynamical systems. The Bernoulli map does not depend on a control parameter.

A binary expansion of the states of the Bernoulli map provides insight into its dynamics: for any number in the unit interval a representation by a proper binary fraction exists. Numbers smaller than 0.5 start with their *most significant digit* "0" while numbers greater than 0.5 begin with the binary digit "1". Let us consider the initial condition $x_0 = 0.1101$ for an example. Its decimal expansion is given by

$$\begin{aligned} x_0 &= \sum_{i=1}^{\infty} a_i \cdot 2^{-i} \\ &= 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} \\ &= 0.8125\,, \end{aligned} \tag{15}$$

where a_i are the binary digits vanishing for non-periodic fractions at some i. What are the successors of x_0 lying at a trajectory of the Bernoulli system? In the decimal number system, we obtain $x_1 = \Phi(x_0) = 0.6250$, $x_2 = \Phi(x_1) = \Phi^2(x_0) = 0.2500$, $x_3 = \Phi(x_2) = \Phi^3(x_0) = 0.500$, $x_4 = \Phi(x_3) = \Phi^4(x_0) = 0.0$. However, using the binary number system yields the trajectory $x_1 = \Phi(x_0) = 0.101$, $x_2 = \Phi(x_1) = 0.01$, $x_3 = \Phi(x_2) = 0.1$, $x_4 = \Phi(x_3) = 0.0$. One easily recognizes that the Bernoulli map acts on the binary numbers as a shift to the left discarding the 2^0 position.

The binary descriptions, e.g. "1101", of the states x together with the representation of the map Φ given by the left shift σ is called the *symbolic dynamics* of the Bernoulli map [Schuster, 1989]. Here, the binary digits "0", "1" are regarded as symbols from a finite alphabet A. States of the dynamical system are mapped onto sequences $s = a_{i_1} a_{i_2} a_{i_3} \ldots$ of symbols stemming from A. Sequences of finite length are called *words*. The set of all sequences of (one-sided) infinite length $s = a_{i_1} a_{i_2} a_{i_3} \ldots$ assumes the notation $A^{\mathbb{N}}$. As mentioned above, the most significant digit a_{i_1} of a binary fraction decides whether the state is taken from the lower or from the upper half interval: a_{i_1} = "0" means $x_0 \in [0, 0.5]$ whereas a_{i_1} = "1" means $x_0 \in]0.5, 1]$. Hence, the most significant digit a_{i_1} *partitions* the system's phase space into disjunct subsets. The symbols "0" and "1" are assigned to the cells $A_0 = [0, 0.5]$ and $A_1 =]0.5, 1]$ respectively, where the indices of the subsets of the partition are taken as symbols of the alphabet. In the same manner, the first two symbols $a_{i_1} a_{i_2}$ of a sequence decide where the initial condition x_0 and its first iterate $x_1 = \Phi(x_0)$ are situated. In the example discussed above, "11" means: $x_0 \in A_1$ and also $x_1 \in A_1$. The latter expression is equivalent to $\Phi(x_0) \in A_1$ and can be reformulated to $x_0 \in \Phi^{-1}(A_1)$ where Φ^{-1} denotes the preimage of a set. It is always defined even if the map itself is not invertible such as the Bernoulli map. By generalizing this expression, we obtain

$$s = a_{i_1} a_{i_2} a_{i_3} \ldots \Rightarrow x_0 \in \bigcap_{k=0}^{\infty} \Phi^{-k}(A_{i_{k+1}})\,. \tag{16}$$

When we apply this construction to the Bernoulli map we find that the first two symbols describe a partition of the unit interval into quarters: "00" means [0, 0.25], "01" means]0.25, 0.5], and so on. In general, by fixing the first n binary digits of a sequence $s = a_{i_1} a_{i_2} a_{i_3} \ldots$ we refer to the interval $[0.a_{i_1} \ldots a_{i_n} 00000 \ldots, 0.a_{i_1} \ldots a_{i_n} 11111 \ldots]$. These sets of symbol strings beginning with a fixed prefix have been introduced in Eq. (6) to define *cylinder sets*. A cylinder of length n is therefore a set of all sequences coinciding in the first n letters of the alphabet.

3.2. *Symbolic dynamics of high-dimensional systems*

Next, we shall introduce symbolic dynamics of general time discrete dynamical systems depending on some control parameters $r \in \mathbb{R}^p$; the phase spaces of which are subsets in some $\mathbb{R}^m$. A coarse-grained description of the dynamics is gained by a partition covering the phase space of the dynamical system [Beck & Schlögl, 1993; Badii & Politi, 1997]. Let $\{A_i | i = 1, 2, \ldots, I\}$ be a family of I pairwise disjunct subsets covering the whole phase space X, i.e. $\bigcup_{i=1}^{I} A_i = X$; $A_i \cap A_j = \emptyset$, $i \neq j$. The index set $A = \{1, 2, \ldots, I\}$ of the partition is interpreted as a (finite) alphabet of letters $a_i = i$. In the case of an invertible deterministic dynamics we are able to determine the system's past as well as its future by iterating the inverse flow Φ_r^{-1} and the flow Φ_r, respectively. By deciding which cell A_i of the partition is visited by a state x_t at time $t = \ldots - 1, 0, 1, \ldots$ we assign a symbol $\ldots, a_{i_{-1}}, a_{i_0}, a_{i_1}, \ldots$ at each instance of time. Thus, we obtain a bi-infinite sequence of letters $s = \ldots a_{i_{-1}} a_{i_0} a_{i_1} a_{i_2} \ldots$ from A. The expression $A^{\mathbb{Z}}$ refers to the set of all these bi-infinite strings of symbols. Given a time discrete and invertible deterministic dynamics Φ_r we construct a map $\pi : X \to A^{\mathbb{Z}}$ that assigns initial

conditions $x_0 \in X$ to bi-infinite symbol strings $s \in A^{\mathbb{Z}}$ by the rule $\pi(x_0) = s$, iff $x_t = \Phi_r^t(x_0) \in A_{i_t}$, $t \in \mathbb{Z}$. Thus, π maps initial conditions x_0 in the state space onto symbolic strings regarded as coarse-grained trajectories starting at x_0. By doing so, the flow Φ_r is mapped onto the left shift σ, as in the case of the Bernoulli map. The shift is hence a map $\sigma : A^{\mathbb{Z}} \to A^{\mathbb{Z}}$ acting according $\sigma(\ldots a_{i_{-1}}\hat{a_{i_0}}a_{i_1}a_{i_2}\ldots) = \ldots a_{i_{-1}}a_{i_0}\hat{a_{i_1}}a_{i_2}\ldots$ where the hat denotes the current state under observation. The map π mediates between the quantitative states $x \in X \subset \mathbb{R}^m$ and the states of the symbolic dynamics $s \in A^{\mathbb{Z}}$ by

$$\pi(\Phi_r(x_t)) = \sigma(\pi(x_t))\,. \tag{17}$$

The map π might not be invertible. If π is invertible the partition is called generating and every string of symbols corresponds to exactly one initial condition generating it [Beck & Schlögl, 1993].

Nevertheless, if π is not invertible, we can apply the map π^{-1} at subsets of $A^{\mathbb{Z}}$ namely at strings of finite length, looking for their preimages in X. In order to do this we generalize the notion of cylinder sets. Let $t \in \mathbb{Z}$, $n \in \mathbb{N}$ and $a_{i_1}, \ldots, a_{i_n} \in A$. The set

$$[a_{i_1}, \ldots, a_{i_n}]_t = \{s \in A^{\mathbb{Z}} | s_{t+k-1} = a_{i_k}\,, \quad k = 1, \ldots, n\} \tag{18}$$

is called *n-cylinder* at time t. For further references see [Badii & Politi, 1997; Beck & Schlögl, 1993; beim Graben *et al.*, 2000b]. Since cylinders are subsets of $A^{\mathbb{Z}}$ of infinite strings coinciding in a sequence of time points $\{t, t+1, \ldots, t+n-1\}$, we can determine their preimages under the deterministic dynamics Φ_r

$$\pi^{-1}([a_{i_1}, \ldots, a_{i_n}]_t) = \bigcap_{k=0}^{n-1} \Phi^{-k}(A_{i_{k+1}})\,. \tag{19}$$

This is almost the same formula as Eq. (16).

As for the Bernoulli map any general symbolic dynamics can be mapped back onto a quantitative dynamical system by performing a b-adic expansion. To archive this the symbols of the alphabet A must be encoded by integers. If I is the cardinality of the alphabet we need I digits $0, 1, 2, \ldots, I-1$ representing the letters $a_1, a_2, a_3, \ldots, a_I$. We know the assignment $g(a_i) = i - 1$ as a *Gödel encoding* from Sec. 2.3. Given a finite or bi-infinite symbolic sequence $s = \ldots a_{i_{-1}}\hat{a_{i_0}}a_{i_1}a_{i_2}\ldots$ where the hat denotes the current state again, we expand the left-half sequence $s_- = \ldots a_{i_{-3}}a_{i_{-2}}a_{i_{-1}}a_{i_0}$ into the I-adic fraction $x = \ldots + g(a_{i_{-3}})I^{-4} + g(a_{i_{-2}})I^{-3} + g(a_{i_{-1}})I^{-2} + g(a_{i_0})I^{-1}$ and the right-half sequence $s_+ = a_{i_1}a_{i_2}a_{i_3}\ldots$ into the I-adic fraction $y = g(a_{i_1})I^{-1} + g(a_{i_2})I^{-2} + g(a_{i_3})I^{-3}\ldots$. The sequence s is thus mapped onto a pair (x, y)

$$x = \sum_{k=0}^{\infty} g(a_{i_{-k}})I^{-k-1} \tag{20}$$

$$y = \sum_{k=1}^{\infty} g(a_{i_k})I^{-k} \tag{21}$$

of real numbers lying in the unit square $[0, 1]\times[0, 1]$.

Furthermore, as cylinder sets of the Bernoulli map were represented by subintervals of the unit interval, we shall demonstrate that cylinder sets of an arbitrary symbolic dynamics [Eq. (18)] are rectangles in the unit square. To this aim let us consider a cylinder of length $n + l + 1$ at time $-l$: $[a_{i_1}, \ldots, a_{i_{n+l+1}}]_{-l} = \{s \in A^{\mathbb{Z}} | s_{-l} = a_{i_1}, s_{-l+1} = a_{i_2}, \ldots, s_0 = a_{i_{l+1}}, s_1 = a_{i_{l+2}}, \ldots, s_n = a_{i_{n+l+1}}\}$. We split the elements of this set into two half-sequences: $s_- = \ldots s_{-l}s_{-l+1}\ldots s_0$ and $s_+ = s_1 s_2 \ldots s_n \ldots$. According to our discussion in Sec. 2.3 we therefore obtain an interval of real numbers for the expansion of s_- and s_+, respectively. The infimum of the interval constituted by the expansions of s_- is

$$x_1 = \sum_{i=0}^{-l} g(s_i)I^{i-1}\,. \tag{22}$$

For the supremum of this interval we find

$$x_2 = x_1 + \sum_{i=l+2}^{\infty} (I-1)I^{-i} = x_1 + I^{-l-1} \tag{23}$$

where we have made use of the limit of geometric series thus proving Eq. (7). Accordingly, we obtain the interval $[y_1, y_1 + I^{-n}]$ with $y_1 = \sum_{i=1}^{n} g(s_i)I^{-i}$ for the right-half sequence s_+. Hence, the cylinder set $[a_{i_1}, \ldots, a_{i_{n+l+1}}]_{-l}$ is represented by the rectangle $[x_1, x_2] \times [y_1, y_2]$ [cf. Eq. (8)].

3.3. *Local ambiguity of context-free grammars*

In this section, we wish to investigate the properties of locally ambiguous grammars as informally presented in Sec. 2.1 in terms of some standard notions from the theory of formal languages. In particular, we are interested in constructing a deterministic top-down recognizer from an arbitrary locally unambiguous grammar, as described in Sec. 2.2.

We will then go on to show that any context-free grammar may be partitioned into a finite set of locally unambiguous types of grammar, as suggested in Sec. 2.1. We begin by presenting some basic concepts from the theory of formal languages.

A context-free grammar G is defined as a 4-tuple, $G = (N, T, P, S)$, where

(24) N is a finite set of n *nonterminal symbols*,
(25) T is a finite set of m *terminal symbols* such that $N \cap T = \emptyset$,
(26) $P \subseteq N \rightarrow (N \cup T)^*$ is a finite set of *production rules*,[4] and
(27) $S \in N$ is the distinguished *start symbol*.

Below, unless otherwise specified, we will assume we are dealing with a context-free grammar $G = (N, T, P, S)$.

A *pushdown automaton* (or "PDA") is a 7-tuple $M = (Q, T, \Gamma, \delta, q_0, Z_0, F)$, where

(28) Q is a finite set of *states*,
(29) T is a finite *input alphabet*,
(30) Γ is a finite *stack alphabet*,
(31) δ: $Q \times (T \cup \{\varepsilon\}) \times \Gamma \rightarrow 2^{(Q \times \Gamma^*)}$ is a partial *transition function*,
(32) $q_0 \in Q$ is the distinguished *initial state*,
(33) $Z_0 \in \Gamma$ is the *initial stack symbol*, and
(34) $F \subseteq Q$ is the set of *final states*.

In this paper, we are concerned with pushdown automata with no final states: $F = \emptyset$. Such automata are said to *accept by empty stack*. The operation of such automata is described in more detail in Sec. 3.4. Given a context-free grammar G, we can construct a pushdown automaton which accepts the language generated by G by simulating rule expansions. Such an automaton is called a *top-down recognizer*. Formally, if $G = (N, T, P, S)$ is a context-free grammar, then the pushdown automaton $M = (\{q\}, T, (N \cup T), \delta, q, S, \emptyset)$ is a *top-down recognizer for* G, where δ is defined as follows for all $A \in N$ and all $a \in T$:

$$\delta(q, \varepsilon, A) = \{(q, \alpha) | A \rightarrow \alpha \in P\} \quad (\text{"}\varepsilon \text{ move"}) \tag{35}$$

$$\delta(q, a, a) = \{(q, \varepsilon)\} \quad (\text{"non-}\varepsilon \text{ move"}) . \tag{36}$$

See [Hopcroft & Ullman, 1969; Aho & Ullman, 1972] for a more complete discussion.

Our concern in this paper is with deterministic pushdown automata. Informally, a deterministic automaton is simply one whose transition function allows at most one move for each possible configuration.[5] Formally, a PDA M as above is said to be *deterministic* (a "DPDA") if for each $q \in Q$ and each $Z \in \Gamma$, either (37) or (38) holds for all $a \in T$:

$$\delta(q, \varepsilon, Z) = \emptyset \quad \text{and} \quad \delta(q, a, Z) \text{ contains at most one element} , \tag{37}$$

$$\delta(q, a, Z) = \emptyset \quad \text{and} \quad \delta(q, \varepsilon, Z) \text{ contains at most one element} . \tag{38}$$

When dealing with deterministic PDAs, the transition function may be redefined: δ: $Q \times (T \cup \{\varepsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$.

We now turn our attention to locally ambiguous grammars as informally presented in Sec. 2.1. Formally, we say for a context-free grammar $G = (N, T, P, S)$ and a nonterminal symbol $A \in N$, that G is *locally ambiguous with respect to* A if and only if there exist productions $A \rightarrow \beta$, $A \rightarrow \gamma \in P$ such that $\beta \neq \gamma$. We say that G is *locally ambiguous* just in case there is some nonterminal symbol $A \in N$ such that G is locally ambiguous with respect to A. G is said to be *locally unambiguous* just in case G is not locally ambiguous.

It should be noted that our definition of "local ambiguity" differs from other notions of ambiguity to be found elsewhere in the literature. When considered in terms of top-down recognizers however, our notion can be seen to resemble the concept of local ambiguity for LR parsers with graph-structured stacks described by Tomita [1987]. Traditionally though, "ambiguity" is defined globally for a grammar as that property which holds when some sentence of the grammar has more than one left derivation with respect to that grammar (see Sec. 1). Clearly, a grammar can only be ambiguous in the traditional sense if it is also locally ambiguous. The reverse does not hold, since there are globally unambiguous grammars which are indeed locally ambiguous, such as our toy-grammar from Sec. 2.

Given our notion of local ambiguity, it is easy to see that every locally unambiguous grammar $G = (N, T, P, S)$ has a deterministic top-down

[4] Here and elsewhere in this section, we use the Kleene star to indicate the reflexive and transitive closure of the *alphabetic concatenation* operation; thus V^* is the set of all strings over the alphabet V, including the empty string ε.
[5] A *configuration* for a pushdown automaton is a triple $(q, w, \alpha) \in Q \times T^* \times \Gamma^*$.

recognizer $M = (\{q\}, T, (N \cup T), \delta, q, S, \emptyset)$. Only Eq. (35) in the construction given above is capable of introducing nondeterminism into M. For a locally unambiguous grammar G, there is at most one rule expanding each nonterminal symbol $A \in N$, so that $\delta(q, \varepsilon, A)$ has at most one element, and M is therefore deterministic.

Having shown that all locally unambiguous grammars have deterministic top-down recognizers, we now present a method by which an arbitrary context-free grammar may be broken down into a finite set of locally unambiguous grammars. Let $G = (N, T, P, S)$ be a context-free grammar. We call a finite set of context-free grammars $\mathcal{G} = \{G_1, \ldots, G_p\}$ a *partitioning of* G just in case $P = \bigcup_{i=1}^{p} P_i$ and $G_i = (N, T, P_i, S)$, for $1 \leq i \leq p$. If $\mathcal{G}$ is a partitioning of some context-free grammar G, we call $\mathcal{G}$ a *partitioned context-free grammar*, and say that G is the *synthesis grammar* of $\mathcal{G}$. We wish to use local ambiguity as a criterion on which to base grammar partitionings. For this purpose, we first define a *local disambiguation function* LocDis, which produces for a context-free grammar $G = (N, T, P, S)$ and a nonterminal symbol $A \in N$ unique partitioning $\mathcal{G}$ of G such that each element of $\mathcal{G}$ is locally unambiguous with respect to A. Let $P_A = \{A \to \alpha | A \to \alpha \in P\}$ and let $P_{\overline{A}} = P - P_A$, then

$$\text{LocDis}(A, G) = \begin{cases} \{G\} & \text{if } P_A = \emptyset \\ \bigcup\limits_{r \in P_A} \{(N, T, P_{\overline{A}} \cup \{r\}, S)\} & \text{otherwise}. \end{cases} \tag{39}$$

To see that the elements of $\text{LocDis}(A, G)$ are locally unambiguous with respect to A, we must consider several cases. If G is not locally ambiguous with respect to A, then either P contains no expansions for A, in which case $P_A = \emptyset$, or P contains exactly one rule $A \to \alpha$ expanding A, in which case $P_A = \{A \to \alpha\}$; in both of these cases, $\text{LocDis}(A, G) = \{G\}$. Otherwise, G is locally ambiguous with respect to A, so there are rules $A \to \alpha_1, \ldots, A \to \alpha_q \in P$, and $P_A = \{A \to \alpha_1, \ldots, A \to \alpha_q\}$. Since P_A is nonempty, each $G' \in \text{LocDis}(A, G)$ contains exactly one rule from P_A, and is therefore locally unambiguous with respect to A.

We can extend the function LocDis to handle a partitioned context-free grammar $\mathcal{G}$ as its second argument in the obvious manner:

$$\text{LocDis}(A, \mathcal{G}) = \bigcup_{G' \in \mathcal{G}} \text{LocDis}(A, G'). \tag{40}$$

By iteratively applying the extended local disambiguation function LocDis, it is possible to partition an arbitrary context-free grammar $G = (N, T, P, S)$ into a finite set $\mathcal{G}$ of context-free grammars such that each $G' \in \mathcal{G}$ is locally unambiguous. We say in this case that $\mathcal{G}$ is a *locally unambiguous partitioning of* G. Let $\{A_1, \ldots, A_n\} = N$ be an enumeration of the nonterminal symbols of the grammar G, and construct the partitioning $\mathcal{G}_n$ of G:

$$\mathcal{G}_0 = \{G\}$$
$$\mathcal{G}_{i+1} = \text{LocDis}(A_{i+1}, \mathcal{G}_i).$$

As shown above, each iteration in the construction of $\mathcal{G}_n$ removes the local ambiguities with respect to a single nonterminal symbol from the partitioned grammar produced by the preceding iteration. Further, no new local ambiguities are introduced by any application of LocDis, since LocDis is monotonic in the sense that it can only remove rules from an argument grammar. Since G has exactly n nonterminals, every element of $\mathcal{G}_n$ must be locally unambiguous.

We have shown that every context-free grammar can be partitioned into a finite set of locally unambiguous grammars, and that each element of this set is associated with a deterministic top-down recognizer. Although we will concern ourselves in the rest of this paper with modeling the recognizers associated with locally unambiguous grammars, the use of a control parameter as described in Sec. 2.5 should allow the extension of our modeling strategy to the recognition of arbitrary context-free languages by means of the locally unambiguous partitionings of the respective grammars.

3.4. *Symbolic dynamics of automata*

In order to describe Turing machines as dynamical systems, Moore [1990, 1991b] introduced the concept of *generalized shifts*. Like the shifts discussed in Sec. 3.2 the generalized shifts are maps defined on the set of bi-infinite symbolic strings $A^{\mathbb{Z}}$. Let $s = \ldots a_{i_{-1}} \hat{a_{i_0}} a_{i_1} a_{i_2} \ldots$ be such a sequence where the hat denotes the current state. In Moore's

construction the hat indicates the tape position of the head of a Turing machine. A generalized shift is characterized by a range of d symbols called *domain of dependence*. The generalized shift acts on this word w of length d by first replacing it by a further word w' of length d' and then by performing a shift k symbols to the left or to the right [Moore, 1990, 1991b; Badii & Politi, 1997]. Moore has proven that these systems are computationally equivalent to any Turing machine.

The b-adic expansion algorithm of symbolic sequences into points of the unit square [Eqs. (20) and (21)] can be applied to states of the generalized shift in quite the same manner. It is therefore possible to map any Turing machine onto a time discrete nonlinear dynamics living at the unit square. The corresponding map has been shown to be piecewise affine linear just as the Bernoulli map discussed in Sec. 3.1 [Moore, 1990, 1991b].

In this section we shall justify our construction of dynamical systems from top-down parsers for the processing of context-free languages supplied in Sec. 2.4. Because formal languages accepted by pushdown automata belong to a lower class in the Chomsky hierarchy than recursively enumerable languages accepted by Turing machines [Hopcroft & Ullmann, 1979; Badii & Politi, 1997], any pushdown automaton can be simulated by a Turing machine. Hence, Moore's proof holds for pushdown automata too.[6]

Here, we shall assume G to be locally unambiguous, belonging, e.g. to a locally unambiguous partitioning of some locally ambiguous grammar which can be constructed by the algorithm described in Sec. 3.3. Then, as we have also demonstrated in Sec. 3.3, the language $\mathcal{L}(G)$ can be processed by a deterministic top-down recognizer $M = (\{q\}, T, \Gamma, \delta, q, S, \emptyset)$ with $\Gamma = N \cup T$. The state descriptions of this automaton are provided by the actual pairs $(\gamma, w) \in \Gamma^* \times T^*$ where $\gamma = \gamma_{i_0}\gamma_{i_1}\dots\gamma_{i_{k-1}}$ is the content of the parser's stack while $w = w_{j_1}w_{j_2}\dots w_{j_l}$ is some finite word at the input tape. From the definition of the transition function δ follows that the automaton has access only to the top of the stack γ_{i_0} and to the first symbol of the input tape w_{j_1} at each instance of time. Thus, we define a map $\tau\colon \Gamma \times T \to \Gamma^* \times T^*$ acting on actual pairs of the topmost symbols of stack $Z \in \Gamma$ and input tape $a \in T$, respectively, by the transition function δ

$$\begin{aligned} \tau(Z, a) &= (\gamma, a): \quad \delta(q, \varepsilon, Z) = (q, \gamma)\,, \\ &\qquad \text{for } \varepsilon \text{ moves} \\ \tau(a, a) &= (\varepsilon, \varepsilon): \quad \delta(q, a, a) = (q, \varepsilon)\,, \\ &\qquad \text{for non-}\varepsilon \text{ moves}\,, \end{aligned} \tag{41}$$

where $\gamma = \gamma_{i_0}\dots\gamma_{i_{k-1}}$ if $Z \in N$ and if P contains a rule $Z \to \gamma$. We see that the ε-moves correspond to the *predict* operations while the non-ε moves correspond to the *attach* operations described in Sec. 2.2.

In the following we establish the relation between deterministic top-down parsers and piecewise affine linear maps at the unit square that has already been used in Sec. 2.4 for the construction of our toy-model of language processing. For this aim we shall first reorder the content of the stack: $\gamma'_{i_{-k}} = \gamma_{i_k}$, obtaining the reversed sequence $\gamma' = \gamma'_{i_{-k+1}}\dots\gamma'_{i_{-1}}\gamma'_{i_0}$. Next, we concatenate the transformed stack with the input tape yielding a two-sided (but finite) string

$$s = \gamma'_{i_{-k+1}}\dots\gamma'_{i_{-1}}\gamma'_{i_0}w_{j_1}w_{j_2}\dots w_{j_l}\,. \tag{42}$$

Then, we could apply the b-adic expansion on the left- and right-half sequences s_- and s_+ after introducing a Gödel encoding of the set $\Gamma = N \cup T$ in order to map the actual pair to a point in the unit square. But this is not what we will do. To avoid gaps in the unit square we decided to use two different encodings: one, g_1, of the set Γ for the stack and another, g_2, of the terminals T for the input tape. Thus, we perform a $b_1 = (n+m)$-adic expansion of the stack and a $b_2 = m$-adic expansion for the input tape. This yields a partition of the unit square into rectangles of equal size $(n+m)^{-1} \times m^{-1}$ according to Eqs. (20) and (21). The two b-adic expansions lead to a point (x, y) in the unit square with coordinates

$$x = \sum_{h=0}^{k-1} g_1(\gamma'_{i_{-h}})(m+n)^{-h-1} \tag{43}$$

$$y = \sum_{h=1}^{l} g_2(w_{j_h})m^{-h}\,. \tag{44}$$

By this construction the most significant digits γ'_{i_0} and w_{j_1} determine the rectangle where the point (x, y) will be found. We obtained therefore a coarse-grained description of the actual pairs represented by states of a quantitative dynamical system.

[6]Moore has also proven that pushdown automata are equivalent to nondeterministic one-sided generalized shifts [Moore, 1991a, 1991b]. However we do not pursue this approach here since we are interested in deterministic dynamical systems.

To proceed further, we need some consideration of the *empty word* ε. In formal language theory the set A^* of all strings of words of finite length formed by letters of the alphabet A constitutes a semi-group with respect to the concatenation "$\cdot$" of words: $u \cdot (v \cdot w) = (u \cdot v) \cdot w$ for all u, v, $w \in A^*$ and $w \cdot \varepsilon = \varepsilon \cdot w = w$. The empty word ε is the neutral element of this semi-group. In order to complete our construction we have to determine the image of ε in the unit interval when we consider one-sided sequences $s \in A^{\mathbb{N}}$. This can be done by using Eq. (19) which states the relation between cylinder sets of length n and their preimages in the phase space of a dynamical system. Decomposing the left-hand side of Eq. (19) into concatenation and the right-hand side into intersection factors leads to

$$\pi^{-1}([(a_{i_1}, \ldots, a_{i_m}) \cdot (a_{i_{m+1}}, \ldots, a_{i_n})]_1) = \left(\bigcap_{k=0}^{m-1} \Phi^{-k}(A_{i_{k+1}})\right) \cap \left(\bigcap_{k=m}^{n-1} \Phi^{-k}(A_{i_{k+1}})\right) \tag{45}$$

or in shorthand notation $\pi^{-1}(u \cdot v) = \pi^{-1}(u) \cap \pi^{-1}(v)$ for words u, v regarded as cylinder sets. Hence, π^{-1} is a semi-group homomorphism mapping the concatenation of words onto the intersection of sets. It is well known from set theory that the neutral element with respect to set intersection is the base set. In our case, this is just the unit interval $[0, 1]$ which has therefore to be identified with the empty word. Correspondingly, the goal of the top-down parser, the actual pair $(\varepsilon, \varepsilon)$ is represented by the whole unit square $[0, 1]^2$. A further consequence of this reasoning is that the content of the stack as well as the content of the input tape might be considered as one-sided infinite strings that are committed to finite words at the very beginning as we have argued in Sec. 2.4. Due to this interpretation we allow uncertainty about forthcoming and already processed input, respectively. Now, we are able to extend the symbol sequence from Eq. (42) towards plus and minus infinity:

$$s = \ldots \gamma'_{i_{-k+1}} \ldots \gamma'_{i_{-1}} \gamma'_{i_0} w_{j_1} w_{j_2} \ldots w_{j_l} \ldots . \tag{46}$$

The actual pair (γ, w) of the parser comes out to be the set of all bi-infinite sequences s coinciding in the symbols $\gamma'_{i_{-k+1}} \ldots \gamma'_{i_{-1}} \gamma'_{i_0}$ at the left-half side and coinciding in the symbols $w_{j_1} w_{j_2} \ldots w_{j_l}$ at the right-half side, that is — a cylinder set. Using the b-adic expansions [Eqs. (43) and (44)] of the parser's state, any actual pair is mapped onto a rectangle in the unit square.

Our remaining job is to establish the parsers dynamics by a generalized shift and finally by a piecewise affine linear map at the unit square. This is achieved by the definition of the map τ introduced by Eq. (41). Since τ acts only on actual pairs of symbols (Z, a) and not of strings, the Gödel numbers of these symbols provide the domains of definition in Moore's construction with $d = 2$. Considering Z and a as the most significant digits of the b-adic representations of stack and input tape, they define a partition of the unit square into basic rectangles with labels $i = g_1(Z)$, $j = g_2(a)$. We therefore construct a map $\Phi : [0, 1]^2 \to [0, 1]^2$ which is affine linear at these rectangles (i, j) thus yielding Eq. (12).

4. Discussion

We have presented a formal approach for modeling language processing by nonlinear dynamical systems. This has been illustrated by a toy-model of the psycholinguistic experiment using event-related brain potentials and event-related cylinder entropies reported by Frisch *et al.* [2004]. In this experiment, it was tested how the syntactic ambiguity of a pronoun in German is resolved in the course of sentence processing. For a proper discussion, see Frisch *et al.* [2004]. The construction of our model comprises eight steps: (1) describing the stimulus material of the experiment by a rather crude formal language, hereby using the grammatical roles (subject and object) of noun phrases; (2) representing the cognitive conflicts by a locally ambiguous context-free grammar generating this language; (3) decomposing the grammar into unambiguous parts, with each subgrammar being interpreted as one processing strategy; (4) constructing appropriate pushdown automata as recognizers of these sublanguages, the parsing process is represented by a "trajectory" of actual pairs; (5) employing a Gödel encoding on the stack and the input tape of the automata separately, thus mapping the parser onto a piecewise affine linear function at the unit square due to Moore's proof [Moore, 1990]; (6) merging the different parsing maps corresponding to the different processing strategies into one dynamical system depending on a control parameter, and including "repair" maps for intermediate values of this; (7) preparing randomly an ensemble of points in the unit square within the initial actual pair of the parser as initial conditions, iterating the map for obtaining sets of trajectories that represent further

states of the parser, invariant sets correspond to garden path interpretations, then bifurcations serve as conflict resolutions; (8) determining the entropy of the parsing states with respect to a measurement partition as a model of event-related cylinder entropies.

The construction and formal justification of the model essentially relies on findings of computational linguistics, automata theory and dynamical system theory, especially on symbolic dynamics. Of course, we are fully aware that our model cannot seriously explain findings from psycholinguistics or cognitive science — as yet. There are some problems and difficulties on the one hand, but on the other hand, given one initial success, our modeling approach might be worth pursuing further. We shall discuss some of these issues and their possible solutions in the concluding paragraphs.

Firstly, we address the psycholinguistic requirements of more realistic modeling. One point is that context-free grammars are not well suited to describe natural languages [Shieber, 1985]. However, natural languages share many properties with context-free languages, chiefly that they can be described by hierarchical tree structures (see [Saddy & Uriagereka, 2004] and [Frisch *et al.*, 2004]). Thus, even if natural languages were not context-free, this might hold for certain subsets of speech. The trees discussed by Frisch *et al.* [2004] serve as an good example: they describe the linguistic properties of the stimulus material of the ERP experiment in the framework of Chomsky's *Government and Binding* (*GB*) theory [Haegeman, 1994] that is also commonly used by researchers in the field of psycholinguistics. Nevertheless, once the trees have been written down, one could forget everything about GB theory and sketch a context-free toy-grammar using only the labels on the nodes of GB trees to identify the production rules of the grammar. This would be the first attempt at the construction of a slightly more realistic model which we are working on.

We have used very simple parsing devices, namely deterministic top-down recognizers. Though there is linguistic evidence that the human parser is deterministic (otherwise garden path effects would not appear), it does not pursue a *top-down strategy*, but rather follows a *left-corner strategy*[7] [Hemforth, 1993]. Staudacher [1990] has shown that the psycholinguistically motivated *Marcus-parser* [Marcus, 1980] is in fact a deterministic left-corner parser. It is therefore desirable to consider these automata for appropriate modeling.

In Sec. 3.4 we pointed out that Moore's proof of the formal equivalence of automata and nonlinear dynamical systems at the unit square holds not only for simple top-down parsers but for any Turing machine. Adopting the Church–Turing thesis that any possible computation can be achieved by a Turing machine [Hopcroft & Ullman, 1979; Hofstadter, 1979], and further adopting that parsing is computation, we conclude that the model's present lack of cognitive appropriateness is not a fundamental objection, because one could imagine a natural language parser emulated by a Turing machine which can then be mapped onto a dynamical system using Moore's construction. However, Turing machines are not really interesting for computational linguistics. One reason for this is their freely accessible memory tape, whereas cognitive scientists are dealing with limited capacities of the human computational system [Mitchell, 1994]. Good candidates for natural language grammars seem to be, e.g. the so-called *extended right linear indexed grammars* [Michaelis & Wartena, 1999] which allow for dependencies across the structure trees. Although the languages generated by such grammars do belong to a higher class of the Chomsky hierarchy,[8] they could be processed by automata with less power than Turing machines since they are highly restricted context-sensitive languages [Michaelis & Wartena, 1999] that must be processed using tape-limited automata. Provided the state descriptions and the machine tables of these automata were available, one could use the idea of Moore's proof again to build a nonlinear dynamics from such automata. We are currently investigating these systems and their relations to Moore's generalized shifts.

From a psycholinguistic perspective, there is evidence that the human parser acknowledges the

[7]In this paper we have dealt only with predicting top-down parsers. There are other parsing strategies known, such as *bottom-up* parsing that is data driven or even *left-corner* parsing which is a mixture of a top-down and a bottom-up parser [Aho & Ullman, 1972].

[8]The Chomsky hierarchy is a system of formal languages classified by the computational complexity of the automata required to process them [Hopcroft & Ullmann, 1979].

presence of syntactic alternatives, in that syntactically ambiguous arguments take longer to read than their unambiguous counterparts [Schlesewsky *et al.*, 2000]. In the ERP, this enhanced cost of processing ambiguous arguments is reflected in a P600 component [Frisch *et al.*, 2002]. This finding is crucial for theories of language processing insofar as it does not support serial parsing, that is, the assumption that the parser initially computes only the structurally simplest analysis independently of any alternative reading (cf. [Mitchell, 1994]). The presence of an alternative reading should therefore not induce enhanced processing cost at the ambiguous items itself, but should come into play only if the initial preference is incompatible with forthcoming, unambiguous information. That an ambiguous argument *is* more difficult to process than an unambiguous one shows that the parser takes the possibility of several possible analysis into account from the beginning. However, it was consistently found that later, namely on the *disambiguating* element, not all continuations are equally easy to process, but that disambiguation towards object-before-subject are more difficult to process compared to subject-before-object (see above; [Frisch *et al.*, 2002, 2004; Schlesewsky *et al.*, 2000]). This shows that the ambiguity is of course acknowledged but that, after that, the parser prefers one alternative, here, subject-before-object, to the expense of others.

The paper of Frisch *et al.* [2004] tested only the effects on the disambiguating element, not in the ambiguous region. Nevertheless, the parsing device developed in the present paper should be able to account for the P600 response due to a syntactic ambiguity found by Frisch *et al.* [2002] in order to approach psychological reality. How could this be accomplished? In the model presented here, a cognitive control unit intervenes into the parsing dynamics by tuning the control parameter in order to account for which rules of the toy-grammar are chosen initially or for reanalysis. This intervention differs from the automatic processes of deterministic parsing when a particular rule has to be applied in order to predict the next word. Therefore, the intervention must be regarded as more controlled than parsing. In the case of an unambiguous argument, tuning the control parameter is cognitively less costly (since there is only one possibility) compared to an ambiguous argument. It is plausible that in the latter case, the cognitive control unit acknowledges that more than one possibilities exist and that a choice has to be made. Assuming that this choice consumes additional cognitive resources could account for the finding of the P600 ambiguity effect in the study of Frisch *et al.* [2002], seeing that the P600 was shown to reflect controlled processing [Hahne & Friederici, 1999].

This last assumption points to another issue that our approach to modeling highlights, that is, how to associate behavioral makers with changes of differences in cognitive processing load. It is the established tradition in cognitive psychology and related disciplines that time equals cost. The more difficult or complex a process is the more time it will take. In the ERP literature the elicitation of components is interpreted as reflecting differential processing loads between one experimental condition and another. In our dynamical model of parsing, transitions of the parser are mapped onto changes in phase space as determined by the calculation of cylinder entropies. For the sake of the toy-grammar we have restricted ourselves to the change in phase space associated with the P600 voltage averaged component elicited under so-called reanalysis conditions. However, the P600 and related positivities can be elicited by a variety of experimental conditions that are not obviously closely related in processing terms [Frisch *et al.*, 2002]. A precise understanding of the relation between the ERP components and the variety of conditions that can elicit them is still an active research goal. Interestingly, the calculation of cylinder entropies associated with experimental conditions standardly reveal more changes in phase space than there are elicited voltage averaged components. That is to say that changes in phase space that do not correspond to ERP components do occur. At the present there is no good theory of cognitive behavior that predicts such contrasts. Saddy and beim Graben [2002] note that the observed phase space changes occur in the time ranges that match well with Friederici's model [Friederici, 1999] and offer a broad interpretation of such changes in system level terms. One obvious route to pursue in hopes of gaining insight into the distribution and nature of the entire suite ERP components and phase space alternations associated with psycholinguistic manipulations is the development and extension of computational models such as those presented here.

The parsers we are dealing with are simple *recognizers*. Such automata reach their accepting

state when both, the stack and the input are completely discarded. Unfortunately, one loses all information about the processed input and about its structural representation. This is a problem with our model, insofar as a structural representation of the input is necessary for further linguistic and cognitive analysis, e.g. for semantic and pragmatic understanding. A structural representation is also indispensable for psycholinguistic modeling since reanalysis implies changes along the sentence structures that have been built up during the parsing process. We are developing such automata generating and storing a left-derivation of the input in the stack memory. In order to achieve this, one has to introduce a "stack pointer" with random access to a limited part of the stack. This will lead to generalized shifts with larger domains of definition in the concatenation of stack and input tape. We will also be able to address the "costs of reanalysis" in such models when the repair maps need access to deeper parts of the structures in the stack thus increasing the complexity of the flow at the unit square. And finally, such devices would agree much better with the essential ideas of symbolic dynamics: namely that dynamics generates information (expressed by the Kolmogorov–Sinai entropy of the system) during transient evolution. This is also compatible with the dynamic understanding of information in the sense of von Weizsäcker [1974, p. 352]: "Information is only what generates information" (see also the contribution of Jirsa [2004]).

The parameterized maps emulating the parsers are deterministic dynamical systems whose behavior is completely determined by the initial conditions. These encode, by construction, the strings to be processed. Hence, any garden path analysis caused by applying the wrong strategy (an inappropriate value of the control parameter) is principally predictable at the very beginning of the dynamics. This is not consistent with psycholinguistic reasoning. It is also inconsistent with the design of psycholinguistic experiments where stimuli are presented item-for-item. We are working on a model that better fits the reality by splitting the input tape into a space limited "working memory" belonging to the automaton and a second part belonging to the "outer world" that is not accessible by the model. From this outer world tape new symbols are read in by the system after each attach operation. Thus, the experimental stimulation is simulated by a forced dynamical system which is disturbed in its evolution by the external input.

There are many other questions raised by our approach. From the natural science point of view: How might a time discrete piecewise affine linear map at the unit square be implemented by living neural networks? How to construe the connection to the measured time continuous ERP data? Let us outline a possible solution we are still pursuing. The discrete time map at the unit square can be regarded as a Poincaré section of a three-dimensional flow. Then it follows from the properties of the parsing map that its invariant sets are sets of limit cycle trajectories. That is, the appropriate continuous time dynamics of the embedded parser are nonlinear oscillators [Guckenheimer & Holmes, 1983]. And it is well known from neural modeling that limit cycle oscillators can indeed be constituted by neural networks [Wilson & Cowan, 1972; Freeman, 2000; Nunez, 1995]. Provided that such an oscillator model of the parsing dynamics is possible, it would also entail an interface for the analysis of real time EEG and ERP measurements we are investigating by using symbolic dynamics.

There is evidence i.e. from the results of Hutt [2004] that ERP components are fixed points of the brain dynamics. If this were generally true, it should be possible to find a two-dimensional embedding of ERP time series where ERP components are separated clusters in the phase space. Then one might attempt to capture these clusters by the cells of a partition and to examine the symbolic dynamics thus obtained. The challenging question is then whether this symbolic dynamics can be interpreted in terms of automata theory.

Although our parsing dynamics is a function that maps points in the unit square onto other points in the unit square, we considered ensembles of these states in order to interpret them as actual pairs of the parser. This appears to be cognitively and physiologically not very plausible since one would expect that a brain state is a point in its phase space and not an ensemble of randomly distributed states. A possible solution for this problem could be provided by stochastic dynamics: One prepares the parser with only *one* randomly chosen initial condition belonging to that rectangle which corresponds to the first actual pair. That is, one initializes the dynamics with a delta-shaped probability distribution function. Due to

the impact of the stochastic forces, the time evolution of the initial probability distribution function is governed by a Fokker–Planck equation. Then, the initial probability distribution will spread out over the unit square when the system evolves thus leading to an ensemble interpretation of the dynamics as desired. However, one has to ensure that the stochastic forces are agreeable to the partition of the phase space for maintaining the parsing interpretation.

Finally let us address a philosophical aspect of our model. In Sec. 2.6 we computed the information content of the states of the parser with respect to an arbitrary measurement partition. That is a crucial point of symbolic dynamics and information theory: there are many ways of partitioning the phase space of a dynamical system (unless a *generating partition* exists); and as C. F. von Weizsäcker states: "An 'absolute' notion of information does not make any sense; information exists only with respect to another notion, namely relative to two 'semantic levels'" [von Weizsäcker, 1988, p. 172]. "Semantic levels" in the sense of von Weizsäcker are, e.g. *macro-* and *microstates* of a dynamical system, where microstates are considered as points in the phase space, whereas macrostates are equivalence classes of points all leading to the same measurement of an observable, such as the energy for the microcanonical ensemble. That is, macrostates provide a partition of the system's state space and hence a symbolic dynamics. On the other hand, two partitions of the state space may be complementary in the sense of Bohr's *Principle of Complementarity* in quantum physics [Bohr, 1948; Atmanspacher *et al.*, 2002; Atmanspacher, 2003]. The relative arbitrariness of a symbolic dynamics has a strong impact on the interpretation of our model. Consider once more the unit square in Fig. 1 without knowing its partition into the six rectangles which establish the domains of definition of the parser's map due to the Gödel encoding. By simulating the deterministic dynamics of the system there are only clouds of points hopping through the unit square at the lower "semantic level". However, after introducing the particular partition we have used in our construction, the dynamics of the system becomes interpretable as language processing at the higher "semantic level", i.e. as cognitive computation. We shall discuss this issue more thoroughly in a forthcoming paper.

Acknowledgments

This work has been supported by the Deutsche Forschungsgemeinschaft (research group "conflicting rules in cognitive systems"). We want to thank Peter Staudacher, Jens Michaelis, Günter Troll, Harald Atmanspacher, Udo Schwarz and Viktor Jirsa for stimulating and helpful discussions. Bryan Jurish acknowledges support from the project "Collocations in the German Language of the 20th Century" of the Berlin-Brandenburg Academy of Sciences.

References

Aho, A. V. & Ullman, J. D. [1972] *The Theory of Parsing, Translation and Compiling Vol. I: Parsing.* (Prentice Hall Englewood Cliffs, NJ).

Allen, J. [1987] *Natural Language Understanding. Benjamin/Cummings Series in Computer Science* (Benjamin/Cummings Publishing Company, Menlo Park, CA).

Anosov, D. V. & Arnol'd, V. I. (eds.) [1988] *Dynamical Systems Encyclopaedia of Math. Sciences*, Vol. 1 (Springer, Berlin).

Atmanspacher, H., Römer, H. & Walach, H. [2002] "Weak quantum theory: Complementarity and entanglement in physics and beyond," *Found. Phys.* **32**, 379–406.

Atmanspacher, H. [2003] "Mind and matter as asymptotically disjoint, inequivalent representations with broken time-reversal symmetry," *BioSystems* **68**, 19–30.

Başar, E. [1980] *EEG–Brain Dynamics. Relations between EEG and Brain Evoked Potentials* (Elsevier/ North Holland Biomedical Press Amsterdam).

Badii, R. & Politi, A. [1997] *Complexity. Hierarchical Structures and Scaling in Physics*, Cambridge Nonlinear Science Series, Vol. 6 (Cambridge University Press Cambridge, UK).

Beck, C. & Schlögl, F. [1993] *Thermodynamics of Chaotic Systems. An Introduction*, Cambridge Nonlinear Science Series, Vol. 4 (Cambridge University Press Cambridge (UK)).

beim Graben, P., Liebscher, T. & Saddy, J. D. [2000a] "Parsing ambiguous context-free languages by dynamical systems: Disambiguation and phase transitions in neural networks with evidence from event-related brain potentials (ERP)," in *Learning to Behave*, eds. Jokinen, K., Heylen, D. & Njiholt, A., Internalising Knowledge of TWLT 18, Vol. II (Enschede Universiteit Twente), pp. 119–135.

beim Graben, P., Saddy, J. D., Schlesewsky, M. & Kurths, J. [2000b] "Symbolic dynamics of event–related brain potentials," *Phys. Rev.* **E62**, 5518–5541.

Bennett, C. H. [1982] "The thermodynamics of computation — a review," *Int. J. Th. Phys.* **21**, 907–940.

Bohr, N. [1948] "On the notions of causality and complementarity," *Dialectica* **2**, 312–319.

Collet, P. & Eckmann, J.-P. [1980] *Iterated Maps on the Interval as Dynamical Systems* (Birkhauser Boston).

Crutchfield, J. P. [1994] "The calculi of emergence: Computation, dynamics and induction," *Physica* **D75**, 11–54.

Engbert, R., Scheffczyk, C., Krampe, R. T., Rosenblum, M., Kurths, J. & Kliegl, R. [1997] "Tempo-induced transitions in polyrhythmic hand movements," *Phys. Rev.* **E56**, 5823–5833.

Fodor, J. D. & Ferreira, F. (eds.) [1999] *Reanalyis in Sentence Processing* (Kluwer Dordrecht).

Freeman, W. J. [2000] *Neurodynamics*: *An Exploration in Mesoscopic Brain Dynamics. Perspectives in Neural Computing* (Springer Verlag London).

Friederici, A. D. [1999] "The neurobiology of language comprehension," *Language Comprehension*: *A Biological Perspective*, ed. Friederici, A. D. (Springer Berlin), pp. 265–304.

Frisch, S., beim Graben, P. & Schlesewsky, M. [2004] "Parallelizing grammatical functions: P600 and P345 reflect different cost of reanalysis," *Int. J. Bifurcation and Chaos* **14**, 531–549.

Frisch, S., Schlesewsky, M., Saddy, D. & Alpermann, A. [2002] "The P600 as an indicator of syntactic ambiguity," *Cognition* **85**, B83–B92.

Gödel, K. [1931] "Über formal unentscheidbare Sätze der *principia mathematica* und verwandter Systeme I," *Monatshefte für Mathematik und Physik* **38**, 173–198.

Guckenheimer, J. & Holmes, P. [1983] *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, Springer Series of Applied Mathematical Sciences, Vol. 42 (Springer, NY).

Haegeman, L. [1994] *Introduction to Goverment & Binding Theory*, Blackwell Textbooks in Linguistics, Vol. 1 (Blackwell Publishers, Oxford).

Hahne, A. & Friederici, A. D. [1999] "Electrophysiological evidence for two steps in syntactic analysis: Early automatic and late controlled processes," *J. Cogn. Neurosci.* **11**, 194–205.

Hemforth, B. [1993] *Kognitives Parsing*: *Repräsentation und Verarbeitung kognitiven Wissens* (Infix Sankt Augustin).

Hofstadter, D. R. [1979] *Gödel*, *Escher*, *Bach*: *an Eternal Golden Braid.* (Basic Books, NY).

Hopcroft, J. E. & Ullman, J. D. [1969] *Formal Languages and Their Relation to Automata* (Addison-Wesley Reading, MA).

Hopcroft, J. E. & Ullmann, J. D. [1979] *Introduction to Automata Theory*, *Languages*, *and Computation* (Addison-Wesley Menlo Park, California).

Hutt, A. [2004] "An analytical framework for modeling evoked and event-related potentials," *Int. J. Bifurcation and Chaos* **14**, 653–666.

Jirsa, V. K. [2004] "Information processing in brain and behavior displayed in large-scale scalp topographies such as EEG and MEG," *Int. J. Bifurcation and Chaos* **14**, 679–692.

Kawamoto, A. H. [1993] "Nonlinear dynamics in the resolution of lexical ambiguity: A parallel distributed processing account," *J. Memory and Language* **32**, 474–516.

Kelso, J. A. S., Bressler, S. L., Buchanan, S., DeGuzman, G. C., Ding, M., Fuchs, A. & Holroyd, T. [1992] "A phase transition in human brain and behavior," *Phys. Lett.* **A196**, 134–144.

Marcus, G. F. [2001] *The Algebraic Mind. Integrating Connectionism and Cognitive Science. Learning*, *Delvelopment*, *and Conceptual Change* (MIT Press, Cambridge, MA).

Marcus, M. [1980] *A Theory of Syntactic Recognition for Natural Language* (MIT Press, Cambrigde, MA).

Michaelis, J. & Wartena, C. [1999] "LIGs with reduced derivation sets," *Constraints and Resources in Natural Language Syntax and Semantics*, eds. Bouma, G., Kruijff, G.-J., Hinrichs, E. & Oehrle, R. T., Studies in Constrained Based Lexicalism, Vol. II (CSLI Publications Stanford, CA), pp. 263–279.

Mitchell, D. C. [1994] "Sentence parsing," *Handbook of Psycholinguistics*, ed. Gernsbacher, M. A. (Academic Press San Diego), pp. 375–409.

Moore, C. [1990] "Unpredictability and undecidability in dynamical systems," *Phys. Rev. Lett.* **64**, 2354–2357.

Moore, C. [1991a] "Generalized one-sided shifts and maps of the interval," *Nonlinearity* **4**, 727–745.

Moore, C. [1991b] "Generalized shifts: Unpredictability and undecidability in dynamical systems," *Nonlinearity* **4**, 199–230.

Nunez, P. L. (ed.) [1995] *Neocortical Dynamics and Human EEG Rhythms* (Oxford University Press, NY).

Ott, E. [1993] *Chaos in Dynamical Systems* (Cambridge University Press, NY).

Raçzasek, J., Tuller, B., Shapiro, L. P., Case, P. & Kelso, S. [1999] "Categorization of ambiguous sentences as a function of a changing prosodic parameter: A dynamical approach," *J. Psycholing. Res.* **28**, 367–393.

Saddy, D. & Uriagereka, J. [2004] "Measuring language," *Int. J. Bifurcation and Chaos* **14**, 383–404.

Saddy, J. D. & beim Graben, P. [2002] "Measuring the neural dynamics of language comprehension processes," *Basic Fuctions of Language*, *Reading and Reading Disorder*, eds. Witruk, Friederici, A. D. & Lachmann, T. (Kluwer Academic Publishers Boston), pp. 41–60.

Schlesewsky, M., Fanselow, G., Kliegl, R. & Krems, J. [2000] "Preferences for grammatical functions in the

processing of locally ambigous wh-questions in German," *Cognitive Parsing in German*, eds. Hemforth, B. & Konieczny, L. (Kluwer Dordrecht), pp. 65–93.

Schuster, H. G. [1989] *Deterministic Chaos* (VCH Weinheim).

Shannon, C. E. & Weaver, W. [1949] *The Mathematical Theory of Communication* (University of Illinois Press Urbana).

Shieber, S. M. [1985] "Evidence against the context-freeness of natural language," *Ling. Philos.* **8**, 333–343.

Staudacher, P. [1990] "Ansätze und Probleme prinzipienorientierten Parsens," *Sprache und Wissen*, eds. Felix, S. W., Kanngießer, S. & Rickheit, G. (Westdeutscher Verlag Opladen), pp. 151–189.

Tomita, M. [1987] "An efficient augmented-context-free parsing algorithm," *Comput. Ling.* **13**, 31–46.

van Gelder, T. [1998] "The dynamical hypothesis in cognitive science," *Behav. Brain Sci.* **21**, 615–628.

von Weizsäcker, C. F. [1988] *Aufbau der Physik* (DTV München).

von Weizsäcker, C. F. [1974] *Die Einheit der Natur* (DTV München).

Wilson, H. R. & Cowan, J. D. [1972] "Excitatory and inhibitory interactions in localized populations of model neurons," *Biophys. J.* **12**, 1–24.

Wright, J., Rennie, C., Lees, G., Robinson, P., Bourke, P., Chapman, C., Gordon, E. & Rowe, D. [2004] "Simulated electrocortical activity at microscopic, mesoscopic and global scales," *Int. J. Bifurcation and Chaos* **14**, 853–872.