LETTER Communicated by Josh Bongard

State-Dependent Computation Using Coupled Recurrent Networks

Ueli Rutishauser
urut@caltech.edu
Computation and Neural Systems, California Institute of Technology, Pasadena, CA 91225, U.S.A.

Rodney J. Douglas
rjd@ini.phys.ethz.ch
Institute of Neuroinformatics, ETH and University of Zurich, Zurich 8057, Switzerland

Although conditional branching between possible behavioral states is a hallmark of intelligent behavior, very little is known about the neuronal mechanisms that support this processing. In a step toward solving this problem, we demonstrate by theoretical analysis and simulation how networks of richly interconnected neurons, such as those observed in the superficial layers of the neocortex, can embed reliable, robust finite state machines. We show how a multistable neuronal network containing a number of states can be created very simply by coupling two recurrent networks whose synaptic weights have been configured for soft winner-take-all (sWTA) performance. These two sWTAs have simple, homogeneous, locally recurrent connectivity except for a small fraction of recurrent cross-connections between them, which are used to embed the required states. This coupling between the maps allows the network to continue to express the current state even after the input that elicited that state is withdrawn. In addition, a small number of transition neurons implement the necessary input-driven transitions between the embedded states. We provide simple rules to systematically design and construct neuronal state machines of this kind. The significance of our finding is that it offers a method whereby the cortex could construct networks supporting a broad range of sophisticated processing by applying only small specializations to the same generic neuronal circuit.

1 Introduction

Quantitative studies of the anatomical connection weights between neurons in cat visual cortex have revealed that one prominent feature of the neocortical circuit of cat visual cortex is the high degree of connectivity between pyramidal cells in the superficial cortical layers (Binzegger,

Douglas, & Martin, 2004). About 30% of all excitatory synapses onto these cells are derived from other superficial pyramids, and most of these connections are short range (arising from source neurons within about 300 μm). In addition to excitatory inputs, these pyramids also receive inhibitory inputs, which constitute about 10% of their synaptic input (Shepherd, Stepanyants, Bureau, Chklovskii, & Svoboda, 2005; Binzegger et al., 2004). Thus, the activation of these excitatory neurons can be strongly affected by both positive and negative feedback loops. In theory, these recurrent circuits exhibit a variety of interesting computations (Abbott, 1994; Douglas, Koch, Mahowald, Martin, & Suarez, 1995; Ben-Yishai, Bar-Or, & Sompolinsky, 1995; Hahnloser, Sarpeshkar, Mahowald, Douglas, & Seung, 2000; Zhaoping, 2001; Pouget, Zhang, Deneve, & Latham, 1998; Machens, Romo, & Brody, 2005; Wang, 2002; Rao, 2004; Hochreiter & Schmidhuber, 1997; Coultrip, Granger, & Lynch, 1992). For example, the soft winner-take-all (sWTA) network is able to selectively enhance one part of its input while suppressing the remainder (Hahnloser, Douglas, Mahowald, & Hepp, 1999; Maass, 2000), and so offers a form of signal restoration between computational stages sought by von Neumann (1958) in his early explorations of brainlike principles of computation.

In theoretical models, the neurons that compose a winner-take-all (WTA) network are usually organized as a simple linear map in which each unit receives excitatory inputs from its neighboring units as well as an inhibitory input that is proportional to the total activity of all units (Dayan & Abbott, 2001; Douglas & Martin, 2007). We use the term *map* to indicate that the elements of the WTA network are not functionally independent. Rather, their activity depends on ordered neighbor connections. The positive feedback effected by the excitory neighbor connections enhances the features of the input that match patterns embedded in the excitatory synaptic weights. The overall strength of the excitatory response is used to suppress outliers via the dynamical inhibitory threshold imposed by the global inhibitory neuron. Thus, the circuit can be seen as imposing an interpretation on an incomplete or noisy input signal by restoring it toward some fundamental activity distribution embedded in its excitatory connections (Hahnloser et al., 2000; Hahnloser, Seung, & Slotine, 2003). This selective amplification can be steered in an attentional-like manner by introducing pointer neurons in the excitatory feedback loop, which bias the WTA so that activity at the preferred location "wins" even if the activity at another competing location is larger (Hahnloser et al., 1999).

One drawback of the sWTA is that it is (for the parameters used here) not hysteretic. The activity of the network relaxes toward zero when the input is removed. However, to perform useful computation, the reaction of a network to the same input should depend on the pattern of previous inputs. Such state-dependent processing requires a hysteretic element that is able to retain a history of previous states. One way to provide this history is by inserting into the network attractors that remain stable in the absence

of input. We show how pairs of sWTA circuits can be configured to provide multiple states, which are different patterns of sustained discharge. These states are stable in the absence of external input, and transitions between states are driven by external signals. This property enables the network to react differently to the same external signal depending on the current state. We demonstrate this property by implementing a neural version of a discrete finite automaton.

A discrete finite automaton (DFA) is a computational device that implements state-dependent processing of strings of input symbols. It comprises a set of states (nodes), a transition function that describes the transitions (edges) between states and their dependence on specific input symbols, and a list of acceptable input symbols. The transition between the current state and one of the allowable next states depends on both the current input symbol and the current state. That is, the processing of input symbols is state dependent. This DFA model can be used to define a language by deciding which input strings of symbols lead to a particular final state (the accept state). The set of all acceptable strings is defined as the language of that DFA. DFAs can implement any language belonging to the class of regular languages (Hopcroft, Motwani, & Ullman, 2000).

In this article, we describe simple rules to systematically construct an arbitrary DFA using nearly identical recurrent maps. The significance of this work is that it offers a method whereby the cortex could achieve a broad range of sophisticated processing by only limited specialization of the same generic neuronal circuit.

2 Results

2.1 Single Recurrent Map. The behavior of the neural state machine depends on the properties of the recurrent map, and so we begin by reviewing these (see also Abbott, 1994; Ben-Yishai et al., 1995; Hahnloser et al., 2000; Dayan & Abbott, 2001; Douglas & Martin, 2007). Our recurrent map $\mathbf{x}$ consists of N neurons with continuous-valued outputs. $N-1$ of these neurons are excitatory ($x_{1..N-1}$), and one, x_N, is inhibitory (see Figure 1). Each excitatory neuron receives excitatory input from itself, its neighbors, and a common inhibitory input (β_1). Each excitatory neuron projects to the inhibitory neuron with strength β_2. The inhibitory neuron does not connect to itself.

For convenience, we choose a firing rate model (Dayan & Abbott, 2001) for the recurrent map neurons. Then, in the simple case of self-excitation of strength α (see Figure 1), the dynamics of each excitatory neuron i on the map is given by

$$\tau \dot{x_i} + x_i = f(I_i + \alpha x_i - \beta_1 x_N - T_i), \tag{2.1}$$

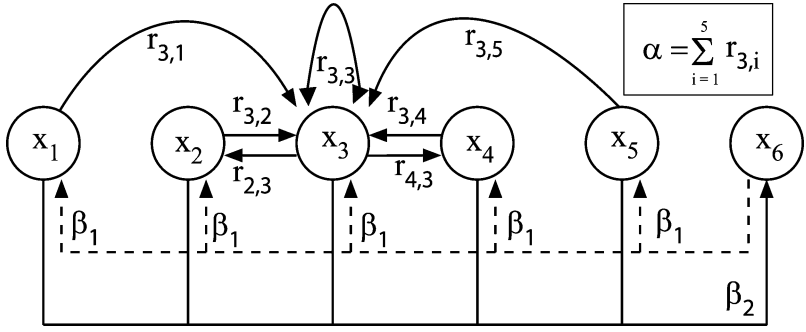

Figure 1: Structure of a single recurrent map that is composed of a number of excitatory units (x_1–x_5), and one inhibitory unit (x_6). Each excitatory unit projects to and receives input from the inhibitory neuron. Each excitatory unit is also connected symmetrically to its neighbors as well as itself (connections are shown for the example of x_3 and x_6). Note that the inhibitory unit x_6 does not connect to itself (no self-inhibition).

and the dynamics of the inhibitory neuron j is given by

$$\tau \dot{x_N} + x_N = f\left(\beta_2 \sum_{j=1}^{N-1} x_j - T_N\right). \tag{2.2}$$

I_i is a constant external input to unit i, which is usually $I_i = 0$, and $\tau = 1$. The firing rate activation function $f(x)$ is a nonsaturating rectification nonlinearity $\max(0, x)$. We also tested a nonlinearity of the form $log(a + \exp(b(x + c)))$, where a, b, and c are constants, and obtained very similar results. This nonlinearity has the benefit that it is continuously differentiable, which is necessary for the analytic analysis of fixed points. We will interpret the thresholds as possible control inputs, and so they are expressed as arguments f. However, these thresholds are usually the same for all units $T = T_i = T_N$. All integration was performed with Euler integration with $\delta = 0.05$, unless specified otherwise.

2.2 Amplification by Single Recurrent Map. Over a broad range of parameters, the recurrent map will linearly amplify a constant external signal (see the appendix). The network is continually sensitive to its input, in the sense that its output will relax toward zero when its input is removed (see Figure 2A). The output follows the input over time, provided that $I(t) > 0$. The amplitude of the steady-state output is the sum of a variable and a constant component (see Figure 2B and equation 2.3). The variable component is the input I amplified by gain $G = \frac{1}{1+\beta_1\beta_2-\alpha}$ (the slope of Figure 2B). The

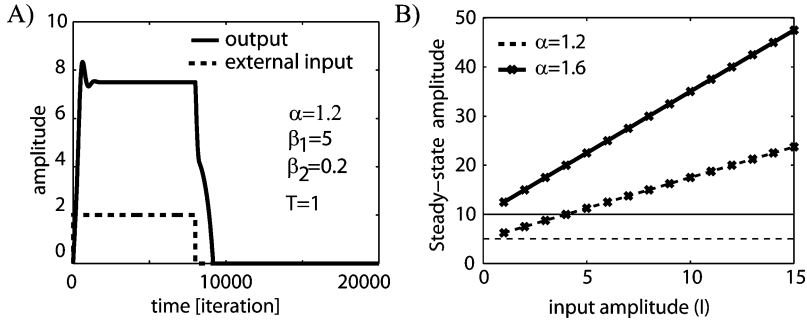

Figure 2: (A) Following application of an external input to one excitatory unit, the network activity increases to steady amplitude. This activity relaxes back to zero after the input is removed ($\alpha = 1.2$, $\beta_1 = 5$, $\beta_2 = 0.2$, and $T = 1$). (B) Steady-state amplitude of the output (bold lines) as a function of the input. The amplitude of the output consists of a constant offset (provided by recurrence, flat lines) and amplification (slope of curve) of the input. Note that the input amplitude needs to be $I > T$ for the network to be activated.

constant component of the response (flat lines in Figure 2B) is independent of the input current (provided that the input threshold is exceeded). However, it depends on G, as well as the threshold and the inhibitory coupling, β_1:

$$x_i = \frac{I_i}{1 + \beta_1\beta_2 - \alpha} + \frac{T(\beta_1 - 1)}{1 + \beta_1\beta_2 - \alpha}. \tag{2.3}$$

2.3 Combining Two Recurrent Maps. A single recurrent map consists of multiple inhibitory and excitatory feedback loops, and it can thus, in principle, give rise to oscillatory or chaotic activity (Strogatz, 1994; Wolfram, 1984). Here, we focus on a range of parameters that result in a steady state whenever the external input is constant (for reasons that will become clear later). The activity of such a single recurrent map relaxes to zero after the input is removed. However, when two recurrent maps are combined by some simple recurrent coupling, their activity can be sustained even in the absence of input (see also Xie, Hahnloser, & Seung, 2002; Zhang, 1996). For example, we combine two recurrent maps x and y, each of length N (see Figure 3), and local connectivity described by the $N \times N$ weight matrices R^x and R^y, respectively. Each excitatory unit is connected to its neighbors with $R_{ij} = \exp(-\sigma d^2)$, where d is the distance between postsynaptic unit i and presynaptic unit j. Here, $\sigma = 1$. Each excitatory unit also drives the single inhibitory neuron of its map with $R_{Ni} = \beta_2$ and receives input from that inhibitory neuron with $R_{iN} = -\beta_1$.

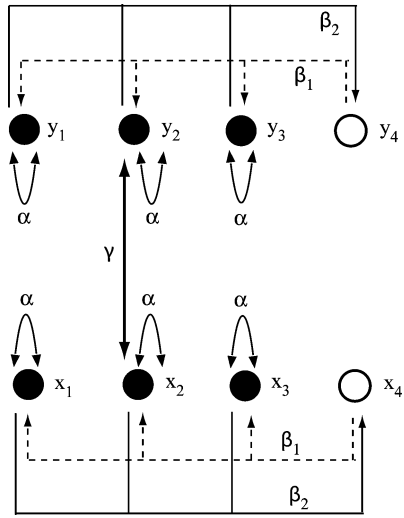

Figure 3: Two recurrently coupled maps. Excitatory neurons are black and inhibitory neurons white.

The performance of the recurrent maps depends on the loop gains for excitation and inhibition (see the appendix). So, we simplify the calculation of the excitatory gain by normalizing to α the excitatory connections received by any unit. That is, $\sum_{i=1}^{N} R_{ij} = \alpha$, so that α now sets the total potential excitatory strength received by postsynaptic neuron i from all presynaptic neurons j.

The connectivity between the two maps is described by a symmetric weight matrix C. The weights of excitatory connections between units k on both maps are set to $C_{jj} = \gamma \exp(-\sigma d^2)$ for $j = 1..N-1$, where d is the distance between unit k and j. The inhibitory neurons of the two maps are not connected ($C_{NN} = 0$).

The R and C matrices of the pair of maps are combined to form the overall weight matrix W:

$$W = \begin{bmatrix} R^x & C \\ C & R^y \end{bmatrix}. \tag{2.4}$$

Thus, W describes both the local connections of each map, as well as their interactions (e.g., Figure 7). Using W, the dynamics of the entire system can be described by

$$\tau \dot{\mathbf{z}} + \mathbf{z} = f(\mathbf{u} + \mathbf{p} + W\mathbf{z}), \tag{2.5}$$

where bold and uppercase letters indicate vectors and matrices, respectively. The vector $\mathbf{z} = [\mathbf{xy}]$ describes the activity of all units of both maps, and $\mathbf{u}$ is the external input. The input $\mathbf{p}$ is from transition neurons that will be described later.

2.4 Stable Memory State with Two Coupled Maps. When two recurrent maps are recurrently interconnected by excitatory neurons with symmetric weights γ, there are a range of conditions (see the appendix) that permit the overall network to retain stable, nonzero states in the absence of input. The amplitude of the memory state of the excitatory units is given by

$$x_i = \frac{T(\beta_1 - 1)}{1 + \beta_1\beta_2 - \alpha - \gamma}. \tag{2.6}$$

This expression is similar (except for the γ) to the constant offset term described above for the single map (see equation 2.3). A memory state exists if $T > 0$ and $\beta_1 > 1$. From the steady state alone, it may appear sufficient to have only one map (equivalent to $\gamma = 0$). However, $\gamma > 0$ is required for reasons of the dynamics: only a nonzero value ensures that the memory state is also an attractor (see the appendix).

The amplitude of the memory state depends on the product of the gain of the network (compare equations 2.6 and 2.3) and the threshold T. This perplexing result can be clarified by decomposing T into two different thresholds T_{exc} and T_{inh} for the excitatory and inhibitory units, respectively. Then the steady-state potential becomes

$$x_i = \frac{\beta_1 T_{inh} - T_{exc}}{1 + \beta_1\beta_2 - \alpha - \gamma}. \tag{2.7}$$

Now, it becomes clear that the memory state depends critically on an inhibitory threshold. Provided $\beta_1 > \frac{T_{exc}}{T_{inh}}$, the memory state exists only if $T_{inh} > 0$, whereas the excitatory units can have a zero threshold $T_{exc} = 0$ (see equation 2.7). T_{inh} is effectively thresholded disinhibition. This means that recurrent excitation can grow with high gain, until the inhibitory threshold is exceeded, so that the response of the network is stabilized. This explains why the inhibitory threshold controls the amplitude of the memory state.

The memory state property is demonstrated in Figure 4B, using a network with the following parameters: $\alpha = 1.3$, $\beta_1 = 3$, $\beta_2 = 0.2$, $T = 0.5$, and $\gamma = 0.1$. In this example, for simplicity, only self-excitatory connections are used within each map. And we have plotted only the two units on maps x and y that are recurrently connected (x_3 and y_3).

While the external input is applied, the x unit output is equal to the anticipated steady-state activation (see Figure 4B) described by equation 2.3.

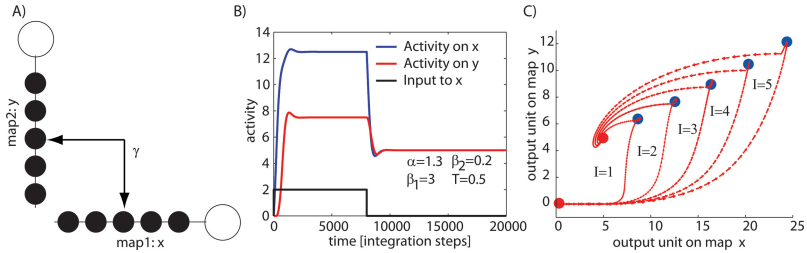

Figure 4: Two connected recurrent maps maintain stable states in the absence of external input. (A) Illustration of connectivity. Maps are recurrently connected by γ. Each map has one inhibitory neuron (larger circles). One neuron on each map (here x_3, y_3) is symmetrically connected to one another with weight $\gamma = 0.1$. (B) Input is applied to a unit on map x that is recurrently connected to map y (here x_3). After offset, the network relaxes to the memory state. (C) Dynamics of the two units x_3 and y_3, shown in phase space for different input amplitudes, applied to neuron x_3. The red dots denote points ($\dot{\mathbf{x}} = 0$ and $\dot{\mathbf{y}} = 0$) that are stable attractors in the absence of input. The blue dots are stable attractors in the presence of input. After the input of any amplitude is withdrawn, the network converges to the common red attractor. The vectors indicate the value of the derivative $\dot{x}_3$ and $\dot{y}_3$. Here, $\delta = 0.01$ (numerical integration).

The output of the y unit is smaller because it does not receive an external input: its response arises from the input it receives from other units in the overall network. When the driving input is removed, the network relaxes to a nonzero stable state where $x_i = y_i$ and $x_N = y_N$ (excitatory and inhibitory neurons, respectively). Thus, the network exhibits state (or memory) by maintaining persistent (and constant) output. The amplitude of this state is described by equation 2.6. The amplitude of the sustained response is independent of the amplitude of the input: the same memory state is reached after various amplitudes of input are removed (see Figure 4C).

2.5 Robustness to Noise. Recurrently connected networks can be very sensitive to noise, particularly when these networks lack inhibition. However, recurrent maps (which by definition have inhibition) are very robust against noise. We confirmed this robustness in the coupled maps by introducing two kinds of signal variability: readout (output) and synaptic (weight) noise. Output noise was added by an additional term in equations 2.1 and 2.2:

$$\tau \dot{x}_i + x_i = f(\alpha x_i + I_i - \beta_1 x_N - T + \mathcal{N}(0, \sigma)). \tag{2.8}$$

We tested the same network as described above (see Figure 4B) while varying the standard deviation of the noise. Then, we determined whether the network reached its memory state by calculating the mean steady-state

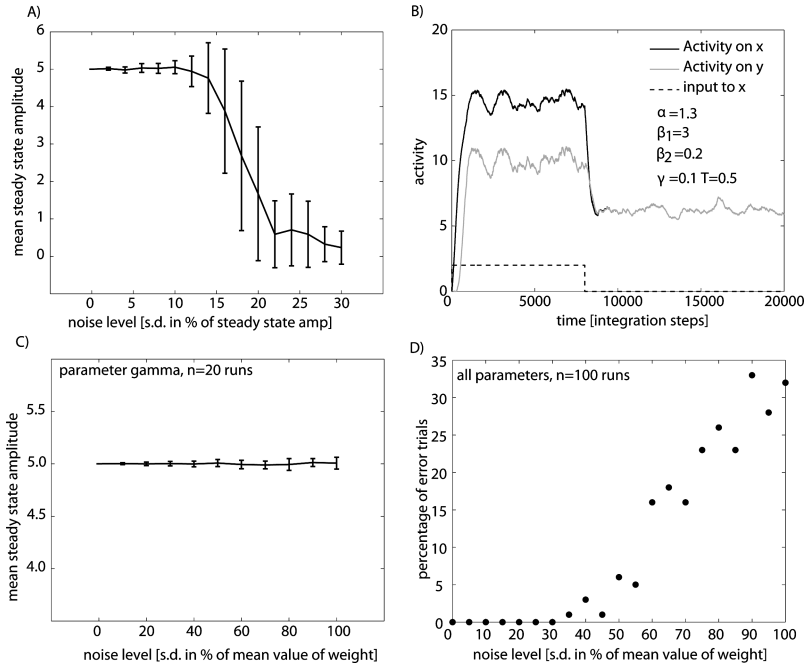

Figure 5: Stable states can be maintained in the presence of noise. All parameters are equal to Figure 4. (A) Mean steady-state amplitude as a function of the amplitude of readout noise. The noise amplitude (SD) is specified relative to the steady-state amplitude. The network can tolerate up to 15% readout noise. (B) Network activity in the presence of weight noise (γ, SD of noise equal to 0.1). Compare to Figure 4B. (C) Mean steady-state amplitude as a function of weight noise. In this example, noise was added only to γ. The network can tolerate noise amplitudes (SD) of up to 100% of the original weight of γ. Here, $\delta = 0.01$ (numerical integration). (D) Percentage of trials where the memory state was maintained successfully as a function of the amplitude of noise applied to all weights of the network simultaneously. The network can tolerate noise with a standard deviation of up to 30% of the nominal value of the weights.

amplitude after the external input was removed (time steps 10,000–20,000 in Figure 4).

First, we evaluated robustness to readout noise by adding variable amounts of noise to the output of each unit. The noise for each unit was drawn independently every $\frac{\tau}{10}$ time steps. The exact sampling interval of the noise is not critical, provided that it is significantly larger than the integration time constant and significantly smaller than τ. The results show that the network can tolerate readout noise with a standard deviation of up to approximately 15% of the steady-state amplitude (see Figure 5A). If the

noise amplitude exceeds this value, the network relaxes to zero instead of the memory state after removal of the input. Note that the readout noise is also equivalent (see equation 2.8) to random variability of the threshold T.

Next, we evaluated robustness to synaptic noise. Noise was added to the weights by setting $j = j + \mathcal{N}(0, \sigma)$ for $j \in [\alpha, \beta_1, \beta_2, \gamma]$. New values of the noise were drawn every $\frac{\tau}{10}$ time steps from a truncated normal distribution (bounds were set to $\pm j$ for $j \in [\alpha, \beta_1, \beta_2, \gamma]$ to prevent negative weights). We varied the standard deviation of the noise (σ) and quantified robustness. σ was set such that it equals a certain percentage of the nominal value of the parameter value. For example, if noise is set to 10% and the nominal value of the parameter under investigation is 2.0, the standard deviation of the noise equals 0.2.

First, we evaluated synaptic noise by adding noise to one of the four weights $\alpha, \beta_1, \beta_2, \gamma$ only. Note that this description refers to the type of weight, not an individual weight. Thus, for example, if noise was added only to α, different noise values were added to each instance of α in the entire network. For demonstration, we focus here on the weight γ, but similar results apply to the other weights. We found that the network can tolerate weight noise with a standard deviation of up to 100% of the original (noiseless) γ value (see Figures 5B and 5C). Similar amounts of noise were tolerated by the remaining weights.

Next, we added synaptic noise to all weights simultaneously. Independent noise was added to each instance of each weight. We found that the network is very robust to noise. When we ran 100 trials for each level, the memory state was stable for all trials up to 30% (see Figure 5D) noise. However, for even higher noise levels (60%), less than 10% of trials failed (see Figure 5D). Thus, the network can tolerate up to 30% noise on all weights.

Another source of synaptic noise not explicitly tested here is frozen weight noise. That is, the weights differ from their specified values, but this difference does not change over time. This kind of noise is particularly relevant for implementing circuits in analog hardware (Pavasovic, Andreou, & Westgate, 1994; Serrano-Gotarredona & Linares-Barranco, 1999). Typically, the effective weight is approximately 30% to 50% of the nominal (specified) weight (Neftci, Chicca, Indiveri, Slotine, & Douglas, 2008). This noise is introduced at the time of fabrication and is fixed. Our circuits are robust to such noise as long as the resulting parameters are still within the valid range for stability (see the appendix). Robustness to such noise can be increased by choosing nominal parameter values, which lie in the middle of the permitted ranges rather than on the edge.

2.6 Controlled Transitions Between Multiple Memory States. A DFA consists of nodes and edges (representing the states and transitions, respectively). So far we have described only how coupled maps can be used to construct a network with multiple stable attractors that can implement the DFA nodes. Now, we show how the transitions between these embedded

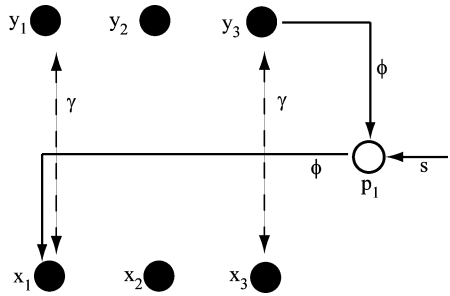

Figure 6: Transition neuron (p_1) that, when activated by input s, leads to a switch to a different memory state. Only connections between the two maps are shown. All local connectivity (α, β_1, β_2) is as shown in Figure 3.

states (the DFA edges) can be induced by transiently applying a short pulse of input to a currently unexpressed target state. That input causes the network to switch from the current state to the target (pulsed) state.

We implemented this transition mechanism using a variant of the pointer neurons that have been used previously to bias the competition on a single recurrent map (Hahnloser et al., 1999), for example, to enable the winning stimulus to emerge at a preferred direction rather than the point of maximal input. A pointer neuron is a unit that is symmetrically connected to the excitatory units on the recurrent map. Thus, the pointer neuron both receives input from and sends output to the excitatory units. This connectivity of the pointer neuron is nonuniform. Previously, a gaussian connectivity profile was used to selectively bias activity in a particular region of the excitatory map (Hahnloser et al., 1999). Here, we apply the same basic idea to implement more specialized transition neurons (TNs; see Figure 6 and equation 2.9). These units temporarily bias the competition such that the currently maximally active unit (the current state) loses the competition in favor of the next (target) state. They combine activity from map y with the external input s. If the appropriate members of y and s become active, the transition neuron becomes active and initiates the required state transition:

$$\tau \dot{\mathbf{p}} + \mathbf{p} = f(P\mathbf{y} + \mathbf{s} - T_p). \tag{2.9}$$

P describes the connectivity between the M TNs and the activity of the y map. It has dimensionality $M \times N$, where M is the number of transitions (edges) of the DFA and N is the number of excitatory units on the map. TNs receive no recurrent input ($P_{ii} = 0$). Each TN i receives input from the excitatory unit j that represents the state from which the transition originates ($P_{ij} = \phi$). Similarly, the TN sends its output to the unit k that represents the state where the transition leads to ($P_{ik} = \phi$). The range for possible values of ϕ is derived in the appendix.

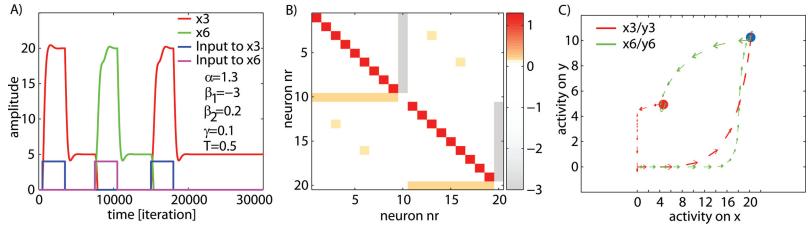

Figure 7: Short bursts of input induce rapid transitions to different states. Two units (x_3 and x_6) are recurrently connected. (A) A short burst of input to either x_3 (blue) or x_6 (magenta) elicits a rapid transition. The peak of activity remains at the same location after removal of the input (the memory state). (B) The weight matrix W of the network. Positive weights are shown in color, negative weights in gray scale. The recurrent excitatory connectivity can be seen on the main diagonal. The connections between the map can be seen on the upper and lower lines parallel to the diagonal. (C) Phase space representation of the dynamics of the first switch (red to green). The blue and red dots indicate steady state while input is applied and in the absence of input, respectively.

$T_p > 0$ is a constant threshold that suppresses TN output in the absence of external input. The external input s represents the input symbols. Every TN that represents the same symbol receives the same input. A short pulse of activity on that input signals the arrival of that particular symbol. T_p is set equal to the steady-state value reached by the map units when external input is applied (see below).

We demonstrated this transition mechanism in a network composed of two recurrent maps of $N = 10$ units, in which we embedded two states by recurrently connecting units 3 and 6 of each map with weights γ. Thus, $C_{jj} = \gamma$ for $j \in [3, 6]$. The other parameters are indicated in Figure 7. The network (see Figure 7) has two stable attractors with peaks of activity at x_3 and x_6.

The behavior of the network is shown in Figure 7. At first, the network is quiescent. After a short initializing pulse of activity to x_3, the network relaxes to a peak of activity at unit x_3 that is sustained even after the input has been withdrawn. A short pulse of input applied to x_6 elicits a transition from the current state x_3 to the target state x_6. The state switch occurs because of the competitive nature of the WTA. Transiently, the total input to x_6 is larger than to x_3. Because of this, the WTA selectively amplifies x_6 and suppresses x_3. The amplitudes of the steady states during application of the input as well as during the memory period are described by equations 2.3 and 2.6.

Thus, two recurrent maps can be used to construct stable attractors that provide persistent output in the absence of input. For convenience, we have shown here only the simplest case of two states. However, additional states can be embedded easily by inserting the necessary additional weights in the connection matrix C (lower left and upper right quadrants in Figure 7).

2.7 Implementation of a State Automaton Using Coupled Recurrent Maps. The multiple stable states embedded in the coupled maps, and the transitions between them, can be utilized to implement a DFA. The DFA operates as follows. Starting in an initial state, it reads a sequence of input symbols. For each successive symbol in the input sequence, the DFA transitions to a next state that is determined by the DFA's transition function. The state that the DFA finds itself in when the input is exhausted determines the DFA's evaluation of the input sequence. If, on exhaustion of input, the DFA is in an "accept" state, then the input sequence is deemed "accepted"; otherwise it is deemed rejected (Hopcroft et al., 2000).

We implemented the neuronal DFA as follows. First, we embedded the same number of attractors as the DFA has states. To achieve this, we created two recurrent maps consisting of $mN + 1$ units each (m is the number of states and N the number of units per state). The number of units N required to represent a state is determined by the width σ of the connection profile between the two maps (see equation 2.4). For each state in the DFA, one pair of units is designated to represent that state (here, for a simple example of only two states, we added $N = 3$ units per state; the state $q0$ is represented by x_3, y_3 and $q1$ by x_6, y_6). Each unit that represents a state is recurrently connected to its counterpart on the other map (here, x_3 to y_3 and x_6 to y_6) with weight γ. The lateral connectivity on each map is the same as detailed above.

Next, a transition neuron (TN) is added for each possible state transition. Each TN receives two inputs: one external (the input symbol that initiates the transformation) and one internal (the state from which the transition originates). Either or both of these inputs may be zero, in which case the output of the TN is bound to be zero, so that no transition will occur. To achieve this behavior, each TN receives a constant negative (inhibitory) input $-T_p$ that prevents it from becoming active in the absence of input, even when the network is currently in the preferred state of the current TN. This arrangement also prevents a state transformation if a symbol is received for which there is no defined transition from the current state.

The output of the TN is connected to the unit on map x that represents the target state that the transition leads to. While activated, the TN output will bias the competition on the coupled maps, encouraging the new state to become dominant. All TNs that represent the same symbol receive the same input. Here, the amplitude of the external input to the TNs is set equal to T_p. The value of T_p is chosen such that it is somewhat larger than the steady-state amplitude reached by the map neurons in the presence of constant input (any value larger than x_i as calculated by equation 2.3 is allowed). Thus, the overall network has as many inputs as there are distinct symbols. The size of the network (in terms of number of units) scales as $O(m + n)$, where m is the number of states and n the number of state transitions. Thus, the network scales linearly with the number of states. While it might be possible to represent states by a combination of

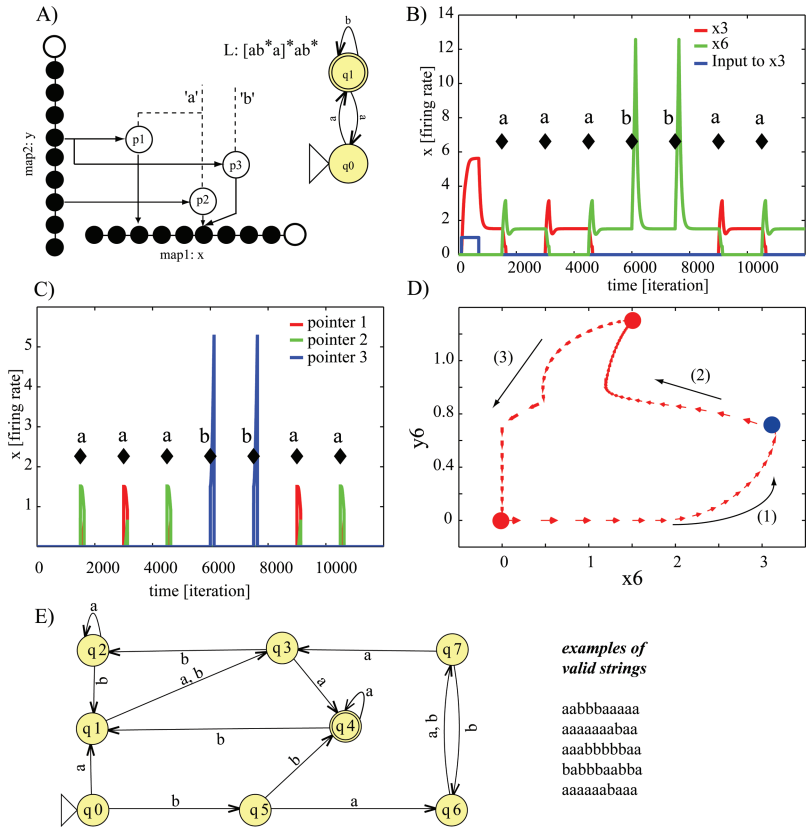

Figure 8: Implementation of a DFA using recurrent maps. (A) DFA that implements the language (ab*a)*ab* (right) and its recurrent map implementation (left). (B, C) Processing of the sequence *aaabbaa*. After a short burst of initialization (blue), no external input is applied to the network except for the symbols. The recurrent net switches to the appropriate state (B), and the appropriate pointer neuron is activated (C). (D) Illustration of the state switch from $q0$ to $q1$. Activation of the pointer neuron induces a transition to the next state (1). After offset of the pointer neuron input, the network settles to the stable state $q1$ (2). After the network transitions to another state, activity returns to the baseline state (3). (E) Example of a randomly generated minimized DFA with eight states. Example strings that lead to the accept state are shown on the right.

units (and thus scale better), we opted here for an explicit representation of each state for purposes of analysis.

2.7.1 Example DFA. We demonstrated these properties by constructing a simple DFA that implements the regular language *[ab*a]*ab** (see Figure 8A).

The coupled map has two states ($q0, q1$). Its input alphabet is the two symbols a and b, and it has the transition functions $q_1 = (q_0, a)$, $q_0 = (q_1, a)$, and $q_1 = (q_1, b)$. The example input string *aaabbaa* is a valid string under this language and brings the DFA to the accept state $q1$ (represented by x_6). In this more sophisticated example of a coupled map, neighbor connections are used between the local excitatory neurons, and between the cross-coupled populations.

The dynamics of x_3 and x_6 during the processing of the input string confirms that the neuronal DFA executes each state transition correctly (see Figure 8B). The dynamics are further illustrated by plotting the activity on map y as a function of the activity on map x (phase space, Figure 8D). Each unit x_i that represents a state of the DFA has two stable attractors (where $\dot{x}_i = 0$), in the absence of input (see Figure 8D, red dots). Before processing the input string, the DFA must be initialized to its initial state by a short external pulse (blue in Figure 8B). Note that such neurons, which indicate the start of a sequence, have been observed in the cortex (see discussion). Thereafter, short pulses of input are applied to the appropriate TNs, depending on whether the symbol a or b is present (black diamonds in Figures 8B and 8C). The duration of the external inputs, representing the symbolic inputs, needs to be long enough for the network to converge. Here, we used 15τ (300 iterations for a $\delta = 0.05$). The number of iterations necessary for the network to execute a state switch as well as to relax after removal of the external input does not depend on the number of states (see the appendix for a calculation) or on the number of TNs. However, it does depend on the parameters of the network.

2.7.2 Implementation of DFAs of Arbitrary Size. So far, we have demonstrated how to implement a DFA with two states and four transformations using the coupled maps. Is it possible to implement DFAs of arbitrary size using the same construction rules? To facilitate the construction of arbitrary DFAs, we have developed software that automatically converts a DFA constructed in a graphical utility (JFLAP) (Rodger & Finley, 2006) into two dynamically coupled maps.[1] We used this software to generate random DFAs using the method described in Bongard and Lipson (2005). We used a standard algorithm (Hopcroft, 1971; Hopcroft et al., 2000) to minimize DFAs with respect to the number of states used to represent the language represented by the random DFA. The random DFAs were automatically converted to a weight matrix for the two WTAs and their interactions (the states), as well as the connections of the TNs.

The weight matrix was constructed using the following rules: (1) add a few units (here, we used five) to both maps for each state; (2) add one TN

[1]Source code is available on the first author's Web page or can be requested by e-mailing the authors.

for each transformation, and connect them appropriately; and (3) connect each state in the weight matrix as described above. How many units are added per state (here, five) depends on the width of the local connectivity (see σ in section 2.3; the bigger σ, the more units per state need to be added such that the activity bumps of two states do not overlap). The generated DFA was then evaluated in terms of its ability to correctly classify randomly generated strings. Like the standard DFA, the coupled map DFA was deemed to classify a string correctly (or "recognize" the string) if, after all input symbols have been processed, the DFA reached its accept state. If any other state was reached, the processing was incorrect (unless the input was not a string of the language). We tested random DFAs of up to 40 states with 100 random strings each and found that all these strings were processed correctly (an example is shown in Figure 8E).

2.7.3 Robustness. Long chains of excitatory neurons are capable of producing sophisticated patterns of activity, as exemplified by synfire chains (Abeles, 1991). However, the more neurons that follow each other in this feedforward arrangement, the more sensitive the output becomes to the input. This is because errors accumulate and propagate in those architectures. Does this error accumulation occur in our architecture? If yes, then the network should fail to process long sequences of symbols. While the transition executed for each input symbol depends on only the current state, the current state includes information about all previous input symbols (it fulfills the Markov property). Thus, the appropriate test is whether the network can reliably process long sequences of input. If errors accumulate, the network behavior should break down beyond a certain string length.

Two principal actions of the network are sensitive to noise: keeping the memory state (see section 2.5) and executing a state switch. Here, we evaluate both simultaneously. Without the presence of artificial noise sources, we found that long random strings (we tested up to 30 symbols) are processed with the same reliability as short strings. Thus, numerical errors of the integration do not accumulate. Next, we added output or synaptic noise as described in section 2.5. The performance of one particular network as a function output and synaptic noise levels is shown in Figure 9 (see the figure legend for network parameters). This particular network can tolerate output noise of 8% of the steady-state amplitude, as well as approximately 10% weight noise. After the noise threshold is reached, performance degrades rapidly and quickly reaches chance performance (four states; thus, 25% in this case). The level of noise that a particular network can tolerate depends on the parameters of the network (i.e., the weights, the number of units per state), the distribution of the noise, and the sampling frequency of the noise (see Figure 9C for an example). Also, if $\alpha_i > 0$ (for $i > 1$), robustness to noise depends on the spread of connectivity (determined by σ; see section 2.3) as well as the number of units dedicated to a particular state. The bigger σ, the more units are required to represent a particular

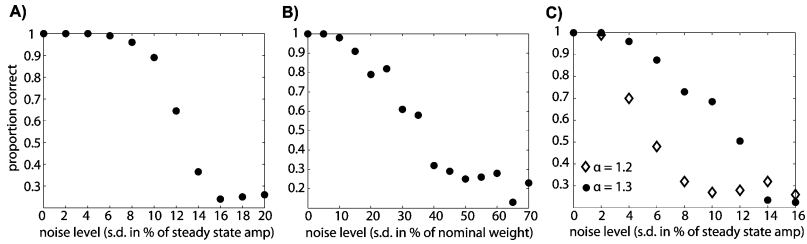

Figure 9: Illustration of noise robustness of a network implementing a random DFA with four states (chance performance is 25%). The parameters of the network are $\alpha = 1.3$, $\beta_1 = 3$, $\beta_2 = 0.2$, $\gamma = 0.1$, $\phi = 0.88$, $T = 0.5$. The integration time constant is $\delta = 0.05$. Networks were fed random strings of length 4 to determine whether the network reaches the correct state. (A) Performance for different levels of output noise (applied to all units). The steady-state amplitude of the network was 5.0; thus, 10% noise corresponds to an SD of 0.5. (B) Performance for different amounts of weight noise. Noise was applied to all weights. The SD of the noise was set relative to the nominal weight of each weight (y-axis). (C) Network performance for output noise and different values of α (1.2 versus 1.3). Note that the larger value of α results in a more noise-robust network for this configuration. All other parameters are the same.

state. Whether this increases or decreases reliability depends on the other parameters of the network. Also, it is beneficial to choose the weights such that they lie in the middle of the permitted values rather than on the edge. This ensures that noise does not cause instabilities in the network.

This result demonstrates one of the crucial aspects of computations with WTAs: states are stable no matter what the initial conditions. Thus, noise does not accumulate over time because the signal is restored (the noise is rejected) at every step. This is the reason that previous approaches, using saturating neural networks (see section 3), cannot process long strings: performance degrades even if no noise is added (due to numerical errors). No matter how accurate the feedforward network, the performance of long string acceptance eventually degrades. In contrast, this problem does not exist for the DFA implemented using WTAs. A similar strategy has also been used to make synfire chains robust to noise (Diesmann, Gewaltig, & Aertsen, 1999).

3 Discussion

It has been known since the foundational report of McCulloch and Pitts (1943) that recurrently connected neuronal networks have the properties of a finite state machine. Subsequent work has elaborated and clarified those properties—for example, Minksy (1967) from the point of view of general computation; Grossberg and Hopfield from stability (Cohen & Grossberg,

1983; Hopfield, 1984); and Elman and Forcada (Carrasco, Forcada, Valdés-Muñoz, & Neco, 2000; Kremer, 1995; Elman, 1991) from the point of view of network computation. In general, however, these approaches have assumed inherently reliable neurons with saturating neuronal activation functions. By contrast, biological cortical neurons are subject to various sources of noise and operate well beneath saturation in the linear range of their activation function (Douglas et al., 1995).

This letter demonstrates that robust state-dependent processing can be embedded in networks of neurons whose activation function is both nonsaturating and nonlinear (thresholded). Consequently, they can depend on only their dynamics of inhibition, excitation, and thesholding for their noise resistance and stability. Moreover, we have provided simple rules that permit the systematic design and construction of a (nearly) arbitrary neuronal state machine composed of nearly identical recurrent maps. We claim "nearly arbitrary" because for practical reasons, we tested the automatic design process only out to 40 embedded states (because it is not easy to find a minimized random DFA consisting of more than 40 states) and "nearly identical" because some specific connections are required to implement a specific state machine within an otherwise generic neural architecture.

Essentially we construct a multistable network that embeds a number of states by coupling two recurrent maps whose weights have been set to provide soft winner-take-all (sWTA) performance. The two sWTAs have identical connectivity except for a small fraction of symmetrical (recurrent) cross-connections between them that are used to embed the required states. This recurrent coupling between the maps allows them to sustain (remember) their current state in the absence of input.

The network can be switched from its current state to another of its states by applying a short burst of input. The required transitions between states are affected by a class of neurons that activate the new state, conditioned on the current state and also an external input symbol. Thus, the network's reaction to the input is state dependent.

A class of transition neurons drives the state transitions by combining the current state of the network (memory) with an input symbol to excite the next state of the network. The transition neurons are similar to the pointer neurons described by Hahnloser et al. (1999), which they used to steer feedback over the entire range of a single map of neurons. For example, pointer neurons are able to bias WTA behavior in favor of inputs at certain locations of the map, so that activity at these locations wins the WTA competition even if they are numerically smaller than activities at others (which would otherwise win an unbiased WTA competition). In that application, the pointer neurons participate directly in the feedback between the neurons of the WTA and so participate directly in the ongoing computation of the WTA. By contrast, our transition neurons are activated only when a new input symbol is processed. As soon as the state switch has occurred (or no input symbol is applied), the transition neurons become inactive again.

In our network, this selective activation is due to inhibition that is continuously applied to all transition neurons. A transition is triggered only when the combined activity of the current state and the external input exceeds the inhibitory threshold.

The networks described here can function over a wide range of weights (as long as certain conditions are met; see the appendix for a summary). Of particular interest is that the coupling constant γ between the two recurrent maps must be small compared to the local recurrent excitatory weights ($\alpha \gg \gamma$). Thus, a very small weight is sufficient to couple two independent WTAs such that they bias each other's winner. This is well in line with our previous finding that the sensory excitatory input to pyramidal neurons in layer 4 of visual cortex is minor compared to excitatory input from other neurons of the same layer (Binzegger et al., 2004).

Our model uses a single global inhibitory neuron per map to implement the competitive component of the sWTA. Which excitatory units compete with each other is determined by whether they share common inhibition. Experimental evidence exists for inhibitory feedback loops on many different scales, such as local lateral inhibition and diffuse inhibitory feedback mediated by the thalamus (Douglas & Martin, 2004). Such feedback loops can enforce competition and thus result in WTA behavior. In some structures, it has been experimentally demonstrated that such global (diffuse) inhibition exists and that they serve to enforce WTA-type competition (Kurt et al., 2008; Baca, Marin-Burgin, Wagenaar, & Kristan, 2008; Tomioka et al., 2005). WTA networks can, however, also be implemented using more local forms of inhibition (Yuille & Grzywacz, 1989). Global inhibition necessarily makes the weight matrix asymmetric, and so it is difficult to derive analytically the conditions on the weights that guarantee convergence. One approach to this difficulty has been to assume that, locally, inhibition is infinitely fast (Grossberg, 1973; Wersing, Beyn, & Ritter, 2001). However, we wanted to keep our network physiologically plausible and so could not accept this assumption. Seung (Seung, Richardson, Lagarias, & Hopfield, 1998) has suggested a Lyapunov function for inhibitory-excitatory networks similar to those used here. His function requires the inhibition to be symmetric ($\beta_1 = -\beta_2$). However, if a WTA is to function properly, it is crucial that this is not upheld, and so we could not use Seung's approach. Instead, we used linearization at steady states and numerical simulations to confirm that our approach is capable of implementing arbitrary DFAs.

We aim to model state-dependent processing of external inputs. Our network thus requires an explicit external input to trigger a transition to the next state. This transition is state dependent. These external inputs could be the result of some action performed at every state and thus would not be available in advance. We used the processing of a list of symbols as an abstract model of this behavior. Another application of sequential transitions between states has been the memorization of sequences. In this case, an external stimulus would trigger the automatic replay of the entire

sequence without requiring any further external inputs. An example is winnerless competition, where networks are constructed such that each state is a saddle point with only one unstable direction, which leads to the next stable point (Seliger, Tsimring, & Rabinovich, 2003; Rabinovich et al., 2001). While this was not our principal aim, our network could also be enhanced to allow autonomous transitions between states. This could be achieved by replacing the TNs with units that connect different states with each other (i.e., the input to such a unit is a state on one map and the output another state on another map). When appropriate time constants are used, such a network would autonomously transition between a number of states, given that it is placed in the appropriate initialization state.

There have been previous approaches to systematically embed an arbitrary DFA into neural networks. However, they have generally used unrealisitic assumptions. For example, the stability of the DFA networks of Omlin (Omlin & Giles, 1996; Giles et al., 1992) and Kremer (1995) depends on the saturating nonlinearity, which we have argued is not a realistic physiological requirement. Other approaches use Elman-type networks. But these networks assume a memory layer from which activity can be artificially copied at every iteration (Kremer, 1995). Thus, the network itself has no memory. These approaches are not physiologically plausible. By contrast, our approach requires no artificial operations (like delay lines), the weights can be easily set, and their settings are independent of the DFA.

Another approach has been to project the input into a high-dimensional space using randomly connected units (the liquid state machine, LSM) (Maass, Joshi, & Sontag, 2007; Jaeger & Haas, 2004). Readout units that receive input from all the units in the pool are then trained to approximate the required output states. However, LSMs have fading memory, and so DFAs implemented in this way can depend on only a restricted number of past inputs (Natschlager & Maass, 2002). This deficiency can be remedied by training readout units to represent the current state and feeding their activity back into the pool so as to maintain their current state. This circuit can implement arbitrary DFAs (Maass et al., 2007). However, that approach relies on supervised learning of the weights of the output units without modification of the units in the liquid. By contrast, our network can be constructed explicitly, entirely without learning, and so allows detailed understanding of the connectivity that underlies its operation. One advantage of the liquid approach is that relatively few weights (connections to the readout) need to be changed, while all others can be held constant. This property is also true for our approach. The entire connectivity on the maps is stereotyped and does not depend on the particular state machine implemented. Adding a new state requires only connecting at least two new units reciprocally between the two sWTAs, with weights γ.

The weights of the connections in our network have no plasticity (they are static). The only changes in dynamics that can occur are thus due to activity. Plasticity occurs on a slower timescale than changes in activity,

and it would thus introduce a second, slower, timescale of dynamics into the system. Such interactions between fast (activity) and slow (plasticity) dynamics in the same networks can result in complex dynamics. One of the questions posed by the introduction of plasticity is how multistability can be preserved (rather than convergence to a single stable state). For our network, this remains to be explored. However, for other networks, it has been shown that configurations exist that allow such multistability to persist in the presence of plasticity (Kalitzin, van Dijk, & Spekreijse, 2000). We designed our network such that only a small number of weak connections need to be modified to erase or introduce stable states. The remaining connectivity is homogeneous and independent of function. Given multiple recurrent maps with no (or random) coupling between them, it is thus imaginable that a simple plasticity rule can be found that learns the necessary states and the transitions between them.

Sophisticated behavior requires that a reaction to a particular stimulus is state dependent and so requires working memory. The frontal cortex is known to be crucial for this function, and it is known to contain neurons that respond differently, according to behavioral state. One well-studied example is the generation of memory-guided sequential motor movements (Fujii & Graybiel, 2003; Shima & Tanji, 2000; Mushiake, Inase, & Tanji, 1990; Barone & Joseph, 1989). During execution of this task, there are two main classes of firing patterns in the supplementary motor area (SMA) (Shima & Tanji, 2000). One class of neurons fires before, during, or after a particular movement is executed, regardless of where in the sequence it is. Another class fires in response to a particular movement, but conditionally so: only if the movement is at a particular position in the sequence or if a particular movement was executed before. Of particular interest are two subclasses of these latter neurons. The first subclass corresponds to our current state neurons and fires for a particular location (e.g., second) in the sequence. The second subclass fires between executions of movements and is conditional on the previous and the next-to-be-executed movement (Shima & Tanji, 2000). They correspond to the transition neurons in our network.

Neurons related to the execution of state-dependent actions have also been described in the prefrontal cortex. For example, some prefrontal units of nonhuman primates fire prominently only at the end of the correct execution of a complex sequence of motor actions (Fujii & Graybiel, 2003). In analogy with our DFA network, these neurons could indicate that the accept state has been reached. Alternatively, they could signal the return to the initial (start) state.

Another important property of our network is that it combines the DFA state property with analog sensitivity: the degree of output activation of the state neurons can be modulated by external input. This is also true during the memory state when no external input is present. Then a positive signal, T_{inh}, applied to the inhibitory neuron can modulate the amplitude of the output without causing the network to leave the memory state. If

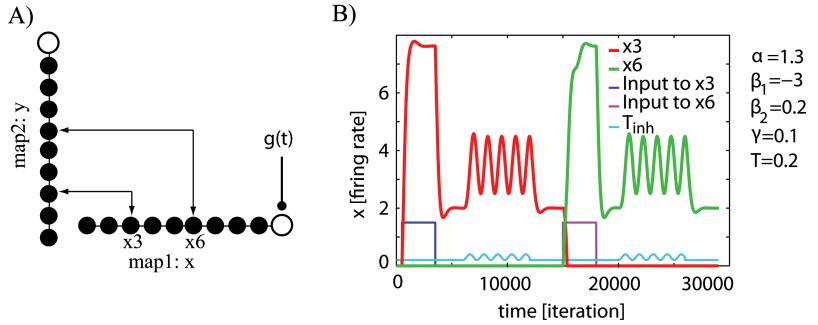

Figure 10: Demonstration of state-dependent routing. (A) Illustration of a possible network architecture. States x_3 and x_6 represent the two possible states of the network. External inhibitory input $g(t)$ is applied to the inhibitory unit. (B) An analog signal applied to the inhibitory neuron (T_{inh}) appears only at the unit representing the current state. Depending on the current state, the external signal appears only at the unit representing state 1 (red) or state 2 (green). Here, the effective T_{inh} is plotted, which is composed of the static nonzero firing threshold (offset from zero) and the external input, which varies as a function of time. Note that the output oscillation is not generated by the network but rather is a reflection of external input.

$T_{inh}(t) = g(t)$ is a function of time. The network output will vary according to the function $g(t)$, while the network is in the memory state. Our network thus combines the processing of digital (the states) and analog (the external input) signals (Hahnloser et al., 2000). One possible application of this feature is state-dependent routing of arbitrary continuous signals. In this case, the signal to be routed would be supplied to the inhibitory neuron, and its evoked output would be visible only at those excitatory neurons representing the current state. Thalamocortical input is known to project directly to inhibitory neurons (Swadlow, 2002). Thus, the routed input signal (provided to the inhibitory neuron) would be visible only to neurons downstream from the current state neurons. When the state changes, the analog signal will be routed to another set of downstream neurons (an example is shown in Figure 10B). Modification of the inhibitory neuron firing threshold (i.e., by an additional input; see Figure 10A) can also be used to dynamically enable and disable the memory state. This, for example, can be used to implement a conditional working memory. By default, $T_{inh} = 0$. Only if the current stimulus is required to remain in working memory is $T_{inh} > 0$. Also note that this feature can be utilized to implement a decaying memory. By default, the persistent activity in our network does not decay as a function of time: given no external perturbations, the network remains in the memory state forever. This is clearly not desirable for some functions, such as working memory. One way to introduce working memory is to continuously decrease T_{inh} relative to stimulus offset. This could, for

example, be implemented by an additional excitatory neuron (with self-excitation) that feeds into the inhibitory neuron.

Our network performs state-dependent computations because it has multiple memory states. Here, we chose to implement this ability by recurrently connecting two maps. Each of the two maps has homogeneous local excitatory connectivity and one global inhibitory unit. While a single map alone can have nonzero steady states (see equation 2.6), this state is usually not stable. Only by connecting two maps ($\gamma > 0$) can the stability of the memory state be guaranteed (also see the appendix). Other recurrent connectivity patterns (such as multiple recurrent loops with different delays) that result in memory states are certainly possible and remain to be explored. We chose the coupled map configuration because it requires only the replication of a simple circuit (a map) and the addition of a few (numerically small) connections between the maps and so offers a method whereby the cortex could achieve a broad range of sophisticated processing by only limited specialization of the same generic circuit.

Appendix: Derivations, Stability Analysis, and Valid Parameter Ranges

A.1 Constraints on Parameters. We derive the steady-state values for various configurations of the network using linear equations of the form $f(x) = x$. For simplicity, we assume that only $\alpha_1 > 1$, whereas all other $\alpha_i = 0$ for $i > 1$. The following equations describe the steady state reached after a sufficient number of iterations. During the dynamic part of the system's activity, some values of x_i might be zero, and the equations are thus not valid. The same approach as used here can, however, also be applied to this situation by replacing the rectification function with a continuously differentiable function of the form $f(x) = \log(a + \exp(b(x + c)))$ (a,b,c are constants).

Also note that the approach presented here is valid if the network contains units with $f(x) = 0$. In that case, these units are effectively nonexistent and can be ignored for purposes of steady-state analysis. As long as the subset of active units $f(x) > 0$ remains constant, this subset can be analyzed separately using the methods described here (i.e. piecewise analysis; Hahnloser, 1998).

A.2 Constraints for Bounded Map Activity for Constant Input I and $T = 0$. First, we define the constraints for bounded activity for constant input I_i to unit i only ($I_j = 0$ for all $j \neq i$) and $T = 0$. Steady state implies $\dot{x_j} = 0$ for all units $j = 1..N$ on the map. Also, $x_N > 0$ (inhibitory neuron) and $x_i > 0$. Solving the system of equations described by equations 2.1 and 2.2 for x_i results in

$$x_i = \frac{I_i}{1 + \beta_1\beta_2 - \alpha}. \tag{A.1}$$

Thus, the recurrent network amplifies the input by a factor of $\frac{1}{1+\beta_1\beta_2-\alpha}$ at steady state, provided $\alpha < 1 + \beta_1\beta_2$.

A.3 Constraints for Bounded Map Activity for Constant Input I and $T > 0$. Next, we define the steady-state value for constant input in the case of $T > 0$. Solving the system of equations with this constraint for x_i results in

$$x_i = \frac{I_i + T(\beta_1 - 1)}{1 + \beta_1\beta_2 - \alpha}. \tag{A.2}$$

This equation describes the steady-state value if $T > 0$ for any value of I_i. One additional constraint added by this solution is that $\beta_1 > 1$.

Next, we describe the same two steady states for the case of two recurrently coupled maps **x** and **y** (see Figure 3). Coupling is symmetric with weight γ. Here, we assume x_2 and y_2 are connected with $\gamma > 0$. This assumption changes the dynamics of the excitatory neuron but not the inhibitory neuron:

$$\tau\dot{x}_i + x_i = f(\alpha x_i + \gamma y_i + I_i - \beta_1 x_N - T). \tag{A.3}$$

A.4 Constraints for Bounded Map Activity for Constant Input I and $T = 0$. Solving the new system of equations (see equations A.3 and 2.2) for $T = 0$ results in

$$x_i = \frac{I_i}{1 + \beta_1\beta_2 - \alpha - \frac{\gamma^2}{1+\beta_1\beta_2-\alpha}}. \tag{A.4}$$

Defining $K = 1 + \beta_1\beta_2 - \alpha$, the amplification factor of the recurrently coupled map is $\frac{1}{K-\frac{\gamma^2}{K}}$. Thus, the gain is well defined when $\gamma^2 < K$.

A.5 Constraints for Bounded Map Activity for Constant Input I and $T > 0$. We continue to assume $K = 1 + \beta_1\beta_2 - \alpha$. Solving the same system with the constraint $T > 0$ results in

$$x_i = \frac{K I_i + T(\gamma + K)(\beta_1 - 1)}{K^2 - \gamma^2}. \tag{A.5}$$

This equation describes the steady-state potential reached for constant input and $T > 0$. It consists of two components. The first term is the input multiplied by the gain. The second term is constant additional input that is provided by recurrent connections from other units of the network. The effect of this second term remains even after the input has been removed ($I = 0$; see below).

A.6 Existence of Memory State. A memory state exists if the excitatory neurons on both maps representing the states are nonzero ($x_i > 0$, $y_i > 0$), the inhibitory neurons on both maps are nonzero ($x_N > 0$, $y_N > 0$), and if a steady state is reached: $\dot{x} = 0$, $\dot{y} = 0$. Also, if the steady state is reached, $x_i = y_i$ for the units i representing the stable state (connected by γ).

Setting $I = 0$ (no external input) in equation A.5 results in

$$x_i = \frac{T(\beta_1 - 1)}{1 + \beta_1\beta_2 - \alpha - \gamma}. \tag{A.6}$$

This equation describes the amplitude reached by the excitatory units. A memory state exists only if $T > 0$ and $\beta_1 > 1$. As also seen previously, $1 + \beta_1\beta_2 > \alpha + \gamma$. Note that the absolute value of the memory state is independent of the previously applied input I, that is, it is entirely determined by the structure (weights) of the network.

The steady state of the inhibitory neuron in the same situation can be calculated by equation A.7. It adds the constraint $\alpha + \gamma > 1 + \beta_2$:

$$x_N = \frac{T(\alpha + \gamma - 1 - \beta_2)}{1 + \beta_1\beta_2 - \alpha - \gamma - 1}. \tag{A.7}$$

A.7 Dynamics After Constant Input Is Removed. The dynamics of the network following removal of the constant input, until the steady-state memory state is reached can be characterized by the eigenvalues of the Jacobian of the entire system. Consider two recurrent maps **x** and **y** with one excitatory and one inhibitory neuron each. The system of equations shown above can be described by

$$\tau\dot{\mathbf{z}} - \mathbf{z} = f(W\mathbf{z} - z + \mathbf{I}). \tag{A.8}$$

Assume $\mathbf{z} = [x_1, y_1, x_2, y_2]$, $\tau = 1$ and

$$W = \begin{bmatrix} \alpha & \gamma & -\beta_1 & 0 \\ \gamma & \alpha & 0 & -\beta_1 \\ \beta_2 & 0 & 0 & 0 \\ 0 & \beta_2 & 0 & -1 \end{bmatrix}. \tag{A.9}$$

There are two nonzero fixed points in this simple system: the point reached with constant external input ($I > 0$, equation A.5) and the point reached after removal of the input ($I = 0$, equation A.6). First, assume constant external input $I > 0$ was applied for a sufficient amount of time for the system to reach steady state (see Figure 4B). Here we are interested in the dynamics that result from removal of this constant external input. There

are two steady states the system can converge to: the memory state and the zero state (no activity). To analyze the stability of the memory state, we linearize the system at the memory state. Since the network reached its steady state with $I > 0$ previously, all $z_i > 0$ and thus $f(x) = x$. Constructing the Jacobian matrix of the system then results in

$$J = \begin{bmatrix} \alpha - 1 & \gamma & -\beta_1 & 0 \\ \gamma & \alpha - 1 & 0 & -\beta_1 \\ \beta_2 & 0 & -1 & 0 \\ 0 & \beta_2 & 0 & -1 \end{bmatrix}. \tag{A.10}$$

The eigenvalues λ_k of J determine which state will be reached (Strogatz, 1994). If the matrix is negative definite (i.e., if the real parts of all eigenvalues are $Re(\lambda_k) < 0$), the memory state is asymptotically stable. Under most situations that satisfy the constraints given for the parameters, we observe that this is the case. The imaginary parts $Im(\lambda_k)$ determine how the system approaches the steady state. If $Im(\lambda_k) \neq 0$, the system oscillates around the attractor with a continually reduced amplitude (because $Re(\lambda_k) < 0$). The larger $\max(|Im(\lambda_k)|)$ is, the larger the initial amplitude of the oscillation. Modifying β_1 predominantly changes $Im(\lambda_k)$ while keeping $Re(\lambda_k)$ approximately constant and thus allows independent modification of the stiffness of the system.

This approach of analyzing the dynamics is also applicable for the case of the nonlinear rectification function $f(z) = \max(0, z - T)$ that we use for the main part of the letter. This is because all $z_i > 0$ and the Jacobian matrix (Strogatz, 1994) of this system of equations is equal to J (see equation A.10) since T is a constant.

A.7.1 Ensuring That the Memory State Is an Attractor. In this section, we detail why it is necessary to have two maps ($\gamma > 0$) to have a functioning memory state. From the perspective of the steady state only (see equation A.6), it is sufficient to have $T > 0$ while γ could be zero. However, the dynamics mandate that $\gamma > 0$.

The four eigenvalues of the Jacobian matrix J (see equation A.10) are described by

$$\lambda_i = -1 \pm \frac{1}{2}\gamma + \frac{1}{2}\alpha \pm \frac{1}{2}\sqrt{\gamma^2 \pm 2\gamma + \alpha + \alpha^2 - 4\beta_2\beta_1}. \tag{A.11}$$

Note that two of the $\pm$ signs are in front of terms that disappear in the case of $\gamma = 0$. This results in only two unique eigenvalues, and the node is thus degenerate (and unstable). For the node to be stable, four unique eigenvalues (with the properties described above) are required, and thus it is necessary that $\gamma > 0$.

The trace of J (the sum of all eigenvalues λ_i) equals $2\alpha - 4$ and is independent of γ. It is required that the trace is smaller than 0 (for a state to be stable) and thus $\alpha < 2$.

A.8 Constraints for Bounded Map Activity for Permanently Active Transition Neurons, $I = 0$ and $T > 0$. In the presence of an active transition neuron (TN), the effective recurrent connectivity between the two maps is increased. In this section, we show the constraints on the parameters of the TN such that the map activity remains bounded. In practice, the external input to the TN is active for only a short time. For the purposes of deriving boundaries that guarantee stability, however, we assume a permanently active TN (as a worst-case scenario). Also, we assume that the TN receives input from and projects to map neurons that represent a currently active state. This is relevant for state transformations of symbols that project to the same state (loops). We use the minimal system, as described in section A.7. It consists of two recurrent maps, with one inhibitory (y_1, y_2) and excitatory unit each (x_1, x_2). The two excitatory units on both maps are recurrently connected with weights γ. Here, we add one unit to the system: the TN p_1. It receives input from the excitatory unit of one map x_2 with weight ϕ and projects to the excitatory unit of the other map x_1 with the same weight (see equation 2.9 and Figure 6). Assuming that the threshold T_p equals the amplitude of the external input (as we assume throughout), a permanently active TN has effectively $T_p = 0$ (since input is present). The steady-state amplitude of the excitatory map unit x_1 and x_2 is then described by

$$x_i = \frac{T(\beta_1 - 1)(K + A)}{K^2 - A\gamma^2}, \tag{A.12}$$

with $A = \phi^2 + \gamma$ and K as defined previously. In steady state, $\dot{x}_i = 0$, $\dot{y}_i = 0$, $\dot{p}_1 = 0$, while $x_i > 0$, $y_i > 0$, $p_1 > 0$. Solving for ϕ shows that the following needs to hold:

$$\phi < \sqrt{\frac{K^2 - \gamma^2}{\gamma}}. \tag{A.13}$$

Also, $\phi > 0$. For example, for the numerical values of the weights used in Figure 7, $0 < \phi < 0.8944$.

A.9 Speed of Convergence. The network requires time to reach steady state. The time (and thus the number of iterations of numerical integration) it takes for the network to converge is important because it determines the minimum time an external input (either to the transition neuron or to the state neuron) needs to be applied to ensure a state change.

The rate-limiting step for convergence is the dimension with the smallest absolute eigenvalue: $\min_i |\lambda_i|$. The smaller this value, the longer the system will take to converge. The eigenvalues are a function of the weights (see equation A.11), and the setting of the weights thus has an influence on the speed of convergence. Note that the size of the network is not a factor, since the number of active units is always the same regardless of which state is currently active.

We confirmed numerically how fast the network converges to the application of external input and state switches. We tested random DFAs with 2 to 40 states and found that the time required for the network to converge to the memory state does not depend on the number of states or the distance (on the map) between the states. Euler integration with $\delta = 0.05$ revealed a relaxation time of the following number of iterations: 143 (7.15τ) after initialization, 305 (15.2τ) after a state switch, and 351 (17.6τ) after a loop (state transformation that points to itself). These numbers remain the same regardless of the number of states and transitions represented by the network. The parameters of the network were as shown in Figure 7.

A.10 Summary of Constraints. Two recurrently coupled maps with coupling weight γ, inhibitory loop constants β_1, β_2, and total excitatory input from the same map α have bounded activity and a stable memory if the following conditions are met:

$$\gamma < 1 + \beta_1\beta_2 - \alpha \tag{A.14}$$

$$\beta_1 > 1 \tag{A.15}$$

$$T > 0 \tag{A.16}$$

$$\gamma > 0 \tag{A.17}$$

$$\alpha < 2 \tag{A.18}$$

$$\beta_2 > 0 \tag{A.19}$$

Acknowledgments

We thank the participants of the Neuromorphic Engineering workshop in Cappo Caccia, Sardinia (CNE 2008) for discussion and Robert Rohrkemper for help with figure preparation. This research was funded by the European Union (DAISY project, FP6-2005-015803) and the California Institute of Technology.

References

Abbott, L. (1994). Decoding neuronal firing and modelling neural networks. *Q. Rev. Biophys.*, *27*, 291–331.

Abeles, M. (1991). *Corticonics: Neural circuits of the cerebral cortex*. Cambridge: Cambridge University Press.

Baca, S., Marin-Burgin, A., Wagenaar, D., & Kristan, W. (2008). Widespread inhibition proportional to excitation controls the gain of a leech behavioral circuit. *Neuron*, *57*(2), 276–289.

Barone, P., & Joseph, J. (1989). Prefrontal cortex and spatial sequencing in macaque monkey. *Experimental Brain Research*, *78*, 447–464.

Ben-Yishai, R., Bar-Or, R., & Sompolinsky, H. (1995). Theory of orientation tuning in visual cortex. *Proceedings of the National Academy of Sciences of the USA—Biological Sciences*, *92*(9), 3844–3848.

Binzegger, T., Douglas, R. J., & Martin, K. A. (2004). A quantitative map of the circuit of cat primary visual cortex. *J. Neurosci.*, *24*(39), 8441–8453.

Bongard, J., & Lipson, H. (2005). Active coevolutionary learning of deterministic finite automata. *Journal of Machine Learning Research*, *6*, 1651–1678.

Carrasco, R., Forcada, M., Valdés-Muñoz M. A., & Neco, R. P. (2000). Stable encoding of finite-state machines in discrete-time recurrent neural nets with sigmoid units. *Neural Computation*, *12*(9), 2129–2174.

Cohen, M. A., & Grossberg, S. (1983). Absolute stability of global pattern-formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, *13*(5), 815–826.

Coultrip, R., Granger, R., & Lynch, G. (1992). A cortical model of winner-take-all competition via lateral inhibition. *Neural Networks*, *5*, 47–54.

Dayan, P., & Abbott, L. (2001). *Theoretical neuroscience*. Cambridge, MA: MIT Press.

Diesmann, M., Gewaltig, M., & Aertsen, A. (1999). Stable propagation of synchronous spiking in cortical neural networks. *Nature*, *402*, 529–533.

Douglas, R., Koch, C., Mahowald, M., Martin, K., & Suarez, H. (1995). Recurrent excitation in neocortical circuits. *Science*, *269*(5226), 981–985.

Douglas, R., & Martin, K. (2004). Neuronal circuits of the neocortex. *Annu. Rev. Neurosci.*, *27*, 419–451.

Douglas, R., & Martin, K. (2007). Recurrent neuronal circuits in the neocortex. *Curr. Biol.*, *17*(13), R496–R500.

Elman, J. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, *7*(2), 195–225.

Fujii, N., & Graybiel, A. (2003). Representation of action sequence boundaries by macaque prefrontal cortical neurons. *Science*, *301*, 1246–1249.

Giles, C., Miller, C., Chen, D., Chen, H., Sun, G., & Lee, Y. (1992). Learning and extracting finite state automata with second-order recurrent neural networks. *Neural Computation*, *4*, 393–405.

Grossberg, S. (1973). Contour enhancement, short term memory, and constancies in reverberating neural networks. *Studies in Applied Mathematics*, *52*, 213–257.

Hahnloser, R. (1998). On the piecewise analysis of networks of linear threshold neurons. *Neural Networks*, *11*(4), 691–697.

Hahnloser, R., Douglas, R., Mahowald, M., & Hepp, K. (1999). Feedback interactions between neuronal pointers and maps for attentional processing. *Nat. Neurosci.*, *2*(8), 746–752.

Hahnloser, R., Sarpeshkar, R., Mahowald, M., Douglas, R., & Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, *405*(6789), 947–951.

Hahnloser, R., Seung, H., & Slotine, J. (2003). Permitted and forbidden sets in symmetric threshold-linear networks. *Neural Comput.*, *15*(3), 621–638.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hopcroft, J. (1971). *An n log n algorithm for minimizing states in a finite automaton* (Tech. Rep. CS-TR-71-190). Stanford, CA: Stanford University.

Hopcroft, J., Motwani, R., & Ullman, J. (2000). *Introduction to automata theory, languages, and computation*. Menlo Park, CA: Addison-Wesley.

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of 2-state neurons. *Proceedings of the National Academy of Sciences of the USA—Biological Sciences*, *81*(10), 3088–3092.

Jaeger, H., & Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, *304*, 78–80.

Kalitzin, S., van Dijk, B. W., & Spekreijse, H. (2000). Self-organized dynamics in plastic neural networks: Bistability and coherence. *Biological Cybernetics*, *83*, 139–150.

Kremer, S. C. (1995). On the computational power of Elman-style recurrent networks. *IEEE Transactions on Neural Networks*, *6*(4), 1000–1004.

Kurt, S., Deutscher, A., Crook, J., Ohl, F., Budinger, E., Moeller, C., et al. (2008). Auditory cortical contrast enhancing by global winner-take-all inhibitory interactions. *PLoS ONE*, *3*(3), e1735.

Maass, W. (2000). On the computational power of winner-take-all. *Neural Computation*, *12*, 2519–2536.

Maass, W., Joshi, P., & Sontag, E. (2007). Computational aspects of feedback in neural circuits. *PLOS Computational Biology*, 15–34.

Machens, C., Romo, R., & Brody, C. (2005). Flexible control of mutual inhibition: A neural model of two-interval discrimination. *Science*, *307*(5712), 1121–1124.

McCulloch, W., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, *5*(4), 115–133.

Minksy, L. (1967). *Computation: Finite and infinite machines*. Englewood Cliffs, NJ: Prentice Hall.

Mushiake, H., Inase, M., & Tanji, J. (1990). Selective coding of motor sequence in the supplementary motor area of the monkey cerebral cortex. *Experimental Brain Research*, *82*, 208–210.

Natschlager, T., & Maass, W. (2002). Spiking neurons and the induction of finite state machines. *Theoretical Computer Science*, *287*, 251–265.

Neftci, E., Chicca, E., Indiveri, G., Slotine, J., & Douglas, R. (2008). Contraction properties of VLSI cooperative competitive neural networks of spiking neurons. In J. Platt, D. Koller, Y. Singer, & S. Roweis, (Eds.), *Advances in neural information processing systems*, *20* (pp. 1073–1080). Cambridge, MA: MIT Press.

Omlin, C. W., & Giles, C. L. (1996). Constructing deterministic finite-state automata in recurrent neural networks. *Journal of the ACM*, *43*(6), 937–972.

Pavasovic, A., Andreou, A., & Westgate, C. (1994). Characterization of subthreshold MOS mismatch in transistors for VLSI systems. *Journal of VLSI Signal Processing*, *8*(1), 75–85.

Pouget, A., Zhang, K., Deneve, S., & Latham, P. (1998). Statistically efficient estimation using population coding. *Neural Computation*, *15*(10), 373–401.

Rabinovich, M., Volkovskii, A., Lecanda, P., Huerta, R., Abarbanel, H. D. I., & Laurent, G. (2001). Dynamical encoding by networks of competing neuron groups: Winnerless competition. *Phys. Rev. Lett.*, *87*(6), 068102.

Rao, P. (2004). Bayesian computation in recurrent neural circuits. *Neural Computation*, *16*, 1–38.

Rodger, S., & Finley, T. (2006). *JFLAP—An interactive formal languages and automata package*. Sudbury, MA: Jones and Bartlett.

Seliger, P., Tsimring, L., & Rabinovich, M. (2003). Dynamics-based sequential memory: Winnerless competition of patterns. *Phys. Rev. E*, *67*(1), 011905.

Serrano-Gotarredona, T., & Linares-Barranco, B. (1999). Systematic width-and-length dependent CMOS transistor mismatch characterization and simulation. *Analog Integrated Circuits and Signal Processing*, *21*(3), 271–296.

Seung, H. S., Richardson, T. J., Lagarias, J. C., & Hopfield, J. J. (1998). Minimax and Hamiltonian dynamics of excitatory-inhibitory networks. In M. Jordan, M. Kearns, & S. Solla (Eds.), *Advances in neural information processing systems*, *10*. Cambridge, MA: MIT Press.

Shepherd, G. M., Stepanyants, A., Bureau, I., Chklovskii, D., & Svoboda, K. (2005). Geometric and functional organization of cortical circuits. *Nat. Neurosci.*, *8*(6), 782–790.

Shima, K., & Tanji, J. (2000). Neuronal activity in the supplementary and presupplementary motor areas of temporal organization of multiple movements. *Journal of Neurophysiology*, *84*, 2148–2160.

Strogatz, S. (1994). *Nonlinear dynamics and chaos*. Boulder, CO: Westview Press.

Swadlow, H. (2002). Thalamocortical control of feed-forward inhibition in awake somatosensory barrel cortex. *Philos. Trans. R. Soc. Lond. B Biol. Sci.*, *357*, 1717–1727.

Tomioka, R., Okamoto, K., Furuta, T., Fujiyama, F., Iwasato, T., Yanagawa, Y., et al. (2005). Demonstration of long-range GABAergic connections distributed throughout the mouse neocortex. *European Journal of Neuroscience*, *21*(6), 1587–1600.

von Neumann, J. (1958). *The computer and the brain*. New Haven, CT: Yale University Press.

Wang, X. (2002). Probabilistic decision making by slow reverberation in cortical circuits. *Neuron*, *36*, 955–968.

Wersing, H., Beyn, W. J., & Ritter, H. (2001). Dynamical stability conditions for recurrent neural networks with unsaturating piecewise linear transfer functions. *Neural Computation*, *13*(8), 1811–1825.

Wolfram, S. (1984). Universality and complexity in cellular automata. *Physica D*, *10*(1–2), 1–35.

Xie, X., Hahnloser, R., & Seung, S. (2002). Double-ring network model of the head-direction system. *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.*, *66*(4 Pt. 1), 041902.

Yuille, A., & Grzywacz, N. (1989). A winner-take-all mechanism based on presynaptic inhibition feedback. *Neural Computation*, *1*(3), 334–347.

Zhang, K. (1996). Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: A theory. *Journal of Neuroscience*, *16*, 2112–2126.

Zhaoping, L. (2001). Computational design and nonlinear dynamics of a recurrent network model of the primary visual cortex. *Neural Computation*, *13*(8), 1749–1780.

Received March 24, 2008; accepted June 24, 2008.