Emulation of Finite State Automata with Networks of Synfire Rings

Jérémie Cabessa
Laboratory of Mathematical Economics
Université Panthéon-Assas – Paris 2
75006 Paris, France
Email: jeremie.cabessa@u-paris2.fr

Paolo Masulli
Neuroheuristic Research Group
University of Lausanne
CH-1015 Lausanne, Switzerland
Email: paolo.masulli@unil.ch

***Abstract*—We propose a novel paradigm of neural computation based on synfire rings, i.e., synfire chains that loop back in on themselves. We show that any finite state automaton can be simulated by a Boolean recurrent neural network made up of synfire rings. More precisely, if the given automaton and its corresponding network are run in parallel on a same input stream, then the successive computational states of the automaton are perfectly reflected by the consecutive sustained activities of the network's synfire rings. Our construction turns out to be robust with respect to the removal of a number of connections. These considerations support the idea that a robust paradigm of neural computation based on sustained activities of cell assemblies is indeed possible.**

I. Introduction

According to the computational theory of mind, the mind itself is a computational system. In a less restrictive sense, we assume that at least some aspects of the brain processes are of a computational nature. Therefore, the following questions naturally arise: what are the computational capabilities of the neural networks involved? How does the brain encode and process information?

In this vein, the approach of theoretical computer scientists to neural computation has mainly been focused on trying to simulate neural network models by abstract computing devices, and vice versa, with the aim of understanding the computational capabilities of the neural models.

In fact, it has early been observed that Boolean recurrent neural networks are computationally equivalent to finite state automata [1]–[3]: any Boolean recurrent neural networks can be simulated by some finite state automaton, and more interestingly, any finite automaton can be simulated by some Boolean network. The latter statement is of specific relevance for the implementation of finite state machines on parallel hardware, as well as for the incorporation of prior symbolic knowledge in neural networks, yielding to better learning performances. It opened the way to some important investigations concerning the simulation of finite state machines by various models of neural networks [4]–[19].

Nowadays, the computational capabilities of diverse neural models have been shown to range from the finite automaton degree [1]–[3], [19], up to the Turing [20]–[26] or even to the super-Turing levels [27]–[31]. These kind of studies have also been extended to alternative bio-inspired paradigms of computations [31]–[37].

But in spite of their obvious theoretical relevance, the neural computational paradigms engaged in the above mentioned studies are most probably far from the neurobiological reality. In fact, information is more likely processed by cell assemblies rather than by isolated entities; single neural connections are unreliable; and neural nets are subjected to various biological phenomena, e.g., synaptic plasticity and cell death. In such a more bio-inspired context, the implementations of associative memory tasks, of logical gates, or of abstract devices have been achieved on various kinds of networks of oscillators [38]–[41]. Moreover, sophisticated logical gates have been physically implemented in patterned neural cultures [42], [43].

In terms of information processing in the brain, the concept of *synfire chain* is of specific relevance [44]. These densely connected feedforward networks have been proposed as fundamental structures of biological neural nets, due to their ability to convey repeated complex spatio-temporal patterns of discharges [44], [45]. Such precise neural firing patterns have been observed in various brain areas and in relation with several neural functions [46]–[48]. Moreover, the spontaneous emergence of an abundance of "looping" synfire chains – referred to as *synfire rings* – in self-organizing neural networks subjected to various mechanisms of plasticity has notably been demonstrated [49].

In this work, we propose a novel paradigm of neural computation based on synfire rings. We show that any finite state automaton can be simulated by a Boolean recurrent neural network consisting of synfire rings. More precisely, if the given automaton and its corresponding network are run in parallel on a same input stream, then the successive computational states of the automaton are perfectly reflected by the consecutive sustained activities of the network's synfire rings. In comparison to Minsky's construction [3], ours has the advantage of being robust with respect to the loss of a fraction of the synapses, due to the redundancy of the inter- and intra-ring connections. Our construction is general and can be realized for any finite state automaton. We illustrate this construction for one specific automaton and discuss its robustness.

978-1-5090-6182-2/17/$31.00 ©2017 IEEE

II. Finite State Automata and Boolean Recurrent Neural Networks

We consider Boolean recurrent neural networks (BRNNs), i.e., recurrent neural networks composed of McCulloch and Pitts cells [1]. The dynamics of these networks is computed as follows: given the activation values of the input and internal neurons at time t, denoted by $(u_j(t))_{j=1}^M$ and $(x_j(t))_{j=1}^N$, respectively, one obtains the activation values of the internal neurons at time $t+1$, denoted by $(x_i(t+1))_{i=1}^N$, via the following equations:

$$x_i(t+1) = \theta\left(\sum_{j=1}^{N} a_{ij} \cdot x_j(t) + \sum_{j=1}^{M} b_{ij} \cdot u_j(t) + c_i\right), \quad \text{for } i = 1, \ldots, N \qquad (1)$$

where the a_{ij}, b_{ij}, and c_i are the synaptic weights and bias of the network, and θ is the classical hard-threshold activation function defined by

$$\theta(x) = \begin{cases} 0 & \text{if } x < 1 \\ 1 & \text{if } x \geq 1. \end{cases}$$

These Boolean recurrent neural networks are computationally equivalent to finite state automata [1]–[3]. On the one hand, any such Boolean neural network can be simulated by some finite state automaton, and on the other hand, any finite automaton can also be simulated by some Boolean network. The latter statement is of specific relevance for the implementation of finite state machines on parallel hardware.

More precisely, in Minsky's original construction (nowadays known to be not optimal), a finite automaton with n states and m input symbols is simulated by a Boolean network whose cells are organized in a grid-like manner. The grid structure displays one row and one column of cells per input symbol and computational state of the automaton, respectively. The weighted synaptic connections are then suitably chosen in such a way that, if the automaton and its corresponding network are running in parallel on a same input stream, then the cell of location (i, j) in the network's grid will produce a spike if and only if the automaton is currently receiving the i-th input symbol and visiting the j-th computational state. In this precise sense, the computation of the original automaton is emulated by the spiking pattern of the corresponding network. This translation from a given finite automaton to its corresponding Boolean network is illustrated in detail in Figure 1.

Table I illustrates a simulation of automaton $\mathcal{A}$ of Figure 1 by its corresponding Boolean network $\mathcal{N}$. We see that the consecutive input symbols i and computational states j of $\mathcal{A}$ are correctly reflected by the sequence of spiking cells $c_{i,j}$ of $\mathcal{N}$, with a time delay of 1.

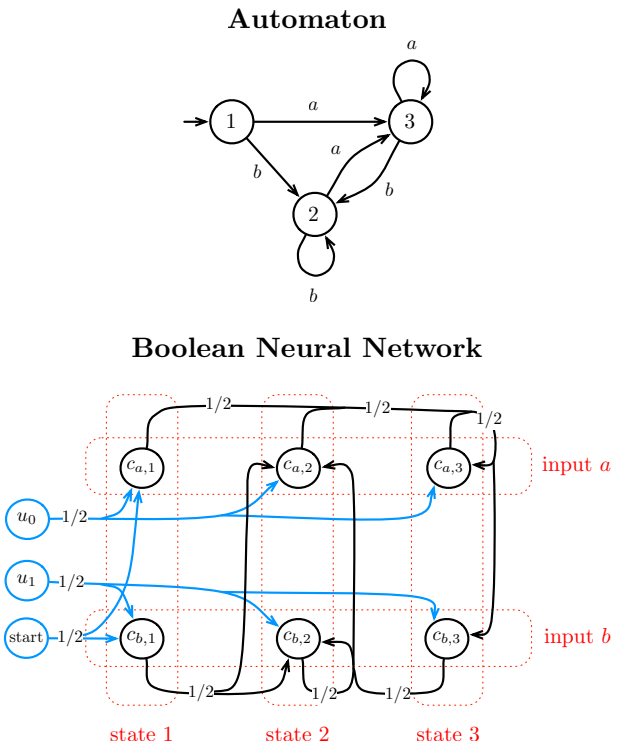

Fig. 1. Translation from a finite state automaton $\mathcal{A}$ to an equivalent Boolean recurrent neural network $\mathcal{N}$. The internal cells of $\mathcal{N}$ are organized in a grid-like structure: the input symbols a and b of $\mathcal{A}$ are associated to the lower and upper rows of $\mathcal{N}$, respectively; the states 1, 2 and 3 of $\mathcal{A}$ are associated to the left, middle and right columns of $\mathcal{N}$, respectively. The fact that $\mathcal{A}$ receives input a or b at time t is reflected by the input cells (u_0, u_1) of $\mathcal{N}$ taking values $(1, 0)$ or $(0, 1)$, respectively. The "start" cell spikes only at time $t = 0$ in order to initiate the dynamics. The weighted synaptic connections are chosen in such a way that, if automaton $\mathcal{A}$ and network $\mathcal{N}$ are running in parallel on a same input stream, then the cell $c_{i,j}$ of $\mathcal{N}$ will produce a spike if and only if $\mathcal{A}$ is currently receiving input symbol i and visiting computational state j. In this very sense, automaton $\mathcal{A}$ is emulated by network $\mathcal{N}$ with a time delay of 1.

TABLE I
Simulation of automaton $\mathcal{A}$ of Figure 1 (top) by its corresponding network $\mathcal{N}$ of Figure 1 (down).

time	0	1	2	3	4	⋯
inputs of $\mathcal{A}$	**a**	**b**	**a**	**a**	**a**	⋯
states of $\mathcal{A}$	**1**	**3**	**2**	**3**	**3**	⋯
cell u_0 of $\mathcal{N}$	1	0	1	1	1	⋯
cell u_1 of $\mathcal{N}$	0	1	0	0	0	⋯
cell *start* of $\mathcal{N}$	1	0	0	0	0	⋯
spiking cell of $\mathcal{N}$	–	$\mathbf{c_{a,1}}$	$\mathbf{c_{b,3}}$	$\mathbf{c_{a,2}}$	$\mathbf{c_{a,3}}$	$\mathbf{c_{a,3}}$

III. Finite State Automata and Networks of Synfire Rings

We now provide an alternative way of emulating finite state automata by Boolean recurrent neural networks composed of synfire rings.

A. General construction

The general idea is to replace each cell $c_{i,j}$ of the Boolean network of Minsky's construction by a synfire chain that

loops back in on itself – referred to as a *synfire ring* – $R_{i,j}$ [49]. In this way, each input symbol and computational state of the original automaton will no more correspond to the punctual activity of a specific single cell, but rather to the sustained activity of a specific synfire ring, that will persist until the appearance of the next input. As a result, every computation of the original automaton will be emulated by a corresponding sequence of sustained activities of synfire rings in the corresponding network. The proposed construction is a general procedure that can be applied to any finite state automaton. It is illustrated in Figure 4.

B. Synfire rings

The so-called *synfire rings* that we consider consist of feedforward neural networks with a same number of cells for each layer. Each cell of each layer is connected to all cells of the next layer (intra-ring connections), and there is no other possible connections, i.e., no connection across non-consecutive layers. The cells of the last layer are connected to those of the first layer. A synfire ring is illustrated in Figure 2.

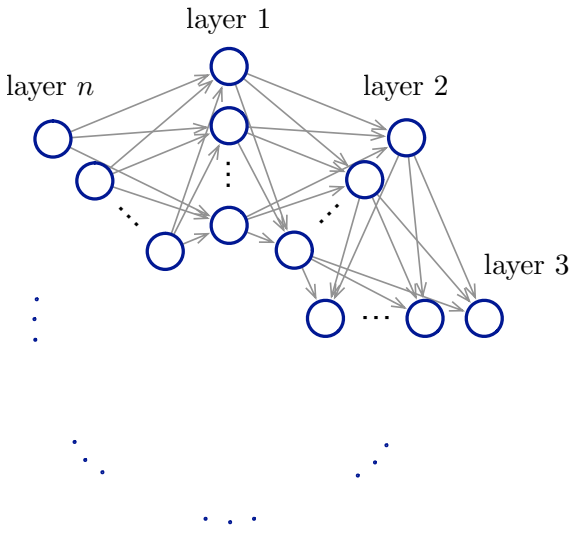

Fig. 2. A synfire ring with n layers. Each cell of each layer is connected to all cells of the next layer. The cells of the last layer are connected to those of the first layer.

C. Input connections

Now, Minsky's construction needs to be adapted for our purpose. First of all, the input connections of Minsky's construction (the blue connections of Figure 1) are replaced by fibres of connections which project from the input cells to the first layer of each of the targeted synfire rings. Their weights are suitably chosen to fulfil the following condition:

(C1) The combined activity of the input cell and inter-ring connections is sufficiently large to activate the targeted synfire ring.

D. Inter-ring connections

Secondly, each excitatory connection between two internal cells $c_{i,j}$ and $c_{i',j'}$ in Minsky's construction (the black connections of Figure 1) is replaced by a fibre of excitatory connections between the corresponding synfire rings $R_{i,j}$ and $R_{i',j'}$ which connects every cells of $R_{i,j}$ to every cells of $R_{i',j'}$ (all-to-all connections). Their excitatory weights are chosen in such a way that the following condition is verified:

(C2) The total activity of a single layer alone does not suffice to activate the cells in the targeted rings.

This means that the internal activity of a synfire ring is enough to sustain the activation from layer to layer within the ring, but not to activate any other synfire ring which it projects onto. The triangular structure discussed below will in fact impose a stronger condition **(C2')** that will replace **(C2)**.

Interestingly, note that the looping connections of Minsky's construction (e.g., connections $(c_{a,3}, c_{a,3})$ and $(c_{b,2}, c_{b,2})$ of Figure 1) do not have to be replaced by corresponding fibres of looping connections: in fact, any synfire ring which is firing would have its activity sustained by its very ring structure, and therefore, doesn't need any additional recurrent connections to remain active.

E. The two-step transition

To replicate the activity of the network in Minsky's construction, we need that, every time that a synfire ring begins to fire, all other rings are switched off, in order for it to remain the only active one. Consequently, for each synfire ring, we add a *triangular structure* as described in Figure 3. The activation of a new synfire ring at time t provokes the activation of its associated triangular structure, which in turn inhibits all the other rings at time $t+1$ by means of its red cell. At time $t+2$, the red cell as well as all the other rings are therefore shut off. An example of such a transition in a network is shown in Figure 7.

Note that the inhibitory weights of the red dotted connections need to be chosen so that the following condition holds:

(C3) The weights of the inhibitory connections projecting from the inhibitory cell of a triangular structure to the other synfire rings must be sufficiently negative to inhibit the total activity of one layer of the rings onto which they project.

Hence, those weights depend on the layer size as well as of the weights of the intra-ring connections. Note that during the transition from $R_{i,j}$ to $R_{i',j'}$, there are two time steps at which the two rings are simultaneously firing, see Figure 7. Depending on the structure of the inter-ring connections, it might happen that both $R_{i,j}$ and $R_{i',j'}$ are projecting onto the same third ring in the network (e.g. in our example, both $R_{a,2}$ and $R_{a,3}$ are projecting a fibre of connections to the ring $R_{b,3}$). Consequently, the excitatory inter-ring weights must be chosen in such a way that they fulfil the following stronger version of condition **(C2)**:

(C2') The combined activity projecting from any two layers of two synfire rings does not suffice to activate the cells of a third ring.

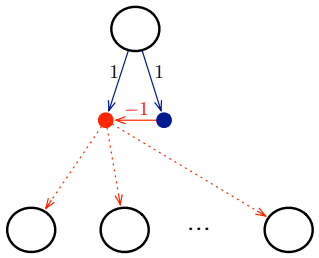

Fig. 3. The triangular structure associated to each synfire ring. Each large node represents a synfire rings and each little node represents a single cell. The two downward solid edges of the structure do not represent single connections, but fibres of excitatory connections of weight 1 which project from every cells of the upper chain to the blue and red units. The horizontal red solid edge is a single inhibitory connection of weight -1. The downward red dotted edges represent fibres of sufficiently large inhibitory connections which project from the red unit to every cells of the targeted synfire ring. If the upper chain fires at time t, it activates both red and blue cells at time $t+1$. Consequently, at time $t+2$, all other synfire rings, represented as the lower nodes, are inhibited via the red connections, and the red cell is also inhibited via the horizontal red connection. At next time steps, as long as the upper chain keeps firing, the red cell is kept inhibited by the horizontal red connection, and therefore, has no more inhibition effect on the lower chains.

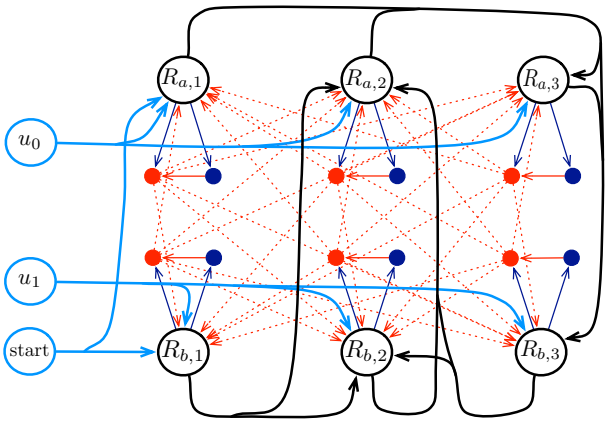

Fig. 4. Boolean recurrent neural network $\mathcal{N}'$ made of synfire rings which simulates the automaton of Figure 1. Each large node represents a synfire ring. To each synfire ring is associated a triangular structure as described in Figure 3. The emulation is performed as follows: the synfire ring $R_{i,j}$ in the grid – and only this one – will fire if and only if the automaton receives input symbol i and is in the j-th computational state.

F. Emulation of Finite State Automata

By replacing each node of Minsky's construction by a *synfire ring* and by adding a *triangular structure* associated to each such ring, one obtains a recurrent network – made of synfire rings – which can simulate any possible finite state automaton. The weights of the connections have to be suitably chosen according to conditions **(C1)**, **(C2')**, and **(C3)**, as explained above. Note that such a choice is always possible.

In fact, any finite state automaton is emulated by its corresponding network of synfire rings in the following sense: when the two systems are run in parallel on a same input stream, the synfire ring of the network of location (i, j) in the grid – and only this one – will fire at a certain time step if and only if the automaton is currently receiving the i-th input symbol and visiting the j-th computational state. Moreover, the activity of that specific ring is self-sustained as long as no other input is received. For such networks, the successive input patterns have to be provided at sufficiently distant time steps, rather than at consecutive time steps, in order for the synfire rings' activities to settle, via the two-step transition illustrated above.

More precisely, in order to simulate one time step of the automaton, one needs at least three time steps of its corresponding network, in order for the two-step transition to be completed. Consequently, the proposed simulation process works in linear time.

In summary, one has the following result:

Theorem 1. *Any finite state automaton can be emulated by some network of synfire rings.*

Table II illustrates a simulation of automaton $\mathcal{A}$ of Figure 1 by its corresponding Boolean network of synfire rings $\mathcal{N}'$ of Figure 4. We see that the consecutive input symbols i and computational states j of $\mathcal{A}$ are correctly reflected by the sequence of sustained activities of the synfire rings $R_{i,j}$ of $\mathcal{N}'$. The implementation of network $\mathcal{N}'$ and its detailed activity during this simulation process are illustrated in Figures 5 and 7.

TABLE II
SIMULATION OF AUTOMATON $\mathcal{A}$ OF FIGURE 1 BY ITS CORRESPONDING NETWORK OF SYNFIRE RINGS $\mathcal{N}'$ OF FIGURE 4.

inputs of $\mathcal{A}$	**a**	**b**	**a**	**a**	**a**	···
states of $\mathcal{A}$	**1**	**3**	**2**	**3**	**3**	···
cell u_0 of $\mathcal{N}'$	1	0	1	1	1	···
cell u_1 of $\mathcal{N}'$	0	1	0	0	0	···
cell *start* of $\mathcal{N}'$	1	0	0	0	0	···
synfire ring of $\mathcal{N}'$	–	$\mathbf{R_{a,1}}$	$\mathbf{R_{b,3}}$	$\mathbf{R_{a,2}}$	$\mathbf{R_{a,3}}$	$\mathbf{R_{a,3}}$

IV. METHODS

A. Implementation

The network of synfire rings $\mathcal{N}'$ of Figure 4, which simulates the finite state automaton $\mathcal{A}$ of Figure 1 (top panel), has been implemented in Python, using the package `igraph` [50]. Figures 5 and 7 illustrate this network and snapshots of its activity. Network $\mathcal{N}'$ is composed of 6 synfire rings, each of which having 6 layers. Each layer is formed by 3 Boolean cells with threshold 1, and each cell in a layer is connected to all the cells in the following one. There are two input cells, u_0 and u_1, and one *start* cell, which is firing only once in correspondence with the first input. The weights of the input connections have been set to 0.9, and the those of the inter-ring connections are set to 0.1. The inhibitory weights projecting from each triangular structure are set to -4. These weights are chosen in order to satisfy the required conditions **(C1)**, **(C2')**, and **(C3)** of Section III. In this way, automaton $\mathcal{A}$ is correctly emulated by network $\mathcal{N}'$.

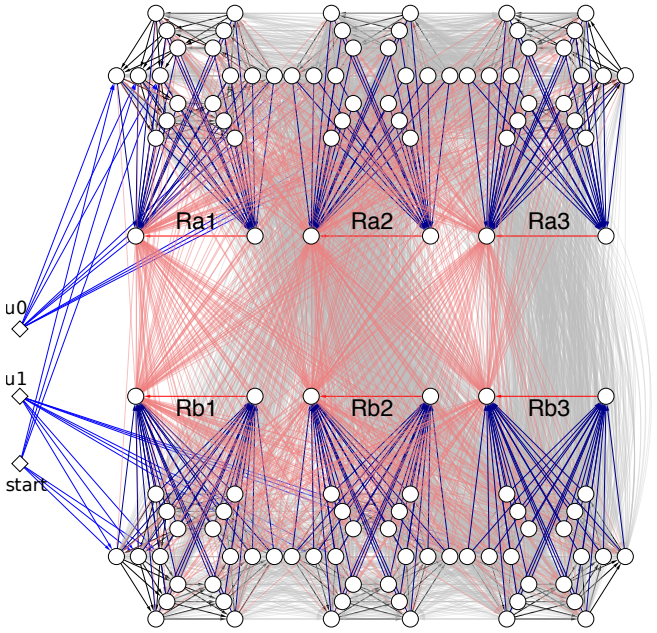

Fig. 5. Implementation of the network of synfire rings of Figure 4 which simulates of the finite state automaton of Figure 1. The network is composed of 6 synfire rings, each of which having 6 layers. Each layer is formed by 3 Boolean cells with threshold 1. There are two input cells, u_0 and u_1, and one *start* cell, which is supposed to firing only once in correspondence with the first input. The light blue, dark blue, red and grey connections correspond to the light blue, dark blue, red and black fibres of Figure 4. The intra-ring connections are also displayed in grey.

B. Robustness

We simulated the activity of network $\mathcal{N}'$ of Figure 5 after removing a certain number of connections. In the first condition, we removed *inter-ring* connections, and in the second, we removed *intra-ring* connections. The fraction of removed connections in each case was determined by a connection removal coefficient (real value in $[0, 1]$), which was increased by increments of 0.05. For each condition, we generated 40 networks by randomly selecting, with different random seeds, the connections to remove. The networks were fed with the same sequence of inputs of length 80. The sequence of states (i.e. synfire rings) activated in the network was compared with that of $\mathcal{N}'$. We calculated how many steps in the sequence of visited states coincided with those visited by $\mathcal{N}'$, and expressed the performance of the network by dividing this value by the total number of states visited by $\mathcal{N}'$.

V. Results

We showed that the network of synfire rings $\mathcal{N}'$ correctly emulates the finite automaton $\mathcal{A}$, as explained in Section III. Four successive steps of the dynamics of $\mathcal{N}'$ are shown in Figure 7. The activity of $\mathcal{N}'$ fed with a particular input sequence is displayed as a raster plot in Figure 6.

One upshot of our construction with respect to Minsky's is its robustness: the strongly-interconnected nature of synfire rings and the redundancy of connections ensures that the emulation of the automaton by the network is correctly performed, even if a significant number of connections are removed.

The results are displayed in Table III and IV. Concerning inter-ring connections, for values of the connection removal coefficient below 0.60, no difference caused by the removal of connections could be observed. Moreover, up to very high values of the coefficients, the networks with removed connections were able to achieve the same activation patterns as the original network, showing the high robustness of our construction.

Concerning intra-ring connections, signs of malfunctioning of the networks with removed connections started to appear for values of the connection removal coefficient around 0.25, with then the performance decreasing to low values when the coefficient was increased. This reflects the fact that the robustness of the construction depends strongly on the dense internal connectivity of the synfire-ring structures.

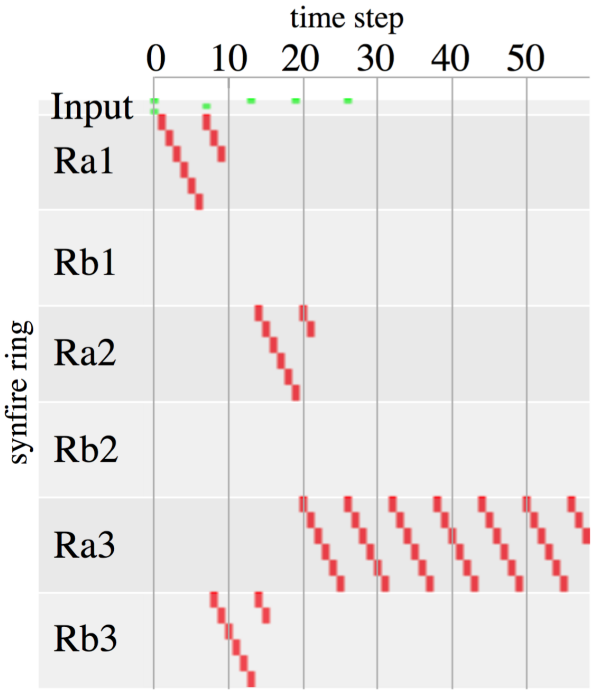

Fig. 6. Raster plot of the activity of network $\mathcal{N}'$ simulating automaton $\mathcal{A}$ of Figure 1. The row labelled as "Input" shows the activity, from top to bottom, of the input and cells u_0, u_1 and *start*. The following blocks of rows show the activity of the six synfire rings. The network is activated with the input sequence $(1,0)^T(0,1)^T(1,0)^T(1,0)^T(1,0)^T$ corresponding to input $abaaa$ of the automaton. Note that, for the network to activate correctly, it is necessary to provide input signals that are sufficiently spaced in time, in order for the two-step transition from one synfire ring to the next to complete. The time steps from $t = 7$ to $t = 10$ correspond to the two-step transition displayed in Figure 7. After the network received the last input signal in the sequence (input $(1,0)^T$ at time $t = 27$), it remains permanently with the synfire ring $\mathbf{R_{a,3}}$ activated, since the activity of synfire rings is self-sustained.

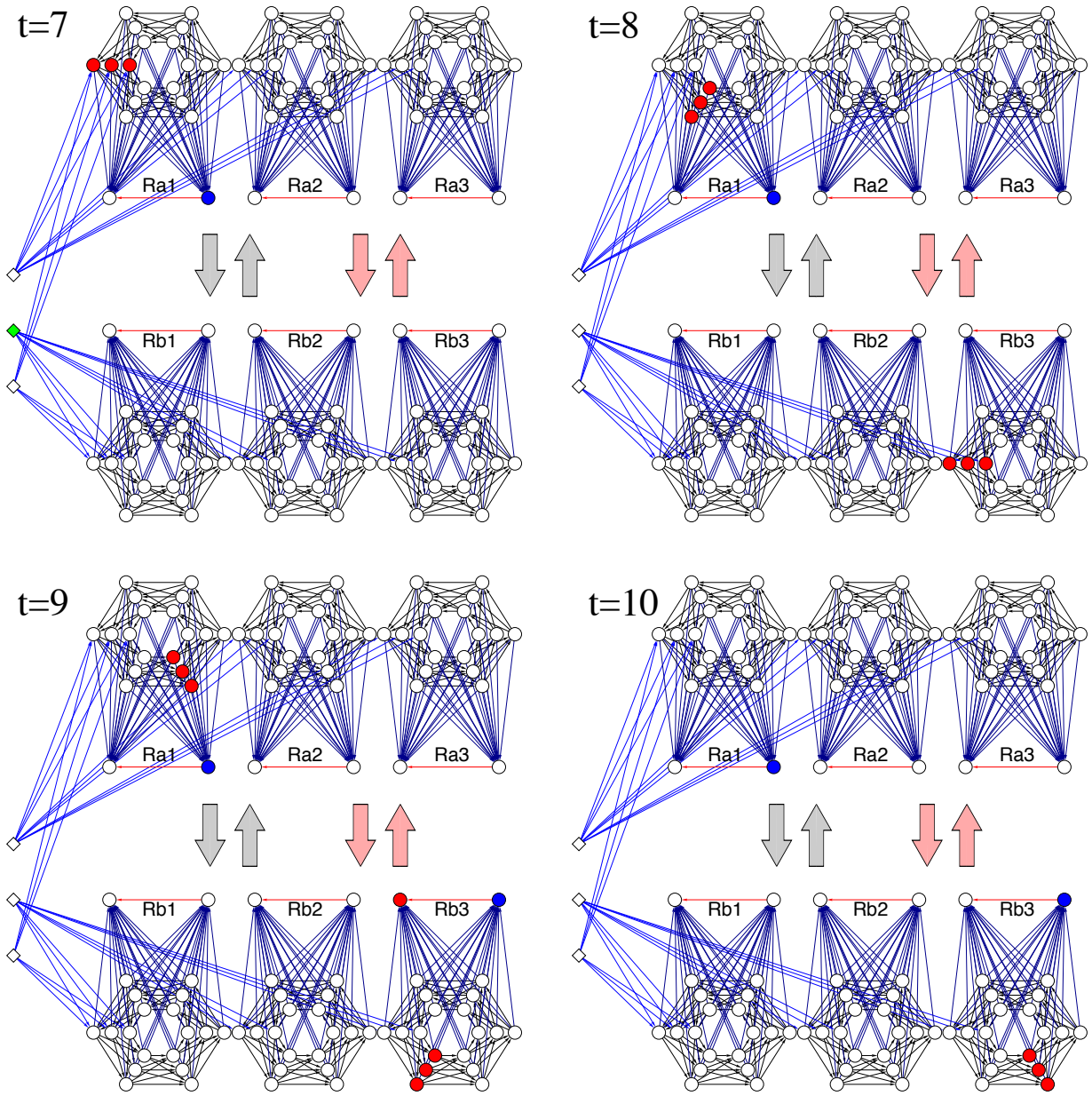

Fig. 7. **The two-step transition.** Four time steps of activity of the network of Figure 5. The transition from one synfire ring to another (two-step transition) is illustrated. In the four successive snapshots, the inter-ring connections (grey in Figure 5) and the inhibitory connections originating from the triangular structures (red in Figure 5) have been omitted for readability. The big solid grey and red arrows are meant as a reminder of their presence. At time $t = 7$ (and in the time steps leading to it), the network has a stable activity where only one synfire ring is active, namely $\mathbf{R_{a,1}}$. At the said time step, the input pattern $(u_0, u_1) = (1, 0)$, corresponding to input $\mathbf{a}$ of the automaton, is received. The input signal combined with the synfire ring action causes the activation of the initial layer of the synfire ring $\mathbf{R_{b,3}}$ at time $t = 8$. Note that at this point, two synfire rings are simultaneously active. The triangular structure has the function of inhibiting the ring that was previously active. In fact, at time $t = 9$, the activity of $\mathbf{R_{b,3}}$ activates the two cells in its triangular structure. The red cell projects inhibitory synapses towards all the other synfire rings, causing the inhibition of the synfire ring $\mathbf{R_{a,1}}$ at time $t = 10$. At the same time, the blue cell of the triangular structure inhibits the red cell at time $t = 10$, switching off its inhibitory synapses and completing the two-step transition. This brings the network to another stable state of self-sustained activity with only one active synfire ring, namely $\mathbf{R_{b,3}}$.

VI. Conclusion

We introduced a novel paradigm of neural computation based on synfire rings, i.e., synfire chains that loop back in on themselves. We showed that any finite state automaton can be emulated by a Boolean recurrent neural network consisting of such synfire rings. The construction turns out to be robust with respect to the removal of synaptic connections.

The robustness of our construction relies on the large number of connections involved in our architecture. The trade-off is a greater computational cost in comparison to other works [3]–[19]. On the other hand, the time complexity of our simulation process is linear. But our interest lies beyond

complexity issues. We intend to show that a robust paradigm of neural computation based on sustained activities of cell assemblies is indeed possible.

Our results support the evidence that synfire chains play a significant role in the processing and coding of information in neural networks [44], [45]. They are also in line with recent findings showing that self-organizing networks spontaneously develop an abundance of synfire rings in their structure [49].

For future work, we plan to extend the present considerations towards a Turing complete paradigm of computation based on networks of synfire rings.

TABLE III
ROBUSTNESS OF THE CONSTRUCTION WHEN INTER-RING CONNECTIONS ARE REMOVED.

Connection removal coefficient	Average number of steps	Performance
0.50	12.00	100.0%
0.55	12.00	100.0%
0.60	12.00	100.0%
0.65	11.45	95.4%
0.70	11.20	93.3%
0.75	10.30	85.8%
0.80	7.58	63.1%
0.85	6.85	57.1%
0.90	4.17	34.8%
0.95	3.75	31.2%

TABLE IV
ROBUSTNESS OF THE CONSTRUCTION WHEN INTRA-RING CONNECTIONS ARE REMOVED.

Connection removal coefficient	Average number of steps	Performance
0.00	12.00	100.0%
0.05	12.00	100.0%
0.10	12.00	100.0%
0.15	12.00	100.0%
0.20	12.00	100.0%
0.25	11.78	98.1%
0.30	11.72	97.7%
0.35	10.35	86.2%
0.40	10.05	83.8%
0.45	7.58	63.1%
0.50	6.70	55.8%
0.55	4.62	38.5%
0.60	3.05	25.4%
0.65	2.15	17.9%
0.70	1.57	13.1%
0.75	1.40	11.7%
0.80	1.12	9.4%
0.85	1.10	9.2%
0.90	1.02	8.5%
0.95	1.00	8.3%

REFERENCES

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysic*, vol. 5, pp. 115–133, 1943.

[2] S. C. Kleene, "Representation of events in nerve nets and finite automata," in *Automata Studies*, C. Shannon and J. McCarthy, Eds. Princeton, NJ: Princeton University Press, 1956, pp. 3–41.

[3] M. L. Minsky, *Computation: finite and infinite machines*. Englewood Cliffs, N. J.: Prentice-Hall, Inc., 1967.

[4] A. Cleeremans, D. Servan-Schreiber, and J. L. McClelland, "Finite state automata and simple recurrent networks," *Neural Computation*, vol. 1, no. 3, pp. 372–381, 1989.

[5] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[6] N. Alon, A. K. Dewdney, and T. J. Ott, "Efficient simulation of finite automata by neural nets," *J. ACM*, vol. 38, no. 2, pp. 495–514, 1991.

[7] J. B. Pollack, "The induction of dynamical recognizers," *Machine Learning*, vol. 7, pp. 227–252, 1991.

[8] C. L. Giles, C. B. Miller, D. Chen, H. Chen, G. Sun, and Y. Lee, "Learning and extracting finite state automata with second-order recurrent neural networks," *Neural Computation*, vol. 4, no. 3, pp. 393–405, 1992.

[9] R. L. Watrous and G. M. Kuhn, "Induction of finite-state languages using second-order recurrent networks," *Neural Computation*, vol. 4, no. 3, pp. 406–414, 1992.

[10] Z. Zeng, R. M. Goodman, and P. Smyth, "Learning finite state machines with self-clustering recurrent networks," *Neural Computation*, vol. 5, no. 6, pp. 976–990, 1993.

[11] M. W. Goudreau, C. L. Giles, S. T. Chakradhar, and D. Chen, "First-order versus second-order single-layer recurrent neural networks," *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 511–513, 1994.

[12] R. Alquézar and A. Sanfeliu, "An algebraic framework to represent finite state machines in single-layer recurrent neural networks," *Neural Computation*, vol. 7, no. 5, pp. 931–949, 1995.

[13] P. Frasconi, M. Gori, M. Maggini, and G. Soda, "Unified integration of explicit knowledge and learning by example in recurrent networks," *IEEE Trans. Knowl. Data Eng.*, vol. 7, no. 2, pp. 340–346, 1995.

[14] S. C. Kremer, "On the computational power of elman-style recurrent networks," *Neural Networks, IEEE Transactions on*, vol. 6, no. 4, pp. 1000–1004, 1995.

[15] P. Frasconi, M. Gori, M. Maggini, and G. Soda, "Representation of finite state automata in recurrent radial basis function networks," *Machine Learning*, vol. 23, no. 1, pp. 5–32, 1996.

[16] B. G. Horne and D. R. Hush, "Bounds on the complexity of recurrent neural network implementations of finite state machines," *Neural Networks*, vol. 9, no. 2, pp. 243–252, 1996.

[17] C. W. Omlin and C. L. Giles, "Stable encoding of large finite-state automata in recurrent neural networks with sigmoid discriminants," *Neural Computation*, vol. 8, no. 4, pp. 675–696, 1996.

[18] ——, "Constructing deterministic finite-state automata in recurrent neural networks," *J. ACM*, vol. 43, no. 6, pp. 937–972, 1996.

[19] H. T. Siegelmann, "Recurrent neural networks and finite automata," *Computational Intelligence*, vol. 12, pp. 567–574, 1996.

[20] A. M. Turing, "Intelligent machinery," National Physical Laboratory, Teddington, UK, Technical Report, 1948.

[21] J. B. Pollack, "On connectionist models of natural language processing," Ph.D. dissertation, Computing Reseach Laboratory, New Mexico State University, Las Cruces, NM, 1987.

[22] R. Hartley and H. Szu, "A comparison of the computational power of neural network models," in *Proceedings of the IEEE First International Conference on Neural Networks*, C. Butler, Ed. IEEE, 1987, pp. 17–22.

[23] H. T. Siegelmann and E. D. Sontag, "On the computational power of neural nets," *J. Comput. Syst. Sci.*, vol. 50, no. 1, pp. 132–150, 1995.

[24] J. Kilian and H. T. Siegelmann, "The dynamic universality of sigmoidal neural networks," *Inf. Comput.*, vol. 128, no. 1, pp. 48–56, 1996.

[25] H. Hyötyniemi, "Turing machines are recurrent neural networks," in *STeP '96 - Genes, Nets and Symbols; Finnish Artificial Intelligence Conference, Vaasa 20-23 Aug. 1996*, J. Alander, T. Honkela, and J. M., Eds. Vaasa, Finland: University of Vaasa, Finnish Artificial Intelligence Society (FAIS), 1996, pp. 13–24.

[26] J. a. P. G. Neto, H. T. Siegelmann, J. F. Costa, and C. P. S. Araujo, "Turing universality of neural nets (revisited)," in *EUROCAST '97: Proceedings of the A Selection of Papers from the 6th International*

Workshop on Computer Aided Systems Theory. London, UK: Springer-Verlag, 1997, pp. 361–366.
[27] H. T. Siegelmann and E. D. Sontag, "Analog computation via neural networks," *Theor. Comput. Sci.*, vol. 131, no. 2, pp. 331–360, 1994.
[28] J. L. Balcázar, R. Gavaldà, and H. T. Siegelmann, "Computational power of neural networks: a characterization in terms of kolmogorov complexity," *IEEE Transactions on Information Theory*, vol. 43, no. 4, pp. 1175–1183, 1997.
[29] H. T. Siegelmann, "Neural and super-Turing computing," *Minds Mach.*, vol. 13, no. 1, pp. 103–114, 2003.
[30] J. Cabessa and H. T. Siegelmann, "Evolving recurrent neural networks are super-Turing," in *Proceedings of IJCNN 2011*. IEEE, 2011, pp. 3200–3206.
[31] ——, "The super-Turing computational power of plastic recurrent neural networks," *Int. J. Neural Syst.*, vol. 24, no. 8, 2014.
[32] ——, "The computational power of interactive recurrent neural networks," *Neural Computation*, vol. 24, no. 4, pp. 996–1019, 2012.
[33] J. Cabessa and A. E. P. Villa, "The expressive power of analog recurrent neural networks on infinite input streams," *Theor. Comput. Sci.*, vol. 436, pp. 23–34, 2012.
[34] ——, "An attractor-based complexity measurement for boolean recurrent neural networks," *PLoS ONE*, vol. 9, no. 4, pp. e94 204+, 2014.
[35] J. Cabessa and A. E. Villa, *Artificial Neural Networks: Methods and Applications in Bio-/Neuroinformatics*. Springer International Publishing, 2015, ch. Recurrent Neural Networks and Super-Turing Interactive Computation, pp. 1–29.
[36] J. Cabessa and J. Duparc, "Expressive power of nondeterministic recurrent neural networks in terms of their attractor dynamics," *IJUC*, vol. 12, no. 1, pp. 25–50, 2016.
[37] J. Cabessa and A. E. P. Villa, "Expressive power of first-order recurrent neural networks determined by their attractor dynamics," *J. Comput. System Sci.*, vol. 82, no. 8, pp. 1232–1250, 2016.
[38] F. C. Hoppensteadt and E. M. Izhikevich, "Synchronization of laser oscillators, associative memory, and optical neurocomputing," *Phys. Rev. E*, vol. 62, pp. 4010–4013, Sep 2000.
[39] D. Xu, J. C. Principe, and J. G. Harris, "Logic computation using coupled neural oscillators," in *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, vol. 5, May 2004, pp. V–788–V–791 Vol.5.
[40] M. Zanin, F. Del Pozo, and S. Boccaletti, "Computation emerges from adaptive synchronization of networking neurons," *PLoS ONE*, vol. 6, no. 11, pp. 1–6, 11 2011.
[41] D. Malagarriga, M. A. García-Vellisca, A. E. P. Villa, J. M. Buldú, J. García-Ojalvo, and A. J. P. Rivero, "Synchronization-based computation through networks of coupled oscillators," *Front. Comput. Neurosci.*, vol. 9, no. 97, 2015.
[42] O. Feinerman, A. Rotem, and E. Moses, "Reliable neuronal logic devices from patterned hippocampal cultures," *Nat. Phys.*, vol. 4, no. 12, pp. 967–973, 2008.
[43] F. Wolf and T. Geisel, "Neurophysics: Logic gates come to life," *Nat. Phys.*, vol. 4, no. 12, pp. 905–906, 2008.
[44] M. Abeles, *Local Cortical Circuits. An Electrophysiological Study*, ser. Studies of Brain Function. Berlin Heidelberg New York: Springer-Verlag, 1982, vol. 6.
[45] ——, "Time is precious," *Science*, vol. 304, no. 5670, pp. 523–524, 2004.
[46] Y. Prut, E. Vaadia, H. Bergman, I. Haalman, H. Slovin, and M. Abeles, "Spatiotemporal structure of cortical activity: Properties and behavioral relevance," *Journal of Neurophysiology*, vol. 79, no. 6, pp. 2857–2874, 1998.
[47] T. Shmiel, R. Drori, O. Shmiel, Y. Ben-Shaul, Z. Nadasdy, M. Shemesh, M. Teicher, and M. Abeles, "Neurons of the cerebral cortex exhibit precise interspike timing in correspondence to behavior," *Proc. Natl. Acad. Sci. USA*, vol. 102, no. 51, pp. 18 655–18 657, 2005.
[48] A. E. P. Villa, "Empirical Evidence about Temporal Structure in Multiunit Recordings," in *Time and the brain*, ser. Conceptual Advances in Brain Research, R. Miller, Ed. Amsterdam, The Netherlands: Harwood Academic, 2000, vol. 3, ch. 1, pp. 1–51.
[49] P. Zheng and J. Triesch, "Robust development of synfire chains from multiple plasticity mechanisms," *Front. Comput. Neurosci.*, vol. 8, no. 66, 2014.
[50] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal*, vol. Complex Systems, p. 1695, 2006. [Online]. Available: http://igraph.org