Robust Optimal-Size Implementation of Finite State Automata with Synfire Ring-Based Neural Networks

Jérémie Cabessa[1,2(✉)] and Jiří Šíma[2]

[1] Laboratory of Mathematical Economics and Applied Microeconomics, University Paris 2 – Panthéon-Assas, 4, Rue Blaise Desgoffe, 75006 Paris, France
jeremie.cabessa@u-paris2.fr

[2] Institute of Computer Science, Czech Academy of Sciences, P. O. Box 5, 18207 Prague 8, Czech Republic
sima@cs.cas.cz

Abstract. Synfire rings are important neural circuits capable of conveying synchronous, temporally precise and self-sustained activities in a robust manner. We describe a robust and optimal-size implementation of finite state automata with neural networks composed of synfire rings. More precisely, given any finite automaton, we build a corresponding neural network partly composed of synfire rings and capable of simulating it. The synfire ring activities encode the successive states of the automaton throughout its computation. The robustness of the network results from its architecture, which involves synfire rings and duplicated core components. We finally show that the network's size is asymptotically optimal: for an automaton with n states, the network has $\Theta(\sqrt{n})$ cells.

Keywords: Recurrent neural networks · Threshold circuits · Finite state automata · Synfire rings

1 Introduction

In theoretical neural computation, the computational capabilities of various neural models has been shown to range from the finite automaton degree, up to the Turing, or even to the super-Turing levels (see the thorough survey [24]). In summary, Boolean recurrent neural networks are computationally equivalent to finite state automata [15,19]; sigmoidal rational-weighted neural networks are Turing complete [22]; and sigmoidal real-weighted and evolving neural networks are super-Turing powerful [5,21].

Supports from DARPA – Lifelong Learning Machines (L2M) program, cooperative agreement No. HR0011-18-2-0023, as well as from the ICS CAS RVO: 67985807 and the Czech Science Foundation, grant No. 19-05704S, are gratefully acknowledged.

In the 90's, the equivalence between Boolean neural networks and finite state automata has been extensively studied, motivated by the possibility to implement abstract machines on parallel hardwares. In particular, it has been shown that any deterministic automaton with n states can be implemented by a neural network of optimal size containing $\Theta(\sqrt{n})$ neurons [11,13]. The energy complexity of this network construction can be minimized without changing its optimal size [23]. Furthermore, any regular language described by a regular expression of length ℓ can be recognized by an optimal-size neural network having $\Theta(\ell)$ units [25].

But the neural models involved in these studies fail to capture biological features that are so essential to brain information processing. For instance, the computational behaviors of those networks do certainly comply with the paradigms of computation of biological neural systems: the computational states are represented by discrete (spiking) configurations of the networks, rather than by sustained and temporally robust activities of cell assemblies. Also, the networks' dynamics is not robust to the possibility of architectural failures.

In biology, the concept of synfire chains and synfire rings have been demonstrated to play significant roles in the processing and coding of information in the brain. *Synfire chains* are feedforward neural circuits whose every layer is connected to the next by means of excitatory convergent/divergent synaptic patterns [1,2,7,12,18]. According to this architecture, the neurons of each layer tend to fire simultaneously, and the firing activity propagates through the successive layers in a synchronized manner. Hence, synfire chains are able to convey repeated complex spatiotemporal patterns of discharges in a robust and highly temporally precise way. *Synfire rings* are looping synfire chains [16,26]. As an additional dynamical feature, the ring shape gives rise to self-sustained activities, which correspond to attractor dynamics. Synfire chains and rings have been shown to spontaneously emerge in self-organizing networks subjected to various kinds of synaptic plasticity (see for instance [8,9,14,16,26]).

Based on these considerations, it has been shown that finite state automata can be simulated by Boolean recurrent neural networks composed of synfire rings [4]. The results have then been generalized to the more biological cases of networks of Izhikevich spiking neurons [3], and even to Hodgkin-Huxley neurons [6]. The obtained architecture is, to a certain extent, robust to synaptic pruning as well as to the introduction of synaptic noises.

Here, we extend these results by describing a robust optimal-size implementation of finite state automata with synfire ring-based neural networks. The paper is organized as follows. Section 2 introduces the concepts of Boolean neural networks and synfire rings. Section 3 recalls the definition of finite state automata, presents the simulation result of finite automata by optimal-size threshold circuits [17], and describes the generalization of this construction to the context of Boolean neural networks [11]. Section 4 contains our results. Given any finite automaton, we build a robust and optimal-size neural network partly composed of synfire rings capable of simulating it. The synfire ring activities encode the successive states of the automaton throughout its computation. The robustness of the network results from its architecture, which is composed of synfire rings and duplicated core components. Based on previous work [11,17], we show that the network's size is asymptotically optimal: for an automaton with n states, the network has $\Theta(\sqrt{n})$ cells. The implementation of this construction is deferred to an extended journal version of this paper. Finally, Sect. 5 offers a brief conclusion.

2 Neural Networks and Synfire Rings

Boolean Neural Networks. A *Boolean recurrent neural network (BRNN)* $\mathcal{N}$ consists of a synchronous network of Boolean cells related together in a general architecture. The network is composed of M input neurons $(u_i)_{i=1}^M$ and N internal neurons $(x_i)_{i=1}^N$. The dynamics of network is computed as follows: given the activation values of the input neurons $(u_j(t))_{j=1}^M$ and internal neurons $(x_j(t))_{j=1}^N$ at time step t, the activation values of the internal neurons $(x_i(t+1))_{i=1}^N$ at time step $t+1$ are given by the following equations:

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

where $a_{ij} = w(x_j, x_i)$ and $b_{ij} = w(u_j, x_i)$ are the *weights* of the synaptic connections from x_j to x_i and from u_j to x_i, respectively, c_i is the *bias* of cell x_i, and θ is the *hard-threshold* activation function defined by

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

Neural networks can be exploited as acceptors of formal languages (here, we consider languages over the alphabet $\{0, 1\}$) [20]. Towards this purpose, several input/output protocols have been proposed in the literature. Here, a so-called offline input/output is considered. The Boolean networks are provided with two input cells called `inp` and `val`, as well as with a specific internal cell called `out`. The neurons `inp` is used to transmit the input strings (words) to the network in a sequential way, i.e., bit by bit. The neuron `out` outputs the decisions of the network to accept or reject its inputs. The cell `val` is used to identify the time steps at which new input bits are received.

Formally, suppose that the input (string) $x = x_0 \cdots x_m \in \{0,1\}^*$ is to be processed by the BRNN $\mathcal{N}$. Assume further that the successive bits of x are presented to the network at successive time steps $0 < t_0 < t_1 < \cdots < t_m$ separated by at least $d \geq 4$ units of time, i.e., $t_{i+1} - t_i \geq d$ for every $i = 0, \ldots, m-1$. The processing of input x is implemented as follows. The activations values of `inp`, `val` are externally set to the following values

$$\texttt{inp}(t) = \begin{cases} x_i & \text{if } t = t_i \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \texttt{val}(t) = \begin{cases} 1 & \text{if } t = t_i \\ 0 & \text{otherwise} \end{cases}$$

for all $t \geq 0$. Now, let $t^* = t_m + d + 1$. We say that x is *accepted* (resp. *rejected*) by $\mathcal{N}$ iff $\texttt{out}(t^*) = 1$ (resp. $\texttt{out}(t^*) = 0$). The set of words accepted by $\mathcal{N}$ is the *language recognized by* $\mathcal{N}$, denoted by $L(\mathcal{N})$. A language L is *recognizable* by some BRNN if there exists some $\mathcal{N}$ such that $L = L(\mathcal{N})$.

Synfire Rings. A *synfire ring* R of width $w \geq 1$ and length $\ell \geq 2$ is a specific BRNN composed of $\ell \cdot w$ cells $(x_{ij})_{i=1,j=1}^{w,\ell}$. For every $j = 1, \ldots, \ell$, the cells $x_{1j}, \ldots, x_{wj}$ is the *j-th layer* of R, and for every $i = 1, \ldots, w$, the cells $x_{i1}, \ldots, x_{i\ell}$ form the *i-th level* of R. For every $i = 1, \ldots, w-1$, each cell of the i-th layer is connected to all cells of the $(i+1)$-th layer with connections of weight 1. Also, each cell of the ℓ-th layer is connected to all cells of the 1-st layer with connections of weight 1.

3 Finite State Automata and Boolean Neural Networks

Finite State Automata. A *deterministic finite state automaton (DFSA)* is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, where $Q = \{q_0, \ldots, q_{n-1}\}$ is a finite set of *states*, Σ is a finite alphabet of *input symbols* (here, $\Sigma = \{0, 1\}$), δ: $Q \times \Sigma \longrightarrow Q$ is the *transition function*, $q_0 \in Q$ is the *initial state* and $F \subseteq Q$ is the set of *final states*. Each relation of the form $\delta(q, x) = q'$ signifies that if the automaton is in state $q \in Q$ and reads input symbol $x \in \Sigma$, then it will move to state $q' \in Q$.

For any input (string) $x = x_0 x_1 \cdots x_m \in \Sigma^*$, the *computation* of $\mathcal{A}$ over x is the finite sequence $\mathcal{A}(x) = \big((q_{i_0}, x_0, q_{i_1}), (q_{i_1}, x_1, q_{i_2}), \ldots, (q_{i_m}, x_m, q_{i_{m+1}})\big)$ such that $q_{i_0} = q_0$ and $\delta(q_{i_k}, x_k) = q_{i_{k+1}}$, for all $k = 0, \ldots, m$. Such a computation is usually denoted as

$$\mathcal{A}(x) : q_0 \xrightarrow{x_0} q_{i_1} \xrightarrow{x_1} q_{i_2} \cdots q_{i_m} \xrightarrow{x_m} q_{i_{m+1}}.$$

The input x is said to be *accepted* by $\mathcal{A}$ iff $q_{i_{m+1}} \in F$. The set of all inputs accepted by $\mathcal{A}$ is the *language recognized by* $\mathcal{A}$. Finite state automata recognize the class of *regular languages*. A finite state automaton is generally represented as a directed graph: the nodes and labelled edges of the graph represent the states and transitions of the automaton [10].

Note that if $|Q| = n$, then each state $q \in Q$ can be encoded by a corresponding Boolean vector $\boldsymbol{q} = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_p) \in \{0, 1\}^p$, where $p = \lceil \log n \rceil + 1$. The first $p - 1$ bits $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{p-1}$ encode the "value" of q and the last bit $\boldsymbol{q}_p$ encodes the "F-membership" of q, i.e., $\boldsymbol{q}_p = 1$ iff $q \in F$. Accordingly, the transition function $\delta : Q \times \{0, 1\} \longrightarrow Q$ can naturally be encoded by the *Boolean transition function* $\boldsymbol{f}_\delta : \{0, 1\} \times \{0, 1\}^p \longrightarrow \{0, 1\}^p$ defined by $\boldsymbol{f}_\delta(x, \boldsymbol{q}) = \boldsymbol{q'}$ iff $\delta(q, x) = q'$ (for the sake of consistency with the notations used in [23], we suppose that the first argument of $\boldsymbol{f}_\delta$ represents an input bit of $\mathcal{A}$, while the p remaining ones represent the encoding of a state of $\mathcal{A}$). In the sequel, the space $\{0, 1\} \times \{0, 1\}^p$ will be naturally identified with $\{0, 1\}^{p+1}$.

Simulation of DFSA by Threshold Circuits. Using the method of threshold circuit synthesis by Lupanov [17], any Boolean transition function $\boldsymbol{f}_\delta$ can be implemented by a four-layer *threshold circuit* $\mathcal{C}$ of asymptotically optimal size $\Theta(\sqrt{2^p}) = \Theta(\sqrt{n})$. The construction is fairly intricate and can be found in detail in [23] (in a slightly different context). We now describe, layer by layer, the threshold gates and connections of these circuit. The description of the weights is not provided (due to space constraint) but can be found in [23]. For the sake of consistency, we respect the notations used in [23]. This *Lupanov circuit* is illustrated in Fig. 1.

Layer 0 (Inputs). Recall that the first argument of $\boldsymbol{f}_\delta$ represents the next input bit $x \in \{0,1\}$ of $\mathcal{A}$, while the remaining p ones encode the current state $q \in Q$ of $\mathcal{A}$. The zeroth layer of $\mathcal{C}$, denoted by l_0, is composed of these $p+1$ arguments, partitioned into three groups as follows:

$$l_0 = \{u_1, \ldots, u_{p_1}\} \cup \{v_1, \ldots, v_{p_2}\} \cup \{z_1, \ldots, z_{p_3}\}$$

where

$$\begin{aligned} p_3 &= \lfloor \log(p+1-\log p) - 2 \rfloor \\ p_1 &= \left\lfloor \frac{p+1-\log p - \log(p+1-\log p)}{2} \right\rfloor \\ p_2 &= p+1-p_3-p_1 \,. \end{aligned}$$

These parameters are chosen such that, for sufficiently large p, the number of units in $\mathcal{C}$ is asymptotically optimal.

Layer 1. The first layer l_1 consists of the following set of 2^{p_2} units:

$$l_1 = \{\mu_{\boldsymbol{b}} : \boldsymbol{b} \in \{0,1\}^{p_2}\}.$$

Each input of the second group $\{v_1, \ldots, v_{p_2}\}$ is connected to all units of this layer.

Layer 2. The second layer l_2 consists of the following set of $p \cdot 2^{p_1+3}$ units:

$$l_2 = \left\{\gamma_{kj}^{\varphi a}, \lambda_{kj}^{\varphi a}, \gamma_{kj}^{\psi a}, \lambda_{kj}^{\psi a} : k \in \{1 \ldots, p\}, j \in \{0, \ldots, 2^{p_1}-1\}, a \in \{0,1\}\right\}.$$

These units are organized into p blocks of 2^{p_1+3} elements each parametrized by index $k \in \{1 \ldots, p\}$, and denoted as $l_{21}, \ldots, l_{2p}$. Each input from the first group $\{u_1, \ldots, u_{p_1}\}$ (including the next input bit to $\mathcal{A}$) and each unit of the first layer l_1 are connected to all units of this layer.

Layer 3. The third layer l_3 consists of a set of $p \cdot 2^{p_3+1}$ units:

$$l_3 = \left\{\pi_{k,\boldsymbol{c}}, \varrho_{k,\boldsymbol{c}} : k \in \{1 \ldots, p\}, \boldsymbol{c} \in \{0,1\}^{p_3}\right\}.$$

These units are also organized into p blocks of 2^{p_3+1} elements each parametrized by index $k \in \{1 \ldots, p\}$, denoted by $l_{31}, \ldots, l_{3p}$. For each $k = 1, \ldots, p$, each unit of the group l_{2k} is connected to all units of the group l_{3k}. In addition, each input of the third group $\{z_1, \ldots, z_{p_3}\}$ is connected to all units of this layer.

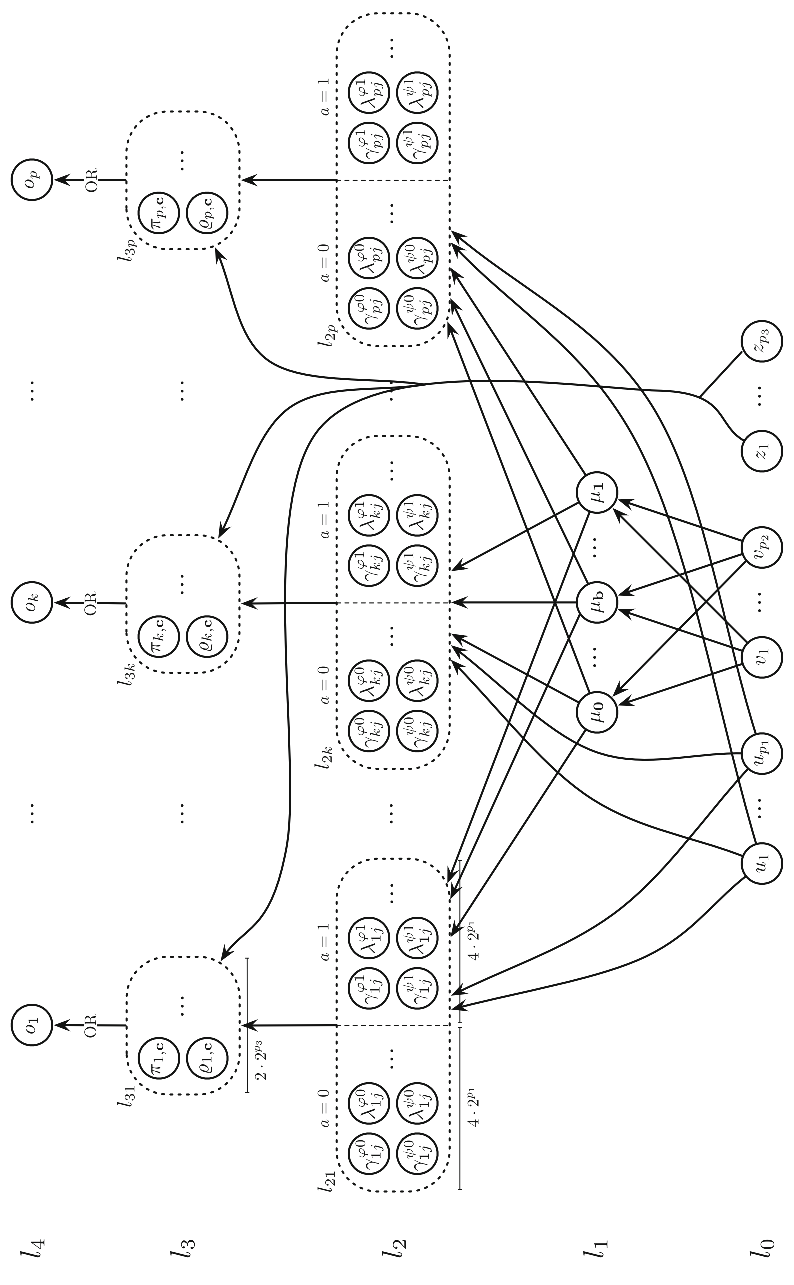
l_4
l_3
l_2
l_1
l_0
o_1
o_k
o_p
OR
l_{31}
l_{3k}
l_{3p}
$\pi_{1,\mathbf{c}}$
$\varrho_{1,\mathbf{c}}$
$\pi_{k,\mathbf{c}}$
$\varrho_{k,\mathbf{c}}$
$\pi_{p,\mathbf{c}}$
$\varrho_{p,\mathbf{c}}$
$2 \cdot 2^{p_3}$
l_{21}
l_{2k}
l_{2p}
$a = 0$
$a = 1$
$\gamma_{1j}^{\varphi 0}$ $\lambda_{1j}^{\varphi 0}$ $\gamma_{1j}^{\psi 0}$ $\lambda_{1j}^{\psi 0}$
$\gamma_{1j}^{\varphi 1}$ $\lambda_{1j}^{\varphi 1}$ $\gamma_{1j}^{\psi 1}$ $\lambda_{1j}^{\psi 1}$
$\gamma_{kj}^{\varphi 0}$ $\lambda_{kj}^{\varphi 0}$ $\gamma_{kj}^{\psi 0}$ $\lambda_{kj}^{\psi 0}$
$\gamma_{kj}^{\varphi 1}$ $\lambda_{kj}^{\varphi 1}$ $\gamma_{kj}^{\psi 1}$ $\lambda_{kj}^{\psi 1}$
$\gamma_{pj}^{\varphi 0}$ $\lambda_{pj}^{\varphi 0}$ $\gamma_{pj}^{\psi 0}$ $\lambda_{pj}^{\psi 0}$
$\gamma_{pj}^{\varphi 1}$ $\lambda_{pj}^{\varphi 1}$ $\gamma_{pj}^{\psi 1}$ $\lambda_{pj}^{\psi 1}$
$4 \cdot 2^{p_1}$
$4 \cdot 2^{p_1}$
$\mu_{\mathbf{0}}$
$\mu_{\mathbf{b}}$
$\mu_{\mathbf{1}}$
u_1
u_{p_1}
v_1
v_{p_2}
z_1
z_{p_3}

Fig. 1. Architecture of the threshold circuit $\mathcal{C}$ computing the Boolean transition function $\boldsymbol{f}_\delta$. The picture is rotated by 90°. The circuit is composed of an input layer l_0 and four layers l_1, l_2, l_3, l_4 of units (gates). An arrow connecting one unit to a block of units means that the former unit is connected to all units of the block (one-to-all connections). An arrow connecting one block of units to one unit means that all units of the block are connected to the latter unit (all-to-one connections). An arrow connecting one block of units to another means that all units of the former block are connected to all units of the latter (all-to-all connections).

Layer 4. The fourth layer l_4 is composed of the p following units:

$$l_4 = \{o_1, \ldots, o_p\}.$$

For each $k = 1, \ldots, p$, the unit o_k computes the logical disjunction (OR gate) of the outputs from the group l_{3k}. In order to implement these p OR gates, all weights associated to these units are equal to 1 whereas their biases equal -1.

Simulation of DFSA by Boolean Neural Networks. The Lupanov threshold circuit $\mathcal{C}$ computing $\boldsymbol{f}_\delta$ can easily be transformed into a recurrent neural network $\mathcal{N}$ simulating the automaton $\mathcal{A}$ [11]. This transformation is schematically illustrated and described in Fig. 2. According to this construction, the $p+1$ inputs forming the *input layer* of $\mathcal{C}$ correspond now to $p+3$ *input and state cells* in $\mathcal{N}$: one input cell `inp`, p state cells, and two additional validation and output cell `val` and `out` in order to comply with the input/output protocol. The activation values of these cells hold the consecutive encodings of the successive input symbols and computational states of $\mathcal{A}$. More specifically, if the input and state cells have activation values $(x, \boldsymbol{q}) \in \{0,1\}^{p+1}$ at time t, then the state cells will have activation values $\boldsymbol{q'} \in \{0,1\}^p$ at time $t+4$, where $\boldsymbol{q'}$ is such that $\delta(q, x) = q'$.

From these considerations, it follows that any finite state automaton with n states can be implemented by an optimal-size recurrent neural net with $\Theta(\sqrt{n})$ cells [11].

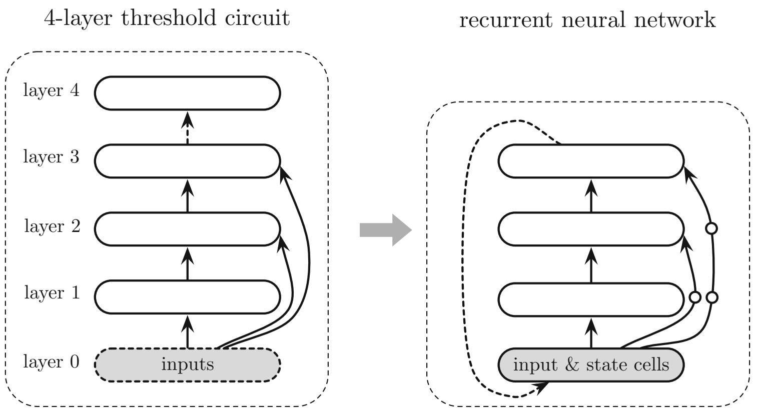

Fig. 2. Transformation of the 4-layer Lupanov threshold circuit $\mathcal{C}$ computing the Boolean function $\boldsymbol{f}_\delta$ into the recurrent neural network $\mathcal{N}$ simulating the automaton $\mathcal{A}$. The fourth layers of $\mathcal{C}$ is removed in $\mathcal{N}$; the connections from the third to the fourth layers in $\mathcal{C}$ (dashed arrow) are replaced by recurrent connections in $\mathcal{N}$ (dashed arrow); the connections from the input layer to the second and third layers in $\mathcal{C}$ are replaced by corresponding connections in $\mathcal{N}$ interspersed with delay cells (little circles), in order to ensure that the input propagation in the network is correctly timed.

4 Finite State Automata and Boolean Neural Networks Composed of Synfire Rings

Based on the results of Sect. 3, we show that any finite state automaton with n states can be implemented by a neural net composed of synfire rings containing $\Theta(\sqrt{n})$ cells. Compared to the construction of Sect. 3 and due to the addition of synfire rings, the proposed architecture has the advantage of being not only of asymptotic optimal-size, but also robust to possible failures of its constitutive cells. The general idea of this construction can be summarized as follows:

- the "state cells" of the network $\mathcal{N}$ of Sect. 3 are replaced by specific synfire rings: hence, the successive states of the automaton are now encoded by self-sustained activities of synfire rings instead of activations of "state cells";
- each *level* (not layer) of the synfire rings is connected to a copy of the network $\mathcal{N}$ of Sect. 3 (i.e., a modified copy of the Lupanov circuit).

Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a finite state automaton and $\mathcal{C}$ be the Lupanov circuit computing $\boldsymbol{f}_\delta$ (Sect. 3). We provide the description of a synfire ring-based neural network $\mathcal{N}^{SR}$ simulating $\mathcal{A}$. The network is illustrated in Fig. 3.

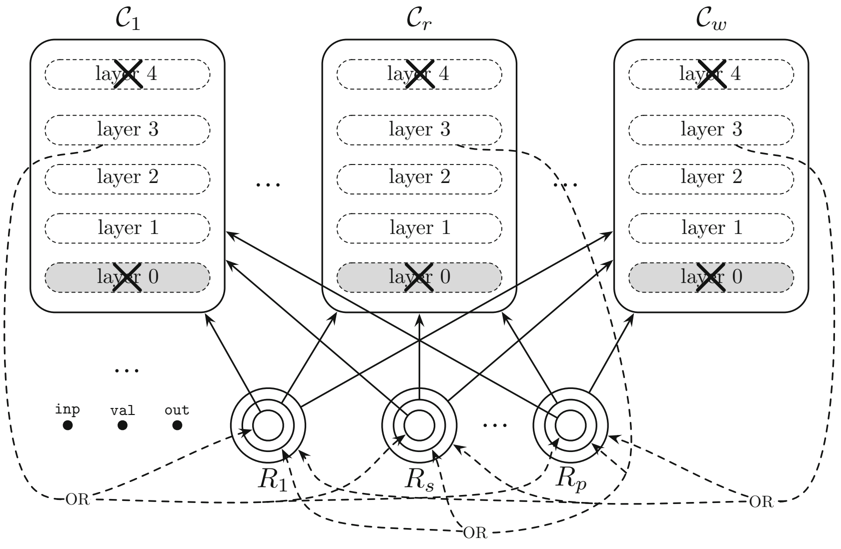

Fig. 3. Recurrent neural network composed of synfire rings $\mathcal{N}^{SR}$ simulating the automaton $\mathcal{A}$. The network is composed of an input, validation and output cell `inp`, `val` and `out`, respectively, of p synfire rings $R_1, \ldots, R_p$ of same widths w, and of w modified copies $\mathcal{C}_1, \ldots, \mathcal{C}_w$ of the Lupanov circuit $\mathcal{C}$. For each $r = 1, \ldots, w$, the r-th levels of the respective rings $R_1, \ldots, R_p$ are connected to the first, second and third layers of $\mathcal{C}_r$ (solid arrows) as described in the text. For each $r = 1, \ldots, w$ also, the third layer of $\mathcal{C}_r$ is recurrently connected to the r-th levels of all rings $R_1, \ldots, R_p$ (dashed arrows).

The network $\mathcal{N}^{SR}$ involves two input cells `inp` and `val` (as well as other cells of this kind described later), one output cell `out`, and p synfire rings $R_1, \ldots, R_p$ of respective lengths $\ell_1, \ldots, \ell_p \geq 2$ and of fixed widths $w \geq 1$. These rings will serve to encode the successive states of automaton $\mathcal{A}$. In addition, $\mathcal{N}^{SR}$ also involves of w copies of the circuit $\mathcal{C}$, denoted by $\mathcal{C}_1, \ldots, \mathcal{C}_w$. For each $r = 1, \ldots, w$, we let the p last inputs and p outputs of $\mathcal{C}_r$, i.e.,

$$u_2, \ldots, u_{p_1}, v_1, \ldots, v_{p_2}, z_1, \ldots, z_{p_3} \text{ and } o_1, \ldots, o_p$$

be denoted by $u'_{r1}, \ldots, u'_{rp}$ and $o_{r1}, \ldots, o_{rp}$, respectively.

As a first step, for each $r = 1, \ldots, w$, we remove the first layer of $\mathcal{C}_r$. Then, for every $r = 1, \ldots, w$ and $s = 1, \ldots, p$, we replace each connection from input u'_{rs} to some unit u in $\mathcal{C}_r$ by a corresponding fibre of connections from all cells of the r-th level of R_s to u. Furthermore, for each $r = 1, \ldots, w$, we remove the fourth layer (OR gates) of $\mathcal{C}_r$. Then, for every $r = 1, \ldots, w$ and $s = 1, \ldots, p$, we replace each connections from some unit u of the third layer of C_r to output o_{rs} by a recurrent connection from u to the first cell (only!) of the r-th level of R_s. In this way, each input u'_{rs} of $\mathcal{C}_r$ is represented by the r-th *level* of the ring R_s (ℓ_s cells). Recall that the dynamics of each ring R_s ensures that at most one cell is active within any of its level. Moreover, each output o_{rs} (OR gate) of C_r is implemented by recurrent connections from the third layer of $\mathcal{C}_r$ to the first cell (only) of the r-th level of the ring R_s.

The cells `inp`, `val` and `out` implement the input/output protocol of $\mathcal{N}^{SR}$. The connectivity related to these cells is described in Fig. 4. Recall that the activity of the last synfire ring R_p indicates whether the state currently encoded by the activities of the other rings $R_1, \ldots, R_{p-1}$ belongs to the set of final states F or not. This information is then transmitted from R_p to the output cell `out` via connections of weights 1 and a bias of -1 (OR gate implementation).

In order to complete the construction, further modifications need to be applied to the circuits $\mathcal{C}_r$, for $r = 1, \ldots, w$. These modifications are illustrated in Fig. 4 also. First of all, each unit of the first, second and third layer of $\mathcal{C}_r$ are provided with a sufficiently large negative bias $-W$ which prevents them from being activated. The validation cell `val` is connected to all units of the first layer of $\mathcal{C}_r$ with weights W. In this way, each time the cell `val` spikes, it cancels the negative biases of the first-layer units of $\mathcal{C}_r$, and therefore releases their activities. In addition, two new cells $\texttt{inp}_{r1}$ and $\texttt{val}_{r1}$ are added to the first layer of $\mathcal{C}_r$. These cells copy the current activities of `inp` and `val` via connections of weights $w(\texttt{inp}, \texttt{inp}_{r1}) = w(\texttt{val}, \texttt{val}_{r1}) = 1$. The cell $\texttt{inp}_{r1}$ is connected to all units of the second layer of $\mathcal{C}_r$ with the weights given in [23], and the cell $\texttt{val}_{r1}$ is also connected to all units of the second layer of $\mathcal{C}_r$ with weights W. Furthermore, a new cell $\texttt{val}_{r2}$ is added to the second layer of $\mathcal{C}_r$, which just copies the current activity $\texttt{val}_{r1}$ by a connection of weight $w(\texttt{val}_{r1}, \texttt{val}_{r2}) = 1$. The cell $\texttt{val}_{r2}$ is connected to all units in the third layer of $\mathcal{C}_r$ via connections of weights W. Finally, connections with large negative weights $-W'$ connect $\texttt{val}_{r2}$ to all cells of the r-th levels of $R_1, \ldots, R_p$. The weight $-W'$ is chosen such that it suffices to inhibit an activity propagating in a synfire ring.

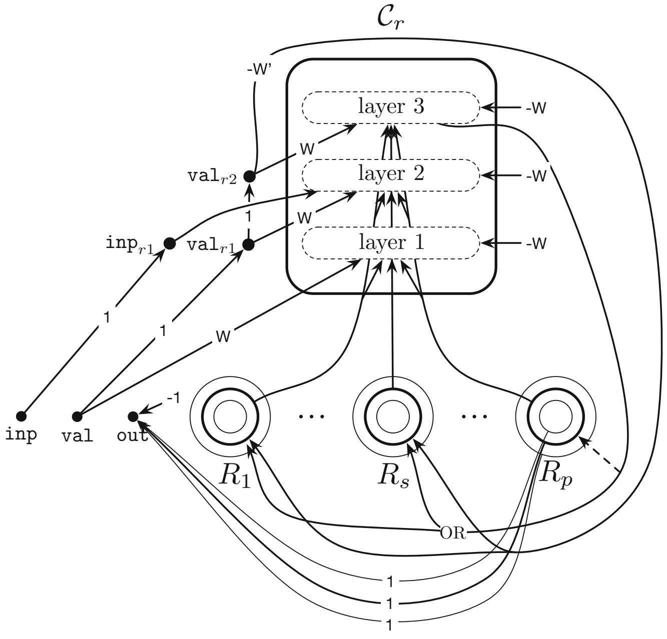

Fig. 4. Illustration of the r-th modified Lupanov circuits involved in the construction of the synfire ring-based network $\mathcal{N}^{SR}$. Biases of weights $-W$ are added to every cells of the first, second and third layers. Three cells $\texttt{inp}_{r1}$, $\texttt{val}_{r1}$, $\texttt{val}_{r2}$ are also added. The connection between those, and from those to the circuit's layers are described in the figure. In addition, recurrent connections of weights $-W'$ from $\texttt{val}_{r2}$ to the r-th levels of all rings $R_1, \ldots, R_p$ (in bold) serve to inhibit the rings (reinitialization), before the latter are reactivated by the recurrent connections from the third layer to their r-th levels (in bold). Finally, the cells `inp`, `val` and `out` implement the input/output protocol as described in the text.

Correctness of the Construction. We now sketch the proof that the synfire ring-based network $\mathcal{N}^{SR}$ simulates the finite state automaton $\mathcal{A}$ correctly. Suppose that the activities of the p synfire rings $R_1, \ldots, R_p$ are currently encoding the state $q \in Q$ of automaton $\mathcal{A}$. Suppose further that the input bit $x \in \{0, 1\}$ is received at time t. According to the input protocol, this means that $\texttt{inp}(t) = x$ and $\texttt{val}(t) = 1$. The network's architecture ensures that, for any $r = 1, \ldots, w$, the combined activities of the r-th levels of $R_1, \ldots, R_p$ together with the activation of `val` at time t will activate the first layer of $\mathcal{C}_r$ at time $t + 1$. The connections of $\mathcal{C}_r$ also ensure that $\texttt{inp}(t) = \texttt{inp}_{r1}(t+1)$ and $\texttt{val}(t) = \texttt{val}_{r1}(t+1) = 1$. Hence, the combined activities of the first layer of $\mathcal{C}_r$ together with the activation of $\texttt{val}_{r1}$ at time $t+1$ will activate the second layer of $\mathcal{C}_r$ at time $t+2$. The connections of $\mathcal{C}_r$ also ensure that $\texttt{val}_{r1}(t+1) = \texttt{val}_{r2}(t+2) = 1$. Consequently, the combined activities of the second layer of $\mathcal{C}_r$ together with the activation of $\texttt{val}_{r2}$ at time $t + 2$ will activate the third layer of $\mathcal{C}_r$ at time $t + 3$. But at time

$t+2$ also, $\texttt{val}_{r2}$ sends strong inhibitions to the r-th levels of all rings $R_1, \ldots, R_p$. Since this happens for all $r \in \{1, \ldots, w\}$ simultaneously, the rings $R_1, \ldots, R_p$ are shut down at time $t+3$. Finally, the (special) recurrent activities from the third layers of all circuits $\mathcal{C}_1, \ldots, \mathcal{C}_w$ at time $t+3$ ensure that the activities of all rings $R_1, \ldots, R_p$ are correctly updated at time $t+4$. From the Lupanov's construction, it follows that, from time $t+4$ onwards, the activities of the rings $R_1, \ldots, R_p$ encode the state q' of $\mathcal{A}$ such that $\delta(q, x) = q'$. Finally, the unit $\texttt{out}$ fires at the next time step $t+5$ iff q' is a final state. In this sense, each transition of $\mathcal{A}$ is correctly simulated by the network $\mathcal{N}^{SR}$.

If the automaton $\mathcal{A}$ contains n states, it has been shown that each Lupanov circuit $\mathcal{C}_r$ $(r = 1, \ldots, w)$ involved in the construction has an optimal-size of $\Theta(\sqrt{n})$. Therefore, the network $\mathcal{N}^{SR}$ has a size of $\Theta(w \cdot \sqrt{n}) = \Theta(\sqrt{n})$, which is also optimal.

5 Conclusion

We described a robust and optimal-size implementation of finite state automata by means of neural networks composed of synfire rings. The robustness of the network results from its architecture, which is composed of synfire rings and duplicated core components. For an automaton with n states, the corresponding network has an optimal size of $\Theta(\sqrt{n})$ cells.

This study makes one step forward in the implementation of finite state machines by biologically inspired neural networks. In an extended journal version of this paper, the construction is expected to be implemented, and examples of such synfire ring-based neural networks will be computationally simulated. Furthermore, the construction is expected to be generalized in such a way that only synfire rings are involved.

References

1. Abeles, M.: Corticonics: Neuronal Circuits of the Cerebral Cortex. Cambridge University Press, Cambridge (1991)
2. Abeles, M.: Time is precious. Science **304**(5670), 523–524 (2004). https://doi.org/10.1126/science.1097725
3. Cabessa, J., Horcholle-Bossavit, G., Quenet, B.: Neural computation with spiking neural networks composed of synfire rings. In: Lintas, A., Rovetta, S., Verschure, P.F.M.J., Villa, A.E.P. (eds.) ICANN 2017. LNCS, vol. 10613, pp. 245–253. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68600-4_29
4. Cabessa, J., Masulli, P.: Emulation of finite state automata with networks of synfire rings. In: 2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14–19, 2017, pp. 4641–4648. IEEE (2017). https://doi.org/10.1109/IJCNN.2017.7966445
5. Cabessa, J., Siegelmann, H.T.: The super-turing computational power of plastic recurrent neural networks. Int. J. Neural Syst. **24**(8), 1450029 (2014). https://doi.org/10.1142/S0129065714500294

6. Cabessa, J., Tchaptchet, A.: Automata computation with Hodgkin-Huxley based neural networks composed of synfire rings. In: 2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, July 8–13, 2018, pp. 1–8. IEEE (2018). https://doi.org/10.1109/IJCNN.2018.8489700
7. Diesmann, M., Gewaltig, M.O., Aertsen, A.: Stable propagation of synchronous spiking in cortical neural networks. Nature **402**, 529–533 (1999). https://doi.org/10.1038/990101
8. Hertz, J., Prügel-Bennett, A.: Learning synfire chains by self-organization. Netw.: Comput. Neural Syst. **7**(2), 357–363 (1996). https://doi.org/10.1088/0954-898X_7_2_017
9. Hertz, J., Prügel-Bennett, A.: Learning synfire chains: turning noise into signal. Int. J. Neural Syst. **7**(4), 445–450 (1996). https://doi.org/10.1142/S0129065796000427
10. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation, 3rd edn. Pearson international edition, Addison-Wesley, Boston (2007)
11. Horne, B.G., Hush, D.R.: Bounds on the complexity of recurrent neural network implementations of finite state machines. Neural Netw. **9**(2), 243–252 (1996). https://doi.org/10.1016/0893-6080(95)00095-X
12. Ikegaya, Y., et al.: Synfire chains and cortical songs: temporal modules of cortical activity. Science **304**(5670), 559–564 (2004). https://doi.org/10.1126/science.1093173
13. Indyk, P.: Optimal simulation of automata by neural nets. In: Mayr, E.W., Puech, C. (eds.) STACS 1995. LNCS, vol. 900, pp. 337–348. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-59042-0_85
14. Jun, J.K., Jin, D.Z.: Development of neural circuitry for precise temporal sequences through spontaneous activity, axon remodeling, and synaptic plasticity. PLOS One **2**(8), 1–17 (2007). https://doi.org/10.1371/journal.pone.0000723
15. Kleene, S.C.: Representation of events in nerve nets and finite automata. In: Shannon, C., McCarthy, J. (eds.) Automata Studies, vol. 34, pp. 3–42. Princeton University Press, Princeton (1956). https://doi.org/10.1515/9781400882618-002
16. Levy, N., Horn, D., Meilijson, I., Ruppin, E.: Distributed synchrony in a cell assembly of spiking neurons. Neural Netw. **14**(6–7), 815–824 (2001). https://doi.org/10.1016/S0893-6080(01)00044-2
17. Lupanov, O.B.: On the synthesis of threshold circuits. Probl. Kibernet. **26**, 109–140 (1973)
18. Mainen, Z., Sejnowski, T.: Reliability of spike timing in neocortical neurons. Science **268**(5216), 1503–1506 (1995). https://doi.org/10.1126/science.7770778
19. Minsky, M.L.: Computation: Finite and Infinite Machines. Prentice-Hall Inc., Englewood Cliffs (1967)
20. Siegelmann, H.T.: Neural Networks and Analog Computation: Beyond the Turing Limit. Birkhauser Boston Inc., Cambridge (1999)
21. Siegelmann, H.T., Sontag, E.D.: Analog computation via neural networks. Theor. Comput. Sci. **131**(2), 331–360 (1994). https://doi.org/10.1016/0304-3975(94)90178-3
22. Siegelmann, H.T., Sontag, E.D.: On the computational power of neural nets. J. Comput. Syst. Sci. **50**(1), 132–150 (1995). https://doi.org/10.1006/jcss.1995.1013
23. Šíma, J.: Energy complexity of recurrent neural networks. Neural Comput. **26**(5), 953–973 (2014). https://doi.org/10.1162/NECO_a_00579
24. Šíma, J., Orponen, P.: General-purpose computation with neural networks: a survey of complexity theoretic results. Neural Comput. **15**(12), 2727–2778 (2003). https://doi.org/10.1162/089976603322518731

25. Šíma, J., Wiedermann, J.: Theory of neuromata. J. ACM **45**(1), 155–178 (1998). https://doi.org/10.1145/273865.273914
26. Zheng, P., Triesch, J.: Robust development of synfire chains from multiple plasticity mechanisms. Front. Comput. Neurosci. **8**(66) (2014). https://doi.org/10.3389/fncom.2014.00066