Analog Neuron Hierarchy

Jiří Šíma[a,*]

[a]*Institute of Computer Science of the Czech Academy of Sciences, P. O. Box 5, 18207 Prague 8, Czech Republic,*

Abstract

In order to refine the analysis of the computational power of discrete-time recurrent neural networks (NNs) between the binary-state NNs which are equivalent to finite automata (level 3 in the Chomsky hierarchy), and the analog-state NNs with rational weights which are Turing-complete (Chomsky level 0), we study an intermediate model αANN of a binary-state NN that is extended with $\alpha \geq 0$ extra analog-state neurons. For rational weights, we establish an analog neuron hierarchy 0ANNs $\subset$ 1ANNs $\subset$ 2ANNs $\subseteq$ 3ANNs and separate its first two levels. In particular, 0ANNs coincide with the binary-state NNs (Chomsky level 3) being a proper subset of 1ANNs which accept at most context-sensitive languages (Chomsky level 1) including some non-context-free ones (above Chomsky level 2). We prove that the deterministic (context-free) language $L_{\#} = \{0^n 1^n \,|\, n \geq 1\}$ cannot be recognized by any 1ANN even with real weights. In contrast, we show that deterministic pushdown automata accepting deterministic languages can be simulated by 2ANNs with rational weights, which thus constitute a proper superset of 1ANNs. Finally, we prove that the analog neuron hierarchy collapses to 3ANNs by showing that any Turing machine can be simulated by a 3ANN having rational weights, with linear-time overhead.

Keywords: recurrent neural network, analog neuron hierarchy, deterministic context-free language, Turing machine, Chomsky hierarchy

*Corresponding author

Email address: sima@cs.cas.cz (Jiří Šíma)

1. Introduction

The majority of standard techniques used in artificial neural networks (NNs) such as Hebbian learning, back-propagation, simulated annealing, support vector machines, deep learning, are of statistical or heuristic nature. NNs often considered as "black box" solutions are mainly subject to empirical research whose methodology is based on computer simulations through which the developed heuristics are tested, tuned, and mutually compared on benchmark data. The efficiency and significance of proposed heuristics are eventually approved by successful practical applications. Nevertheless, the development of NN methods has, among others, its own intrinsic limits given by mathematical, computability, or physical laws. By exploring these limits one can understand what is computable in principle or efficiently by NNs. This is a necessary prerequisite for pushing or even overcoming the respective boundaries in future intelligent technologies.

In order to answer these issues, rigorous mathematical foundations of NNs need to be further developed, which is the main motivation for this study. We will thus not provide particular algorithmic solutions to practical special-purpose machine learning problems, but instead we will explore the computational potential and limits of NNs for general-purpose computation. In order to achieve this objective, the computational power of NNs is investigated by comparing them with more traditional models of computation such as finite or pushdown automata, Chomsky grammars, and Turing machines.

The computational power of discrete-time recurrent NNs with the saturated-linear activation function[1] depends on the descriptive complexity of their weight parameters (Siegelmann, 1999; Šíma and Orponen, 2003). NNs with *integer* weights, corresponding to binary-state (shortly binary) networks employing the Heaviside activation function (with Boolean outputs 0 or 1), coincide with finite automata (FAs) recognizing regular languages (Alon et al., 1991; Horne and Hush, 1996; Indyk, 1995; Minsky, 1967; Šíma, 2014; Šíma and Wiedermann, 1998). *Rational* weights make the analog-state (shortly analog) NNs (with real-valued outputs in the interval $[0, 1]$) computationally equivalent to Turing machines (TMs) (Indyk, 1995; Siegelmann and Sontag, 1995), and thus (by a real-time simulation due to Siegelmann and

[1]The results are partially valid for more general classes of activation functions (Koiran, 1996; Siegelmann, 1996; Šíma, 1997; Šorel and Šíma, 2004) including the logistic function (Kilian and Siegelmann, 1996).

Sontag, 1995) polynomial-time computations of such networks are characterized by the fundamental complexity class P.

In addition, NNs with arbitrary *real* weights can even derive "super-Turing" computational capabilities (Siegelmann, 1999). Namely, their polynomial-time computations correspond to the nonuniform complexity class P/poly while any input/output mapping (including algorithmically undecidable problems) can be computed within exponential time (Siegelmann and Sontag, 1994). Moreover, a proper infinite hierarchy of nonuniform complexity classes between P and P/poly has been established for polynomial-time computations of NNs with increasing Kolmogorov complexity of real weights (Balcázar et al., 1997).

As can be seen, our understanding of the computational power of NNs is satisfactorily fine-grained when changing from rational to arbitrary real weights. In contrast, there is still a gap between integer and rational weights which results in a jump from regular languages capturing the lowest level 3 in the Chomsky hierarchy to recursively enumerable languages on the highest Chomsky level 0. In order to refine the classification of NNs which do not possess the full power of TMs (Chomsky level 0), we have initiated the study of binary-state NNs employing integer weights, that are extended with $\alpha \geq 0$ extra analog neurons having real weights, which are denoted as αANNs.

This study has primarily been motivated by theoretical issues of how the computational power of NNs increases with enlarging analogicity when we change step by step from binary to analog states, or equivalently, from integer to arbitrary rational weights. Thus, the proposed model of αANNs itself has been intended for measuring the expressive power of a binary-state NN to which analog neurons are added one by one, rather than for solving special-purpose practical tasks. Nevertheless, as a secondary use, this analysis may potentially be relevant to practical hybrid NNs that combine binary and analog neurons in deep networks employing the LSTM, GRU or ReLU units (Schmidhuber, 2015), which deserves specialized studies such as recent work by Korsky and Berwick (2019); Merrill (2019); Merrill et al. (2020); Weiss et al. (2018).

In our previous work (Šíma, 2019b), we have characterized syntactically the class of languages that are accepted by 1ANNs with *one* extra analog unit, in terms of so-called *cut languages*[2] (Šíma and Savický, 2018) which

[2]A cut language $L_{<c} = \{x_1 \dots x_n \in A^* \mid \sum_{k=1}^{n} x_k \beta^{-k} < c\}$ contains finite representa-

are combined in a certain way by usual operations[3] under which the classes of regular and context-sensitive languages are closed. By using this syntactic characterization of 1ANNs we have derived a sufficient condition when a 1ANN recognizes only a regular language (Chomsky level 3), which is based on the *quasi-periodicity*[4] (Šíma and Savický, 2018) of some parameters depending on its real weights. In particular, a 1ANN with weights from the smallest field extension[5] $\mathbb{Q}(\beta)$ over the rational numbers $\mathbb{Q}$ including a Pisot number[6] $\beta > 1$, such that the self-loop weight w of its only analog neuron equals $1/\beta$, is computationally equivalent to a FA. For instance, since every integer $n > 1$ is a Pisot number (in fact, such integers are the only rational Pisot numbers), it follows that any 1ANN with rational weights such that $w = 1/n$, accepts a regular language. An example of a 1ANN that accepts the regular language (23), is depicted in Figure 2 with parameters (21). More complex examples of such neural FAs, are 1ANNs that have rational weights except for the irrational (algebraic) self-loop weight $w = 1/\rho \approx 0.754878$ or $w = 1/\varphi = \varphi - 1 \approx 0.618034$ for the plastic constant[7] ρ or the golden ratio φ, respectively, which are Pisot numbers.

On the other hand, we have introduced (Šíma, 2019b) examples of lan-

---

tions of numbers in a real *base* β (so-called *β-expansions*) where $|\beta| > 1$, using real *digits* from a finite alphabet A, that are less than a given real *threshold* c (i.e. a Dedekind cut). It is known that $L_{<c}$ is regular iff c is quasi-periodic[4] while it is not context-free otherwise.

[3]complementation, intersection, union, concatenation, Kleene star, reversal, the largest prefix-closed subset, and a letter-to-letter morphism

[4]For a real base β satisfying $|\beta| > 1$, and a finite alphabet A of real digits, an infinite β-expansion[2], $\sum_{k=1}^{\infty} x_k \beta^{-k}$ where $x_k \in A$, is called *quasi-periodic* if the sequence $\left(\sum_{k=1}^{\infty} x_{n+k} \beta^{-k}\right)_{n=0}^{\infty}$ contains a constant infinite subsequence. We say that a real number x is *quasi-periodic* if all its infinite β-expansions $x = \sum_{k=1}^{n} x_k \beta^{-k}$ are quasi-periodic.

[5]Recall that in algebra, the rational numbers (fractions) form the field $\mathbb{Q}$ with the two usual operations, the addition and the multiplication over real numbers. For any real number $\beta \in \mathbb{R}$, the *field extension* $\mathbb{Q}(\beta) \subset \mathbb{R}$ is the smallest set containing $\mathbb{Q} \cup \{\beta\}$ that is closed under these operations. For example, the golden ratio $\varphi = (1 + \sqrt{5})/2 \in \mathbb{Q}(\sqrt{5})$ whereas $\sqrt{2} \notin \mathbb{Q}(\sqrt{5})$. Note that $\mathbb{Q}(\beta) = \mathbb{Q}$ for every $\beta \in \mathbb{Q}$.

[6]A *Pisot number* is a real algebraic integer (a root of some monic polynomial with integer coefficients) greater than 1 such that all its Galois conjugates (other roots of such a unique monic polynomial with minimal degree) are in absolute value less than 1. A characteristic property of Pisot numbers is that their powers approach integers at an exponential rate.

[7]The *plastic constant* $\rho \approx 1.324718$ is the unique real root of the cubic equation $x^3 = x + 1$, which is the smallest Pisot number.

guages accepted by 1ANNs with rational weights that are not context-free (CFLs) (i.e. are above Chomsky level 2), while we have proven that any language accepted by this model online[8], is context-sensitive (CSL) at Chomsky level 1. For example, the 1ANN depicted in Figure 2 with parameters (8), accepts the context-sensitive language $L^R_{<1}$ defined in (20), which is not context-free. These results refine the analysis of the computational power of NNs with the weight parameters between integer and rational weights. Namely, the computational power of binary-state networks having integer weights can increase from regular languages (Chomsky level 3) to that between CFLs (Chomsky level 2) and CSLs (Chomsky level 1), when an extra analog unit with rational weights is added, while a condition when adding one analog neuron does not increase the power of binary-state networks including even real weights, was formulated.

In this paper, we establish an analog neuron hierarchy of classes of languages recognized by αANNs with α extra analog units having rational weights, for $\alpha = 0, 1, 2, 3, \ldots$, that is, 0ANNs $\subseteq$ 1ANNs $\subseteq$ 2ANNs $\subseteq$ 3ANNs $\subseteq \cdots$, respectively. Note that we use the notation αANNs also for the class of languages accepted by αANNs, which can clearly be distinguished by the context. Obviously, the 0ANNs are purely binary-state NNs which are equivalent to FAs and hence, 0ANNs $\subsetneq$ 1ANNs because we know that the non-context-free language $L^R_{<1}$ is accepted by the 1ANN depicted in Figure 2 with parameters (8). We will prove that the deterministic context-free language (DCFL) $L_{\#} = \{0^n 1^n \,|\, n \geq 1\}$, which contains the words of n zeros followed by n ones, cannot be recognized even offline[8] by any 1ANN with arbitrary real weights. The proof is based on an asymptotic analysis of computations by 1ANNs whose dynamics is quite restricted for recalling a stored number of zeros. Since typical CFLs inherently include the language $L_{\#}$, we conjecture that 1ANNs cannot recognize any non-regular CFL (Chomsky level 2 strictly above level 3), which has already been shown in the deterministic case (Šíma and Plátek, 2019). In contrast, recall there exist non-context-free languages (above Chomsky level 2) that are accepted by 1ANNs such as $L^R_{<1}$. Anyway, we thus know that 1ANNs with real weights are not Turing-complete.

Furthermore, we will show that any deterministic pushdown automaton

[8]In *online* input/output protocols, the time between reading two consecutive input symbols as well as the delay in outputting the result after an input has been read, is bounded by a constant, while in *offline* protocols these time intervals are not bounded.

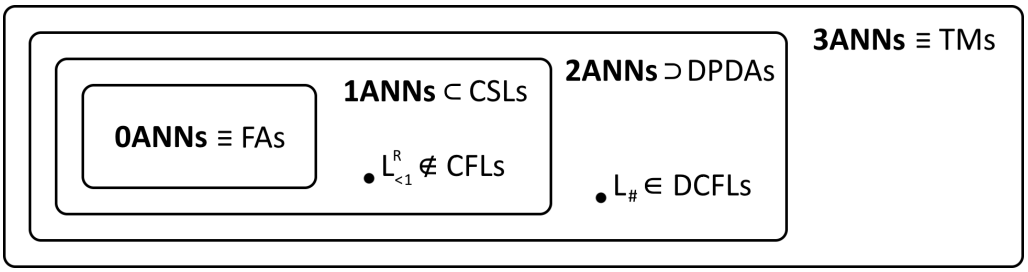

Figure 1: The analog neuron hierarchy.

(DPDA) can be simulated by a 2ANN with two extra analog neurons having rational weights. This means that the DCFLs (included in Chomsky level 2) are recognized by 2ANNs with rational weights. Thus, 1ANNs $\subsetneqq$ 2ANNs since the DCFL $L_\#$ is not accepted by any 1ANN. In addition, we will prove that any TM can be simulated by a 3ANN having rational weights with a linear-time overhead. It follows that recursively enumerable languages (Chomsky level 0) are accepted by 3ANNs with rational weights and thus this model including only three analog neurons is Turing-complete. Since αANNs with rational weights can be simulated by TMs for any $\alpha \geq 0$, the analog neuron hierarchy collapses to 3ANNs:

$$\begin{aligned} \text{FAs} \equiv \text{0ANNs} \subsetneqq \text{1ANNs} \subsetneqq \text{2ANNs} \subseteq \text{3ANNs} &= \text{4ANNs} = \ldots \\ &\equiv \text{TMs}\,, \end{aligned} \tag{1}$$

which is schematically depicted in Figure 1. The separation 2ANNs $\subsetneqq$ 3ANNs of the third level remains open as the most important challenge for further research. It appears that the analog neuron hierarchy (1) is only partially comparable to that of Chomsky.

The underlying simulations of DPDAs and TMs by 2ANNs and 3ANNs, respectively, are based on the classical technique of implementing the PDA's stack by two analog neurons, one for the `pop` operation and the other one for `push`, where the stack contents are encoded by analog states using a Cantor-like set (Siegelmann and Sontag, 1995). Moreover, two stacks are known to be sufficient for simulating TMs. The technical part of the proof then consists in synchronizing the `swap` operation on the states of analog neurons.

The paper is organized as follows. In Section 2, we introduce basic definitions concerning the language acceptors based on αANNs with an offline[8]

input/output protocol using the binary input alphabet $\{0,1\}$. In Section 3, we prove that the deterministic language $L_{\#}$ cannot be recognized by any 1ANN with real weights. Section 4 shows that any DPDA can be simulated by 2ANNs which is illustrated by an example of a 2ANN recognizing $L_{\#}$, while the simulation of any TM by a 3ANN is presented in Section 5. Finally, we summarize and discuss the results, and list some open problems in Section 6. Preliminary versions of the results in this paper appeared in extended abstracts (Šíma, 2019a, 2018) containing only proof sketches.

2. Neural Language Acceptors with α Extra Analog Units

For an integer constant $\alpha \geq 0$, we specify a computational model of a *discrete-time binary-state recurrent neural network αANN with α extra analog units*, $\mathcal{N}$, which will be used as a formal language acceptor. The network $\mathcal{N}$ consists of $s \geq \alpha$ *units (neurons)*, indexed as $V = \{1, \ldots, s\}$. The units in $\mathcal{N}$ are assumed to be binary-state (shortly *binary*) neurons (i.e. *perceptrons, threshold gates)* except for the first α neurons $1, \ldots, \alpha \in V$ which are *analog* units. The neurons are connected into a directed graph representing an *architecture* of $\mathcal{N}$, in which each edge $(i,j) \in V^2$ leading from unit i to j is labeled with a real *weight* $w_{ji} \in \mathbb{R}$. The absence of a connection within the architecture corresponds to a zero weight between the respective neurons, and vice versa.

The *computational dynamics* of $\mathcal{N}$ determines for each unit $j \in V$ its *state (output)* $y_j^{(t)}$ at discrete time instants $t = 0, 1, 2, \ldots$. The outputs $y_1^{(t)}, \ldots, y_\alpha^{(t)}$ from analog units $1, \ldots, \alpha \in V$ are real numbers from the unit interval $\mathbb{I} = [0,1]$, whereas the states $y_j^{(t)}$ of the remaining $s - \alpha$ neurons $j \in V' = V \setminus \{1, \ldots, \alpha\} = \{\alpha + 1, \ldots, s\}$ are binary values from $\{0,1\}$. This establishes the *network state*

$$\mathbf{y}^{(t)} = \left(y_1^{(t)}, \ldots, y_\alpha^{(t)}, y_{\alpha+1}^{(t)}, \ldots, y_s^{(t)}\right) \in \mathbb{I}^\alpha \times \{0,1\}^{s-\alpha}$$

at each discrete time instant $t \geq 0$.

For notational simplicity, we assume a synchronous fully parallel mode without loss of efficiency (Orponen, 1997). At the beginning of a computation, the αANN $\mathcal{N}$ is placed in a predefined *initial state* $\mathbf{y}^{(0)} \in \{0,1\}^s$. At discrete time instant $t \geq 0$, an *excitation* of any neuron $j \in V$ is evaluated as

$$\xi_j^{(t)} = \sum_{i=0}^{s} w_{ji} y_i^{(t)}, \tag{2}$$

including a real *bias* value $w_{j0} \in \mathbb{R}$ which, as usually, can be viewed as the weight from a formal constant unit input $y_0^{(t)} \equiv 1$ for every $t \geq 0$ (i.e. the set of neurons is formally extended with $0 \in V' \subseteq V$). At the next instant $t+1$, all the neurons $j \in V$ compute their new outputs $y_j^{(t+1)}$ in parallel by applying an *activation function* $\sigma_j : \mathbb{R} \longrightarrow \mathbb{I}$ to $\xi_j^{(t)}$, that is,

$$y_j^{(t+1)} = \sigma_j\left(\xi_j^{(t)}\right) \quad \text{for } j \in V\,. \tag{3}$$

The analog units $j \in \{1, \ldots, \alpha\}$ employ the *saturated-linear* function $\sigma_j(\xi) = \sigma(\xi)$ where

$$\sigma(\xi) = \begin{cases} 1 & \text{for } \xi > 1 \\ \xi & \text{for } 0 \leq \xi \leq 1 \\ 0 & \text{for } \xi < 0\,, \end{cases} \tag{4}$$

while for neurons $j \in V'$ with binary states $y_j \in \{0,1\}$, the *Heaviside* activation function $\sigma_j(\xi) = H(\xi)$ is used where

$$H(\xi) = \begin{cases} 1 & \text{for } \xi \geq 0 \\ 0 & \text{for } \xi < 0\,. \end{cases} \tag{5}$$

This determines the new network state $\mathbf{y}^{(t+1)} \in \mathbb{I}^\alpha \times \{0,1\}^{s-\alpha}$ at time $t+1$.

The computational power of NNs has been studied analogously to the traditional models of computations so that the networks are exploited as acceptors of formal languages $L \subseteq \Sigma^*$ over a finite alphabet $\Sigma \neq \emptyset$ (Šíma and Orponen, 2003). For simplicity, we further assume the binary alphabet[9], $\Sigma = \{0,1\}$. For an αANN $\mathcal{N}$, we use the following offline[8] input/output protocol employing its three special binary neurons $\mathrm{inp}, \mathrm{out}, \mathrm{nxt} \in V'$. An input word (string) $\mathbf{x} = x_1 \ldots x_n \in \{0,1\}^n$ of arbitrary length $n \geq 0$, is sequentially presented to the finite $\mathcal{N}$, bit after bit, via the so-called *input neuron* $\mathrm{inp} \in V'$, at time instants $0 < \tau_1 < \tau_2 < \cdots < \tau_n$ after queried by $\mathcal{N}$. The neuron $\mathrm{nxt} \in V'$ is used by $\mathcal{N}$ to prompt a user to enter the next input bit. Thus, once the prefix $x_1, \ldots, x_{k-1}$ of $\mathbf{x}$ for $1 \leq k \leq n$, has been read, the state of inp is externally set to the next input bit $x_k \in \{0,1\}$ at the

[9]For an arbitrary alphabet Σ, one can employ one-hot encoding using $|\Sigma|$ input neurons so that only one neuron corresponding to a current input symbol that is presented to $\mathcal{N}$, is activated (Šíma, 2019b).

time instant τ_k that is one computational step after $\mathcal{N}$ activates the neuron $\mathrm{nxt} \in V'$, which means the underlying states satisfy

$$y_{\mathrm{inp}}^{(t)} = \begin{cases} x_k & \text{if } t = \tau_k \\ 0 & \text{otherwise} \end{cases} \qquad y_{\mathrm{nxt}}^{(t-1)} = \begin{cases} 1 & \text{if } t = \tau_k \\ 0 & \text{otherwise} \end{cases} \quad \text{for } k = 1, \ldots, n\,. \tag{6}$$

At the same time, $\mathcal{N}$ carries its computation, possibly deciding about each prefix of the input word $\mathbf{x}$ whether it belongs to L, which is indicated by the output neuron $\mathrm{out} \in V'$ when the next input bit is presented (one step after the neuron nxt is active):

$$y_{\mathrm{out}}^{(\tau_{k+1})} = \begin{cases} 1 & \text{if } x_1 \ldots x_k \in L \\ 0 & \text{if } x_1 \ldots x_k \notin L \end{cases} \quad \text{for } k = 0, \ldots, n\,, \tag{7}$$

where $\tau_{n+1} > \tau_n$ is the time instant when the input word $\mathbf{x}$ is decided (e.g. formally define $x_{n+1} = 0$ to ensure the consistency with the input protocol (6) for $k = n + 1$). For instance, $y_{\mathrm{out}}^{(\tau_1)} = 1$ iff the empty word ε belongs to L. Note that $\mathcal{N}$ may not halt when it does not prompt for the next input bit (i.e. τ_{k+1} is not defined) or $\mathcal{N}$ may not provide the output (i.e. τ_{n+1} is not defined). We say that a language $L \subseteq \{0,1\}^*$ is *accepted (recognized)* by αANN $\mathcal{N}$, which is denoted as $L = \mathcal{L}(\mathcal{N})$, if for any input word $\mathbf{x} \in \{0,1\}^*$, $\mathbf{x} \in L$ iff $\mathcal{N}$ halts and accepts $\mathbf{x}$.

Example 1 We illustrate the definition of the αANN language acceptor and its input/output protocol on a simple 1ANN $\mathcal{N}$ with $\alpha = 1$. This 1ANN is used for recognizing a non-context-free language while for other parameters its power reduces to regular languages. The network $\mathcal{N}$ is composed of $s = 7$ neurons, that is, $V = \{0, 1, 2, 3, 4, \mathrm{inp}, \mathrm{out}, \mathrm{nxt}\}$ where the first neuron $1 \in V$ is the analog unit whereas $V' = V \setminus \{1\} = \{0, 2, 3, 4, \mathrm{inp}, \mathrm{out}, \mathrm{nxt}\}$ contains the remaining binary-state neurons including the formal unit $0 \in V'$ for biases. The architecture of $\mathcal{N}$ is depicted in Figure 2 where the directed edges connecting neurons are labeled with the respective weights $w_{1,\mathrm{inp}} = \beta^{-1}/\nu = (\beta - 1)/\beta$, $w_{11} = \beta^{-1/3}$, $w_{2,\mathrm{nxt}} = w_{32} = w_{\mathrm{nxt},3} = w_{41} = w_{43} = w_{\mathrm{out},\mathrm{nxt}} = 1$, and $w_{\mathrm{out},4} = -1$, while the edges drawn without the originating unit $0 \in V'$ correspond to the biases $w_{40} = -1 - c/\nu = -1 - (\beta - 1)c$ and $w_{\mathrm{nxt},0} = w_{20} = w_{30} = w_{\mathrm{out},0} = -1$, where the parameter c is a real threshold and $\beta > 1$ is a real base which defines $\nu = \sum_{k=1}^{\infty} \beta^{-k} = 1/(\beta - 1) > 0$.

The 1ANN $\mathcal{N}$ is employed for recognizing a language $L = \mathcal{L}(\mathcal{N})$ over the binary alphabet $\Sigma = \{0, 1\}$. For this purpose, the special units $\mathrm{inp}, \mathrm{out}, \mathrm{nxt} \in$

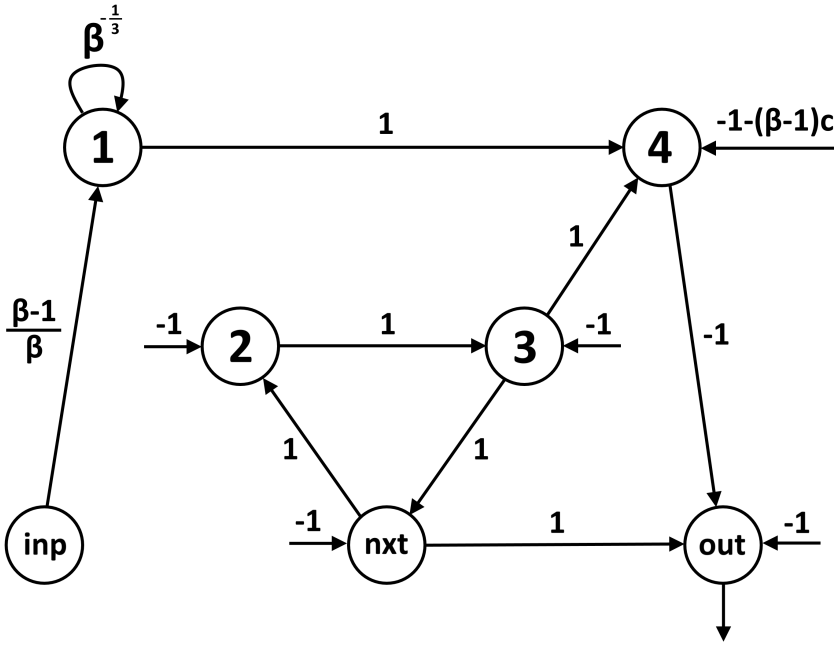

Figure 2: Example of a 1ANN language acceptor with parameters β and c.

V' implement the input/output protocol (6) and (7). For example, let

$$\beta = \left(\frac{6}{5}\right)^3 = \frac{216}{125} \qquad \text{and} \qquad c = 1 \tag{8}$$

which determine the parameterized weights and bias of $\mathcal{N}$,

$$w_{1,\text{inp}} = \frac{91}{216}\,, \qquad w_{11} = \frac{5}{6}\,, \qquad w_{40} = -\,\frac{216}{125}\,, \tag{9}$$

and suppose that the input word $\mathbf{x} = 1011 \in \{0,1\}^4$ of length $n = 4$ is externally presented to $\mathcal{N}$ where $x_1 = 1$, $x_2 = 0$, $x_3 = 1$, $x_4 = 1$, and formally let $x_5 = 0$. Table 1 shows the sequential schedule of presenting the bits x_1, x_2, x_3, x_4 of $\mathbf{x}$ to $\mathcal{N}$ through the input neuron inp at the time instants $t = 1, 4, 7, 10$, respectively, which is indicated in boldface. Each input bit is queried by the neuron nxt one step beforehand according to (6). Thus, the neuron nxt is the only initially active unit, that is, $y_{\text{nxt}}^{(0)} = 1$, and this activity propagates repeatedly around the oriented cycle composed of

t	$y_1^{(t)}$	$y_2^{(t)}$	$y_3^{(t)}$	$y_4^{(t)}$	$y_{\mathrm{inp}}^{(t)}$	$y_{\mathrm{nxt}}^{(t)}$	$y_{\mathrm{out}}^{(t)}$	the result of recognition
0	0	0	0	0	0	1	0	
1	0	1	0	0	**1**	0	**1**	$\varepsilon \in \mathcal{L}(\mathcal{N})$
2	$\frac{91}{216}$	0	1	0	0	0	0	
3	$\frac{455}{1296}$	0	0	0	0	1	0	
4	$\frac{2275}{7776}$	1	0	0	**0**	0	**1**	$1 \in \mathcal{L}(\mathcal{N})$
5	$\frac{11375}{46656}$	0	1	0	0	0	0	
6	$\frac{56875}{279936}$	0	0	0	0	1	0	
7	$\frac{284375}{1679616}$	1	0	0	**1**	0	**1**	$10 \in \mathcal{L}(\mathcal{N})$
8	$\frac{5667571}{10077696}$	0	1	0	0	0	0	
9	$\frac{28337855}{60466176}$	0	0	0	0	1	0	
10	$\frac{141689275}{362797056}$	1	0	0	**1**	0	**1**	$101 \in \mathcal{L}(\mathcal{N})$
11	$\frac{1625516711}{2176782336}$	0	1	0	0	0	0	
12	$\frac{8127583555}{13060694016}$	0	0	1	0	1	0	
13	$\frac{40637917775}{78364164096}$	1	0	0	0	0	**0**	$1011 \notin \mathcal{L}(\mathcal{N})$

Table 1: The rejecting computation by the 1ANN $\mathcal{N}$ from Figure 2 with parameters (8), on the input 1011.

three neurons $2, 3, \mathrm{nxt} \in V'$ through the edges with the unit weights, which ensures the neuron nxt fires at the time instants $t = 3k$ for $k \geq 0$, when the next input bits are prompted. In addition, the units 3 and nxt from this cycle synchronize the incident neurons 4 and out, respectively, so that the unit 4 can be activated only at the time instants $t = 3k$, whereas the output neuron out can fire only at $t = 3k + 1$. Hence, the result of the recognition is reported by the output neuron out as indicated in Table 1 in boldface, even for each of the five prefixes of $\mathbf{x}$, the empty string ε, 1, 10, 101 and 1011, at the time steps $t = 1, 4, 7, 10, 13$, respectively, according to (7).

According to (2)–(4), we obtain the recurrence equation for the analog

state of unit $1 \in V$,

$$y_1^{(t)} = \xi_1^{(t-1)} = w_{1,\mathrm{inp}}\, y_{\mathrm{inp}}^{(t-1)} + w_{11}\, y_1^{(t-1)} = \frac{\beta^{-1}}{\nu}\, y_{\mathrm{inp}}^{(t-1)} + \beta^{-\frac{1}{3}}\, y_1^{(t-1)} \tag{10}$$

at time instant $t \geq 1$, where $y_1^{(t)} = \xi_1^{(t-1)} \in \mathbb{I}$ by the definition of parameter ν. Hence, the input bits $y_{\mathrm{inp}}^{(1)} = x_1$, $y_{\mathrm{inp}}^{(4)} = x_2$, etc. are encoded in this analog state as

$$y_1^{(1)} = y_1^{(0)} = 0 \tag{11}$$

$$y_1^{(2)} = \frac{\beta^{-1}}{\nu}\, x_1 \tag{12}$$

$$y_1^{(4)} = \beta^{-\frac{1}{3}}\, y_1^{(3)} = \beta^{-\frac{2}{3}}\, y_1^{(2)} = \frac{\beta^{-\frac{5}{3}}}{\nu}\, x_1 \tag{13}$$

$$y_1^{(5)} = \frac{1}{\nu}\left(x_2\beta^{-1} + x_1\beta^{-2}\right)\,, \tag{14}$$

which generalizes to

$$y_1^{(3k-1)} = \frac{1}{\nu}\sum_{i=1}^{k} x_{k-i+1}\,\beta^{-i}\,. \tag{15}$$

It follows that the neuron $4 \in V'$, activating only at the time instant $t = 3k$, satisfies $y_4^{(3k)} = 1$ iff $\xi_4^{(3k-1)} = w_{40} + w_{41}\, y_1^{(3k-1)} + w_{43}\, y_3^{(3k-1)} \geq 0$ iff

$$-1 - \frac{c}{\nu} + 1 + \frac{1}{\nu}\sum_{i=1}^{k} x_{k-i+1}\,\beta^{-i} \geq 0 \tag{16}$$

according to (2), (3), (5), and (15), which reduces to

$$y_4^{(3k)} = 1 \quad \text{iff} \quad \sum_{i=1}^{k} x_{k-i+1}\,\beta^{-i} \geq c\,. \tag{17}$$

At the time instant $t = 3k + 1$, the output neuron $\mathrm{out} \in V'$ computes the negation of $y_4^{(3k)}$, and hence,

$$y_{\mathrm{out}}^{(3k+1)} = 1 \quad \text{iff} \quad \sum_{i=1}^{k} x_{k-i+1}\,\beta^{-i} < c\,. \tag{18}$$

It follows from (18) that the neural language acceptor $\mathcal{N}$ accepts the reversal of the cut language[2],

$$\mathcal{L}(\mathcal{N}) = L^R_{<c} = \left\{ x_1 \ldots x_n \in \{0,1\}^* \;\middle|\; \sum_{k=1}^{n} x_{n-k+1}\, \beta^{-k} < c \right\} . \tag{19}$$

Since the threshold $c = 1$ is not a quasi-periodic number[4] for the base $\beta = \frac{216}{125}$ and the binary digits $\{0, 1\}$, the corresponding instance of (19),

$$\mathcal{L}(\mathcal{N}) = L^R_{<1} = \left\{ x_1 \ldots x_n \in \{0,1\}^* \;\middle|\; \sum_{k=1}^{n} x_{n-k+1} \left(\frac{216}{125}\right)^{-k} < 1 \right\} , \tag{20}$$

is a context-sensitive language that is not *context-free* (Šíma and Savický, 2018).

In contrast, if we choose the integer (Pisot) base and a quasi-periodic[4] threshold for this base,

$$\beta = 3^3 = 27 \qquad \text{and} \qquad c = \frac{1}{52} \tag{21}$$

(cf. (8)) for defining another instance of the 1ANN in Figure 2, say $\mathcal{N}'$, then the language accepted by $\mathcal{N}'$, which instantiates (19) as

$$\mathcal{L}(\mathcal{N}') = L^R_{<\frac{1}{52}} = \left\{ x_1 \ldots x_n \in \{0,1\}^* \;\middle|\; \sum_{k=1}^{n} x_{n-k+1}\, 27^{-k} < \frac{1}{52} \right\} , \tag{22}$$

is *regular* (Šíma and Savický, 2018). The description of language (22) can be simplified as

$$\mathcal{L}(\mathcal{N}') = \{ x_1 \ldots x_n \in \{0,1\}^* \mid x_n = 0 \} , \tag{23}$$

since for any $x_1 \ldots x_{n-1}0 \in \{0,1\}^*$, we have $\sum_{k=1}^{n} x_{n-k+1}\, 27^{-k} < \sum_{k=2}^{\infty} 27^{-k} = \frac{1}{702} < \frac{1}{52}$, whereas $\sum_{k=1}^{n} x_{n-k+1}\, 27^{-k} \geq \frac{1}{27} > \frac{1}{52}$ for every $x_1 \ldots x_{n-1}1 \in \{0,1\}^*$.

3. Separating One Analog Neuron

In this section, we present an example of a DCFL, $L_{\#} = \{0^n 1^n \mid n \geq 1\}$ containing the words of n zeros followed by n ones, which cannot be accepted

offline by any 1ANN with one extra analog unit even with real weights, which means $L_{\#} \notin$ 1ANNs. This provides a separation of the second level of the analog neuron hierarchy, that is, 1ANNs $\subsetneq$ 2ANNs (see Figure 1) since in Section 4 we show that 2ANNs can simulate any DPDA.

The main idea of the proof is based on the fact that a 1ANN $\mathcal{N}$ that would recognize $L_{\#} = \mathcal{L}(\mathcal{N})$ must remember the count of the initial segment of zeros in an input word because this must later be compared to the number of subsequent ones in order to decide whether the input is accepted. However, this count is unbounded while $\mathcal{N}$ has only finitely many possible binary states. Thus, this number can just be encoded by using a real state of the one analog neuron. By presenting a series of zeros as an input to $\mathcal{N}$, we obtain an infinite bounded sequence of these real analog-state values which has a monotone convergent subsequence according to the Bolzano-Weierstrass theorem[10].

This subsequence is further pruned so that it remains infinite while the following condition is satisfied. Starting the computation of $\mathcal{N}$ with any analog value from this pruned convergent subsequence, the binary states enter the same cycle in a while when a subsequent series of ones is presented to $\mathcal{N}$, which induces a periodic behavior of $\mathcal{N}$ in the limit. This periodicity provides only a finite number of thresholds for separating an infinite number of analog values from each other. However, these analog values that are indistinguishable by $\mathcal{N}$, encode the original counts of zeros. This means that $\mathcal{N}$ would not differentiate between two input words composed of a distinct number of initial zeros and could thus not recognize the language $L_{\#}$ correctly, which is a contradiction. The technical details are presented in the following proof which is, for clarity, split into subsections and lemmas.

Theorem 1 *The deterministic context-free language* $L_{\#} = \{0^n 1^n \,|\, n \geq 1\}$ *cannot be recognized by a 1ANN with one extra analog unit having real weights.*

Proof. On the contrary, assume that $\mathcal{N}$ is a neural network 1ANN with one extra analog unit such that $L_{\#} = \mathcal{L}(\mathcal{N})$. Let $y_j^{(t)}(\mathbf{x})$ and $\xi_j^{(t)}(\mathbf{x})$ be the state and the excitation of neuron $j \in V$ at time instant $t \geq 0$, respectively, when an input word $\mathbf{x} \in \{0,1\}^n$ of length n is presented to $\mathcal{N}$, which satisfies $t < \tau_{n+1}$ by (6). Formally, we also allow infinite input strings $\mathbf{x} \in \{0,1\}^\omega$ where Σ^ω denotes the set of all infinite words (ω-words) over Σ. Denote by $\mathbf{y}^{(t)}(\mathbf{x}) =$

[10]Each bounded sequence of real numbers has a monotone convergent subsequence.

$\left(y_1^{(t)}(\mathbf{x}),\ldots,y_s^{(t)}(\mathbf{x})\right) \in \mathbb{I}\times\{0,1\}^{s-1}$ and $\tilde{\mathbf{y}}^{(t)}(\mathbf{x}) = \left(y_2^{(t)}(\mathbf{x}),\ldots,y_s^{(t)}(\mathbf{x})\right) \in \{0,1\}^{s-1}$ the corresponding global network state, respectively restricted to binary neurons $V' = V \setminus \{1\}$.

3.1. Analog States as β-Expansions When $\mathcal{N}$ Reads Zeros

For the infinite input string 0^ω, there exists $t_0 \geq 0$ such that the states of the analog unit meet

$$y_1^{(t_0)}(0^\omega) \in \{0,1\} \quad \text{and} \quad 0 < y_1^{(t)}(0^\omega) < 1 \text{ for every } t > t_0 \tag{24}$$

(we know $y_1^{(0)}(0^\omega) \in \{0,1\}$ by definition), since otherwise there would be infinitely many time instants t with $y_1^{(t)}(0^\omega) \in \{0,1\}$. This means that $\mathcal{N}$ would find in the same state $\mathbf{y}^{(t)}(0^\omega) = \mathbf{y}$ for infinitely many t and some $\mathbf{y} \in \{0,1\}^s$, due to $\{0,1\}^s$ is finite. Hence, there would exist $t_1 < t_2$ such that $\mathbf{y}^{(t_1)}(0^\omega) = \mathbf{y}^{(t_2)}(0^\omega) = \mathbf{y}$ and $n_1 < n_2$ where n_i is the number of input zeros that has been read by $\mathcal{N}$ until the time instant t_i for $i \in \{1,2\}$. Thus, $\mathbf{y}^{(t_1)}(0^{n_1}) = \mathbf{y}^{(t_2)}(0^{n_2}) = \mathbf{y}$ for $n_1 < n_2$, which implies $\mathbf{y}^{(t_1+t)}(0^{n_1}1^{n_1}) = \mathbf{y}^{(t_2+t)}(0^{n_2}1^{n_1})$ for every $t \geq 0$. It follows that $0^{n_2}1^{n_1} \in \mathcal{L}(\mathcal{N})$ because of $0^{n_1}1^{n_1} \in L_\# = \mathcal{L}(\mathcal{N})$, which means $\mathcal{N}$ would accept incorrectly the input word $0^{n_2}1^{n_1} \notin L_\#$. For the same reason, the self-loop weight meets

$$w_{11} \neq 0 \tag{25}$$

since for $w_{11} = 0$, the analog unit could produce only a finite number of output values $y_1^{(t)} \in \left\{ \sum_{i\in V'} w_{1i}y_i \;\middle|\; (y_2,\ldots,y_s) \in \{0,1\}^{s-1} \right\}$ for $t > t_0$, according to (2)–(4).

Define the *base*

$$\beta = \frac{1}{w_{11}} \tag{26}$$

which is a correct definition due to (25), and the set of *digits*,

$$A = \left\{ \beta \sum_{i\in V'} w_{1i}y_i \;\middle|\; (y_2,\ldots,y_s) \in \{0,1\}^{s-1} \right\} \cup \{0,\beta\}\,. \tag{27}$$

We introduce an infinite sequence of digits, $a_1a_2a_3\ldots \in A^\omega$ as

$$a_k = \begin{cases} \beta\, y_1^{(t_0)}(0^\omega) \in \{0,\beta\} \subseteq A & \text{for } k = 1 \\ \beta \sum_{i\in V'} w_{1i}y_i^{(t_0+k-2)}(0^\omega) \in A & \text{for } k \geq 2\,. \end{cases} \tag{28}$$

For every $t \geq t_0$, we obtain the recurrence

$$y_1^{(t+1)}(0^\omega) = \sum_{i=0}^{s} w_{1i} y_i^{(t)}(0^\omega) = \beta^{-1} \left(a_{t-t_0+2} + y_1^{(t)}(0^\omega) \right) \tag{29}$$

by (2)–(4), (26), and (28), which can be solved as

$$y_1^{(t)}(0^\omega) = \sum_{k=1}^{t-t_0+1} a_{t-t_0-k+2}\, \beta^{-k} \,. \tag{30}$$

It follows from formula (30) which represents analog states as β-expansions using the digits from A, that

$$|\beta| > 1 \tag{31}$$

because $0 < y_1^{(t)}(0^\omega) < 1$ for every $t > t_0$, according to (24).

3.2. Monotone Convergent Subsequence of Analog States

Consider an infinite sequence of time instants $0 < t_1 < t_2 < t_3 < \cdots$ such that for each n, $t_n = \tau_{n+1} - 1$ is the last time instant before the next $(n+1)$th bit is presented to $\mathcal{N}$ after the input 0^n has been read, that is, $y_{\text{nxt}}^{(t_n)}(0^n) = 1$ by (6). Since the infinite sequence of real numbers $y_1^{(t_n)}(0^n) \in \mathbb{I}$ for $n \geq 1$, is bounded, according to Bolzano-Weierstrass theorem[10], there exists its monotone convergent subsequence $y_1^{(t_{n_p})}(0^{n_p}) \in (0\,,1)$ for $p \geq 1$, where $t_{n_1} > t_0$, $n_1 < n_2 < n_3 < \cdots$, and denote the respective limit by

$$c_0 = \lim_{p \to \infty} y_1^{(t_{n_p})}(0^{n_p}) \,. \tag{32}$$

We assume that this subsequence is nondecreasing, that is,

$$y_1^{(t_{n_p})}(0^{n_p}) \leq y_1^{(t_{n_{p+1}})}(0^{n_{p+1}}) \quad \text{for every } p \geq 1 \,, \tag{33}$$

while the argument for a nonincreasing subsequence is analogous (cf. Footnote 11). In the following considerations, we will repeatedly remove some elements from the sequence (n_p) given by Bolzano-Weierstrass theorem, so that infinitely many elements remain, which satisfy additional conditions. For simplicity, we will keep the original notation (n_p) for these pruned sequences without loss of generality.

There are only finitely many possible states of binary neurons taken from $\{0,1\}^{s-1}$, and hence, there exists $\tilde{\mathbf{u}} \in \{0,1\}^{s-1}$ which occurs infinitely many

times in the corresponding subsequence $\tilde{\mathbf{y}}^{(t_{n_p})}(0^{n_p})$ for $p \geq 1$. By skipping the remaining elements, we can assume without loss of generality that

$$\tilde{\mathbf{y}}^{(t_{n_p})}(0^{n_p}) = \tilde{\mathbf{u}} \quad \text{for every } p \geq 1\,. \tag{34}$$

It follows that the subsequence $y_1^{(t_{n_p})}(0^{n_p})$ for $p \geq 1$, is increasing since for $y_1^{(t_{n_p})}(0^{n_p}) = y_1^{(t_{n_{p+1}})}(0^{n_{p+1}})$, we would have $\mathbf{y}^{(t_{n_p})}(0^{n_p}) = \mathbf{y}^{(t_{n_p})}(0^{n_{p+1}})$ by (34), and hence, the input $0^{n_{p+1}}1^{n_p} \notin L_{\#}$ would be incorrectly accepted by $\mathcal{N}$. Thus,

$$y_1^{(t_{n_p})}(0^{n_p}) < y_1^{(t_{n_{p+1}})}(0^{n_{p+1}}) \quad \text{for every } p \geq 1\,. \tag{35}$$

3.3. Binary States When $\mathcal{N}$ Reads Ones

We will inductively construct an increasing infinite sequence (m_p) of natural numbers $m_p \geq 0$, which satisfies the following conditions by pruning the corresponding sequence (n_p) so that the number of elements in (n_p) remains infinite. The number m_p is the maximum length of the computational trajectory since the time instant t_{n_p} when $\mathcal{N}$ starts to read ones, such that the same binary states are traversed for every greater n_p. Namely, for each $p \geq 1$ and for every $q > p$, the binary states in $\mathcal{N}$ at the time instants $t_{n_p}, t_{n_p+1}, \ldots, t_{n_p+m_p}, t_{n_p+m_p+1}$ and $t_{n_q}, t_{n_q+1}, \ldots, t_{n_q+m_p}, t_{n_q+m_p+1}$ after $\mathcal{N}$ reads n_p and n_q zeros, respectively, followed by (at most n_p) ones, will meet

$$\tilde{\mathbf{y}}^{(t_{n_p}+k)}(0^{n_p}1^{n_p}) = \tilde{\mathbf{y}}^{(t_{n_q}+k)}(0^{n_q}1^{n_p}) \quad \text{for every } k = 0, 1, \ldots, m_p \tag{36}$$

$$\tilde{\mathbf{y}}^{(t_{n_p}+m_p+1)}(0^{n_p}1^{n_p}) \neq \tilde{\mathbf{y}}^{(t_{n_q}+m_p+1)}(0^{n_q}1^{n_p})\,. \tag{37}$$

This means that for the increasing number n_p of the initial input zeros 0^{n_p} that have been read by $\mathcal{N}$ at the time instant t_{n_p}, the binary states $\tilde{\mathbf{y}}^{(t_{n_p}+k)}(0^{n_p}1^{n_p})$ for $k = 0, \ldots, m_p$, further follow the same computational trajectory for a period of m_p computational steps when $\mathcal{N}$ reads the subsequent input ones 1^{n_p}. Moreover, this period length m_p will be shown below to be increasing since the corresponding analog state $y_1^{(t_{n_p})}(0^{n_p})$ is converging by (32). Observe that by definition, $m_p \leq m_{p+1}$, and condition (36) holds at least for $k = 0$, according to (34), whereas condition (37) is met before the next input bit is presented to $\mathcal{N}$ after the input $0^{n_p}1^{n_p} \in L_{\#}$ has been read, due to $0^{n_q}1^{n_p} \notin L_{\#}$ for $q > p$.

Suppose $m_1 < m_2 < \cdots < m_{p-1}$ have been constructed, satisfying (36) and (37). For the next index $p \geq 1$, let $\tilde{m}_p \geq 0$ be the maximal natural

number that meets (36) with m_p replaced by $\tilde{m}_p$, which means $\tilde{m}_p \geq m_{p-1}$. On the contrary assume that $\tilde{m}_p = m_{p-1}$. There exists $\tilde{\mathbf{u}}' \in \{0,1\}^{s-1}$ such that the set

$$Q = \left\{ q \geq p \;\middle|\; \tilde{\mathbf{y}}^{(t_{n_q}+\tilde{m}_p+1)}(0^{n_q}1^{n_p}) = \tilde{\mathbf{u}}' \right\} \tag{38}$$

is infinite since there are only 2^{s-1} possible states of binary neurons. We omit all the elements n_q in (n_p) such that $p \leq q \notin Q$, while the pruned sequence (n_p), including the indices from infinite Q, remains infinite, and $p = \min Q$ is the new succeeding index in the pruned (n_p). In addition, the new maximal value of $\tilde{m}_p$ satisfying (36) for this index p, increases by at least 1 according to (38), and hence, we have $\tilde{m}_p > m_{p-1}$.

Moreover, we can assume without loss of generality that there are infinitely many indices q that meet (37) with m_p replaced by $\tilde{m}_p$, since otherwise we could skip them in (n_p), while increasing $\tilde{m}_p$. Thus, the constructed sequence $m_1, \ldots, m_{p-1}$ is extended with $m_p = \tilde{m}_p > m_{p-1}$ and the sequence (n_p) is further pruned by removing those indices $q > p$ for which (37) is not satisfied. This completes the inductive construction which ensures the sequence (m_p) which corresponds to (n_p) and satisfies (36) and (37), is increasing, and hence unbounded. Hereafter, we assume there are infinitely many even numbers in (m_p) while the proof for the opposite case when there are infinitely many odd numbers in (m_p), is analogous (cf. Footnote 11). Thus, by pruning the sequence (n_p) we can assume without loss of generality that m_p is even for every $p \geq 1$.

3.4. Analog States as β-Expansions When $\mathcal{N}$ Reads Ones

For each $p \geq 1$, define m'_p to be the maximum number such that $0 \leq m'_p \leq m_p$ and

$$0 \leq \xi_1^{(t_{n_p}+k)}(0^{n_p}1^{n_p}) \leq 1 \quad \text{for every } k = 0, \ldots, m'_p\,, \tag{39}$$

which holds at least for $k = 0$ because $\xi_1^{(t_{n_p})}(0^{n_p}1^{n_p}) = y_1^{(t_{n_p}+1)}(0^{n_{p+1}}) \in (0\,,1)$ according to (24). We introduce

$$b_k = \beta \sum_{i \in V'} w_{1i} y_i^{(t_{n_p}+k-1)}(0^{n_p}1^{n_p}) \in A \quad \text{for } k = 1, \ldots, m_p + 1\,, \tag{40}$$

which is a consistent definition for different $p \geq 1$, due to (36). We obtain the recurrence

$$\begin{aligned} y_1^{(t_{n_p}+k)}(0^{n_p}1^{n_p}) &= \sum_{i=0}^{s} w_{1i} y_i^{(t_{n_p}+k-1)}(0^{n_p}1^{n_p}) \\ &= \beta^{-1}\left(b_k + y_1^{(t_{n_p}+k-1)}(0^{n_p}1^{n_p})\right) \text{ for } k = 1, \ldots, m'_p + 1 \quad (41) \end{aligned}$$

by (2)–(4), (26), (39) and (40), which can be solved as

$$\xi_1^{(t)}(0^{n_p}1^{n_p}) = \beta^{-(t-t_{n_p}+1)} y_1^{(t_{n_p})}(0^{n_p}) + \sum_{k=1}^{t-t_{n_p}+1} b_{t-t_{n_p}-k+2}\, \beta^{-k}\,. \quad (42)$$

for each $p \geq 1$ and $t_{n_p} \leq t \leq t_{n_p} + \min(m'_p + 1, m_p)$.

In the following lemma, we will show that m'_p coincides with m_p. This means that the excitation $\xi_1^{(t_{n_p}+k)}(0^{n_p}1^{n_p})$ of the analog neuron stays in the unit interval $\mathbb{I}$, satisfying (39) for the whole period of m_p computational steps as long as the corresponding binary states $\tilde{\mathbf{y}}^{(t_{n_p}+k)}(0^{n_p}1^{n_p})$ for $k = 0, \ldots, m_p$, follow the underlying computational trajectory that meets (36) and (37). Informally, if this is not the case, then the analog neuron would saturate at the same binary state $y_1^{(t_{n_p}+m'_p+1)}(0^{n_p}1^{n_p}) = y_1^{(t_{n_q}+m'_p+1)}(0^{n_q}1^{n_p}) \in \{0, 1\}$ for different $p < q$ because of (4) and (32). This fact together with the coinciding states $\tilde{\mathbf{y}}^{(t_{n_p}+m'_p+1)}(0^{n_p}1^{n_p}) = \tilde{\mathbf{y}}^{(t_{n_q}+m'_p+1)}(0^{n_q}1^{n_p})$ of binary neurons by (36) for $m'_p < m_p$, would not allow $\mathcal{N}$ to distinguish between the inputs $0^{n_p}1^{n_p}$ and $0^{n_q}1^{n_p}$, which is a contradiction. The technical details are presented in the following proof of Lemma 1.

Lemma 1 *For every $p \geq 1$, $m'_p = m_p$.*

Proof. Clearly, we can skip a finite number of elements in (n_p) for which $m'_p < m_p$. Thus, on the contrary assume there are infinitely many p such that $m'_p < m_p$. By pruning the sequence (n_p), we can assume without loss of generality that for every $p \geq 1$, it holds $m'_p < m_p$ and $m'_p \leq m'_{p+1}$ because any sequence of natural numbers contains an infinite nondecreasing subsequence. According to (42),

$$\lim_{p\to\infty}\left(\xi_1^{(t_{n_{p+1}}+m'_p+1)}(0^{n_{p+1}}1^{n_{p+1}}) - \xi_1^{(t_{n_p}+m'_p+1)}(0^{n_p}1^{n_p})\right)$$

$$
\begin{aligned}
&= \lim_{p\to\infty} \left(\beta^{-(m'_p+2)} y_1^{(t_{n_{p+1}})}(0^{n_{p+1}}) + \sum_{k=1}^{m'_p+1} b_{m'_p-k+3}\, \beta^{-k} \right. \\
&\qquad \left. -\beta^{-(m'_p+2)} y_1^{(t_{n_p})}(0^{n_p}) - \sum_{k=1}^{m'_p+1} b_{m'_p-k+3}\, \beta^{-k} \right) \\
&= \lim_{p\to\infty} \beta^{-(m'_p+2)} \left(y_1^{(t_{n_{p+1}})}(0^{n_{p+1}}) - y_1^{(t_{n_p})}(0^{n_p}) \right) = 0 \qquad (43)
\end{aligned}
$$

due to (31) and (32). By the maximality of m'_p, we know $\xi_1^{(t_{n_p}+m'_p+1)}(0^{n_p}1^{n_p}) \notin \mathbb{I}$. Hence, for a sufficiently large p, $\xi_1^{(t_{n_{p+1}}+m'_p+1)}(0^{n_{p+1}}1^{n_{p+1}}) \notin \mathbb{I}$ and $\xi_1^{(t_{n_{p+2}}+m'_p+1)}(0^{n_{p+2}}1^{n_{p+1}}) \notin \mathbb{I}$, according to (43), which means $m'_p = m'_{p+1} = m'_{p+2} < m_{p+1} - 1$ due to $m'_p < m_p < m_{p+1}$. It follows from (3) and (4) that $y_1^{(t_{n_{p+1}}+m'_p+2)}(0^{n_{p+1}}1^{n_p}) = y_1^{(t_{n_{p+2}}+m'_p+2)}(0^{n_{p+2}}1^{n_p}) \in \{0,1\}$ which gives $\mathbf{y}^{(t_{n_{p+1}}+m'_p+2)}(0^{n_{p+1}}1^{n_p}) = \mathbf{y}^{(t_{n_{p+2}}+m'_p+2)}(0^{n_{p+2}}1^{n_p})$ by (36) for $p+1$ and $k = m'_p + 2 \le m_{p+1}$, implying the contradiction $0^{n_{p+2}}1^{n_{p+1}} \in L(\mathcal{N})$. This completes the proof that $m'_p = m_p$ for every $p \ge 1$. □

It follows from Lemma 1 that formula (42) provides β-expansions of analog states

$$
y_1^{(t)}(0^{n_p}1^{n_p}) = \xi_1^{(t-1)}(0^{n_p}1^{n_p}) = \beta^{-(t-t_{n_p})} y_1^{(t_{n_p})}(0^{n_p}) + \sum_{k=1}^{t-t_{n_p}} b_{t-t_{n_p}-k+1}\, \beta^{-k} \quad (44)
$$

for every $p \ge 1$ and $t_{n_p} \le t \le t_{n_p} + m_p + 1$, when $\mathcal{N}$ reads ones preceded by n_p zeros, using the digits from A, cf. (30).

3.5. Asymptotic Analog States

It follows from (35) and (44) that for every $p \ge 1$,

$$
\begin{aligned}
y_1^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p}) &= \beta^{-m_p}\, y_1^{(t_{n_p})}(0^{n_p}) + \sum_{k=1}^{m_p} b_{m_p-k+1}\, \beta^{-k} \\
< \beta^{-m_p}\, y_1^{(t_{n_{p+1}})}(0^{n_{p+1}}) &+ \sum_{k=1}^{m_p} b_{m_p-k+1}\, \beta^{-k} = y_1^{(t_{n_{p+1}}+m_p)}(0^{n_{p+1}}1^{n_p}) \qquad (45)
\end{aligned}
$$

due to $y_1^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p}) = \xi_1^{(t_{n_p}+m_p-1)}(0^{n_p}1^{n_p})$ and m_p is even[11]. There exists $\tilde{\mathbf{v}} \in \{0,1\}^{s-1}$ such that $\tilde{\mathbf{y}}^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p}) = \tilde{\mathbf{v}}$ for infinitely many $p \geq 1$, since there are only 2^{s-1} states of binary neurons, and by pruning the sequence (n_p), we can assume without loss of generality that

$$\tilde{\mathbf{y}}^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p}) = \tilde{\mathbf{v}} \quad \text{for every } p \geq 1\,. \tag{46}$$

Similarly, assume without loss of generality that there exists a binary neuron $j_0 \in \{2,\ldots,s\}$ such that

$$y_{j_0}^{(t_{n_p}+m_p+1)}(0^{n_p}1^{n_p}) \neq y_{j_0}^{(t_{n_{p+1}}+m_p+1)}(0^{n_{p+1}}1^{n_p}) \quad \text{for every } p \geq 1\,, \tag{47}$$

according to (37), since there are only $s-1$ binary neurons. It follows from (47) that $w_{j_0,1} \neq 0$ because $\tilde{\mathbf{y}}^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p}) = \tilde{\mathbf{y}}^{(t_{n_{p+1}}+m_p)}(0^{n_{p+1}}1^{n_p})$ by (36), and we can thus define

$$c = -\frac{1}{w_{j_0,1}} \sum_{i \in V'} w_{j_0,i}\, y_i^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p})\,, \tag{48}$$

which is a consistent definition for different $p \geq 1$ due to (46).

Hereafter, assume $w_{j_0,1} > 0$, while the argument for $w_{j_0,1} < 0$ is analogous[12]. We have $y_{j_0}^{(t_{n_p}+m_p+1)}(0^{n_p}1^{n_p}) = 1$ iff $\xi_{j_0}^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p}) \geq 0$ iff $y_1^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p}) \geq c$, according to (2), (3), (5), and (48), and similarly, $y_{j_0}^{(t_{n_{p+1}}+m_p+1)}(0^{n_{p+1}}1^{n_p}) = 1$ iff $y_1^{(t_{n_{p+1}}+m_p)}(0^{n_{p+1}}1^{n_p}) \geq c$ because of (46), which implies

$$y_1^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p}) < c \leq y_1^{(t_{n_{p+1}}+m_p)}(0^{n_{p+1}}1^{n_p}) \tag{49}$$

by (45) and (47), corresponding to

$$0 = y_{j_0}^{(t_{n_p}+m_p+1)}(0^{n_p}1^{n_p}) \neq y_{j_0}^{(t_{n_{p+1}}+m_p+1)}(0^{n_{p+1}}1^{n_p}) = 1\,. \tag{50}$$

[11] Note that the less-than sign in (45) is replaced by the greater-than sign if either the sequence $y_1^{(t_{n_p})}(0^{n_p})$ is nonincreasing which is the opposite assumption to (33) and m_p remains even, or m_p is odd and (33) holds.

[12] Note that for $w_{j_0,1} < 0$, inequality (49) reads $y_1^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p}) \leq c < y_1^{(t_{n_{p+1}}+m_p)}(0^{n_{p+1}}1^{n_p})$.

According to (45) and (49), we obtain

$$y_1^{(t_{n_p})}(0^{n_p}) < \beta^{m_p}\left(c - \sum_{k=1}^{m_p} b_{m_p-k+1}\,\beta^{-k}\right) \le y_1^{(t_{n_{p+1}})}(0^{n_{p+1}}) \tag{51}$$

for every $p \ge 1$, due to m_p is even, which implies

$$\lim_{p\to\infty} \sum_{k=1}^{m_p} b_{m_p-k+1}\,\beta^{-k} = c \tag{52}$$

due to (31) and (32). For $p \ge 1$, we have

$$c_p = \lim_{q\to\infty} y_1^{(t_{n_q}+m_p)}(0^{n_q}1^{n_p}) = \beta^{-m_p}c_0 + \sum_{k=1}^{m_p} b_{m_p-k+1}\beta^{-k} \tag{53}$$

according to (32) and (44), which implies

$$\lim_{p\to\infty} c_p = c \tag{54}$$

by (52).

3.6. Periodic Binary States

It follows from (51), (32), and (35) that for every $p \ge 1$ and $q > p$,

$$\beta^{m_p}\left(c - \sum_{k=1}^{m_p} b_{m_p-k+1}\,\beta^{-k}\right) \le y_1^{(t_{n_{p+1}})}(0^{n_{p+1}}) \le y_1^{(t_{n_q})}(0^{n_q}) < c_0 \tag{55}$$

which implies

$$c \le y_1^{(t_{n_{p+1}}+m_p)}(0^{n_{p+1}}1^{n_p}) \le y_1^{(t_{n_q}+m_p)}(0^{n_q}1^{n_p}) < c_p \tag{56}$$

by (45), (44), and (53) due to m_p is even. For notational simplicity, we further assume $\beta > 1$ while for $\beta < -1$ the argument is similar[13]. Thus, we can define the intervals,

$$I_{p,r} = \left[\beta^{-r}c + \sum_{k=1}^{r} b_{m_p+r-k+1}\beta^{-k}\,,\ \beta^{-r}c_p + \sum_{k=1}^{r} b_{m_p+r-k+1}\beta^{-k}\right) \tag{57}$$

[13]For $\beta < -1$, the endpoints of intervals (57) are swapped for odd r.

for every $p \geq 1$ and $r = 0, \ldots, \ell_p - 1$, where $\ell_p = m_{p+1} - m_p$ is even, for which inequality (56) generalizes to

$$y_1^{(t_{n_q}+m_p+r)}(0^{n_q}1^{n_{p+1}}) \in I_{p,r} \quad \text{for every } q > p\,, \tag{58}$$

since $\beta > 1$ and

$$\begin{aligned} y_1^{(t_{n_q}+m_p+r)}(0^{n_q}1^{n_{p+1}}) &= \beta^{-(m_p+r)} y_1^{(t_{n_q})}(0^{n_q}) + \sum_{k=1}^{m_p+r} b_{m_p+r-k+1}\beta^{-k} \\ = \beta^{-r}\left(\beta^{-m_p} y_1^{(t_{n_q})}(0^{n_q}) \right. &+ \left. \sum_{k=1}^{m_p} b_{m_p-k+1}\beta^{-k}\right) + \sum_{k=1}^{r} b_{m_p+r-k+1}\beta^{-k} \\ &= \beta^{-r} y_1^{(t_{n_q}+m_p)}(0^{n_q}1^{n_p}) + \sum_{k=1}^{r} b_{m_p+r-k+1}\beta^{-k} \end{aligned} \tag{59}$$

according to (44).

In the following lemma, we show by induction on r that for sufficiently large p, the intervals $I_{p,r}$ shrink for fixed r and the binary states become periodic when $\mathcal{N}$ reads ones preceded by n_p zeros.

Lemma 2 *There exists an integer $p_0 \geq 1$ such that for every $p \geq p_0$ and $r = 0, \ldots, \ell_p - 1$,*

$$I_{p+1,r} \subset I_{p,r}\,, \tag{60}$$

$$\tilde{\mathbf{y}}^{(t_{n_{p+1}}+m_p+r)}(0^{n_{p+1}}1^{n_{p+1}}) = \tilde{\mathbf{y}}^{(t_{n_{p+2}}+m_{p+1}+r)}(0^{n_{p+2}}1^{n_{p+2}})\,. \tag{61}$$

Proof. For a sufficiently large p, we proceed by induction on $r = 0, \ldots, \ell_p - 1$. For the base case $r = 0$, the length of $I_{p,0} = [c, c_p)$ is β^{ℓ_p} times greater than that of the interval

$$I_{p+1} = \left[\beta^{-\ell_p} c + \sum_{k=1}^{\ell_p} b_{m_{p+1}-k+1}\beta^{-k}\,,\; c_{p+1}\right) \tag{62}$$

because $c_{p+1} = \beta^{-\ell_p} c_p + \sum_{k=1}^{\ell_p} b_{m_{p+1}-k+1}\beta^{-k}$ by (53). It follows from (49) and (44) that

$$\begin{aligned} \beta^{-\ell_p} c + \sum_{k=1}^{\ell_p} b_{m_{p+1}-k+1}\beta^{-k} &\leq \beta^{-\ell_p} y_1^{(t_{n_{p+1}}+m_p)}(0^{n_{p+1}}1^{n_p}) + \sum_{k=1}^{\ell_p} b_{m_{p+1}-k+1}\beta^{-k} \\ &= y_1^{(t_{n_{p+1}}+m_{p+1})}(0^{n_{p+1}}1^{n_{p+1}}) < c\,, \end{aligned} \tag{63}$$

which means $I_{p+1,0} = [c, c_{p+1}) \subset I_{p+1}$. Hence, $c_{p+1} < c_p$ and $I_{p+1,0} \subset I_{p,0}$. In addition,

$$\begin{aligned}\tilde{\mathbf{y}}^{(t_{n_{p+1}}+m_p)}(0^{n_{p+1}}1^{n_{p+1}}) &= \tilde{\mathbf{y}}^{(t_{n_p}+m_p)}(0^{n_p}1^{n_p}) = \tilde{\mathbf{v}} = \tilde{\mathbf{y}}^{(t_{n_{p+1}}+m_{p+1})}(0^{n_{p+1}}1^{n_{p+1}}) \\ &= \tilde{\mathbf{y}}^{(t_{n_{p+2}}+m_{p+1})}(0^{n_{p+2}}1^{n_{p+2}}) \qquad (64)\end{aligned}$$

by (36) and (46).

For the induction step, assume

$$I_{p+1,k} \subset I_{p,k}\,, \qquad (65)$$

$$\tilde{\mathbf{y}}^{(t_{n_{p+1}}+m_p+k)}(0^{n_{p+1}}1^{n_{p+1}}) = \tilde{\mathbf{y}}^{(t_{n_{p+2}}+m_{p+1}+k)}(0^{n_{p+2}}1^{n_{p+2}}) \qquad (66)$$

for every $k = 0, \ldots, r-1$. According to (40) and (66), we obtain

$$\begin{aligned}b_{m_p+k} &= \beta \sum_{i \in V'} w_{1i} y_i^{(t_{n_{p+1}}+m_p+k-1)}(0^{n_{p+1}}1^{n_{p+1}}) \\ &= \beta \sum_{i \in V'} w_{1i} y_i^{(t_{n_{p+2}}+m_{p+1}+k-1)}(0^{n_{p+2}}1^{n_{p+2}}) = b_{m_{p+1}+k} \qquad (67)\end{aligned}$$

for $k = 1, \ldots, r$. Hence, the intervals $I_{p,r}$ and $I_{p+1,r}$ have the same left endpoint by definition (57) which ensures $I_{p+1,r} \subset I_{p,r}$ due to their right endpoints satisfy $c_{p+1} < c_p$, which completes the induction step for (60).

Furthermore, suppose on the contrary that

$$\tilde{\mathbf{y}}^{(t_{n_{p+1}}+m_p+r)}(0^{n_{p+1}}1^{n_{p+1}}) \neq \tilde{\mathbf{y}}^{(t_{n_{p+2}}+m_{p+1}+r)}(0^{n_{p+2}}1^{n_{p+2}})\,, \qquad (68)$$

which means there is $j_1 \in \{2, \ldots, s\}$ such that

$$y_{j_1}^{(t_{n_{p+1}}+m_p+r)}(0^{n_{p+1}}1^{n_{p+1}}) \neq y_{j_1}^{(t_{n_{p+2}}+m_{p+1}+r)}(0^{n_{p+2}}1^{n_{p+2}})\,. \qquad (69)$$

It follows that $w_{j_1,1} \neq 0$ since by (66) we know

$$\tilde{\mathbf{y}}^{(t_{n_{p+1}}+m_p+r-1)}(0^{n_{p+1}}1^{n_{p+1}}) = \tilde{\mathbf{y}}^{(t_{n_{p+2}}+m_{p+1}+r-1)}(0^{n_{p+2}}1^{n_{p+2}})\,. \qquad (70)$$

We define

$$c' = -\frac{1}{w_{j_1,1}} \sum_{i \in V'} w_{j_1,i}\, y_i^{(t_{n_{p+1}}+m_p+r-1)}(0^{n_{p+1}}1^{n_{p+1}})\,. \qquad (71)$$

We can distinguish four cases depending on the sign of $w_{j_1,1}$ and the binary state value $y_{j_1}^{(t_{n_{p+1}}+m_p+r)}(0^{n_{p+1}}1^{n_{p+1}}) \in \{0, 1\}$. For example, consider hereafter

the case when $w_{j_1,1} > 0$ and $y_{j_1}^{(t_{n_{p+1}}+m_p+r)}(0^{n_{p+1}}1^{n_{p+1}}) = 1$ which ensures $y_{j_1}^{(t_{n_{p+2}}+m_{p+1}+r)}(0^{n_{p+2}}1^{n_{p+2}}) = 0$ by (69), while the argument for the remaining cases is similar[14]. By the analogy to (47)–(49), we have

$$\xi_{j_1}^{(t_{n_{p+2}}+m_{p+1}+r)}(0^{n_{p+2}}1^{n_{p+2}}) < 0 \le \xi_{j_1}^{(t_{n_{p+1}}+m_p+r-1)}(0^{n_{p+1}}1^{n_{p+1}}) \qquad (72)$$

which reduces to

$$y_1^{(t_{n_{p+2}}+m_{p+1}+r-1)}(0^{n_{p+2}}1^{n_{p+2}}) < c' \le y_1^{(t_{n_{p+1}}+m_p+r-1)}(0^{n_{p+1}}1^{n_{p+1}})\,. \qquad (73)$$

according to (71) and (70).

Suppose that $c' \in I_{p+1,r-1} \subset I_{p,r-1}$. Since

$$\lim_{q\to\infty} y_1^{(t_{n_q}+m_{p+1}+r-1)}(0^{n_q}1^{n_{p+2}}) = \beta^{-(r-1)}c_{p+1} + \sum_{k=1}^{r-1} b_{m_{p+1}+r-k+1}\beta^{-k} \qquad (74)$$

by (53) and (59), which is the right endpoint of the interval $I_{p+1,r-1} \ni c'$, there exists $q > p+2$ such that $y_1^{(t_{n_q}+m_{p+1}+r-1)}(0^{n_q}1^{n_{p+2}}) \ge c'$. Hence, $0 = y_{j_1}^{(t_{n_{p+2}}+m_{p+1}+r)}(0^{n_{p+2}}1^{n_{p+2}}) \ne y_{j_1}^{(t_{n_q}+m_{p+1}+r)}(0^{n_q}1^{n_{p+2}}) = 1$ by (73), which contradicts (36). We conclude that $c' \notin I_{p+1,r-1}$, that is, $c' \notin I_{q,r-1} \subseteq I_{p+1,r-1}$ for every $q > p$, by (65), which ensures

$$y_{j_1}^{(t_{n_{q+1}}+m_q+r)}(0^{n_{q+1}}1^{n_{q+1}}) = y_{j_1}^{(t_{n_{q+2}}+m_{q+1}+r)}(0^{n_{q+2}}1^{n_{q+2}}) \text{ for every } q > p \quad (75)$$

according to (58) whereas (69) reduces to (73). Since by (71) there are only $(s-1)2^{s-1}$ possible values of c' for different $j_1 \in \{2,\dots,s\}$ and $\tilde{\mathbf{y}}^{(t_{n_{p+1}}+m_p+r-1)}(0^{n_{p+1}}1^{n_{p+1}}) \in \{0,1\}^{s-1}$, there exists p_0 such that for every $p \ge p_0$, any such value does not belong to $I_{p+1,r-1}$ by (54) and (65), which gives

$$\tilde{\mathbf{y}}^{(t_{n_{p+1}}+m_p+r)}(0^{n_{p+1}}1^{n_{p+1}}) = \tilde{\mathbf{y}}^{(t_{n_{p+2}}+m_{p+1}+r)}(0^{n_{p+2}}1^{n_{p+2}})\,. \qquad (76)$$

This completes the induction step for (61) and the proof of Lemma 2. □

[14]Note that for $w_{j_1,1} < 0$, inequality (73) reads $y_1^{(t_{n_{p+1}}+m_p+r-1)}(0^{n_{p+1}}1^{n_{p+1}}) \le c' < y_1^{(t_{n_{p+2}}+m_{p+1}+r-1)}(0^{n_{p+2}}1^{n_{p+2}})$, whereas $y_1^{(t_{n_{p+1}}+m_p+r-1)}(0^{n_{p+1}}1^{n_{p+1}})$ and $y_1^{(t_{n_{p+2}}+m_{p+1}+r-1)}(0^{n_{p+2}}1^{n_{p+2}})$ are swapped in these inequalities when $y_{j_1}^{(t_{n_{p+1}}+m_p+r)}(0^{n_{p+1}}1^{n_{p+1}}) = 0$.

3.7. The Periodicity Prevents from Separating Analog States

It follows from Lemma 2 that for all sufficiently large $p \geq p_0$,

$$b'_r = b_{m_p+r} = b_{m_{p+1}+r} \quad \text{for } r = 1, \ldots, \ell_p = \ell\,, \tag{77}$$

according to (40) and (61), which implies

$$c = B \sum_{q=1}^{\infty} \beta^{-\ell(q-1)} = \frac{B}{1-\beta^{-\ell}} \tag{78}$$

where $B = \sum_{r=1}^{\ell} b'_{\ell-r+1}\beta^{-r}$, due to (52). Hence,

$$\begin{aligned} & \beta^{m_p-m_{p_0}} \left(c - \sum_{k=1}^{m_p-m_{p_0}} b_{m_p-k+1}\beta^{-k} \right) \\ = \;& \beta^{\ell(p-p_0)} \left(B \sum_{q=1}^{\infty} \beta^{-\ell(q-1)} - B \sum_{q=p_0+1}^{p} \beta^{-\ell(p-q)} \right) = B \sum_{q=1}^{\infty} \beta^{-\ell(q-1)} = c\,, \end{aligned} \tag{79}$$

which ensures that the expression

$$\begin{aligned} & \beta^{m_p} \left(c - \sum_{k=1}^{m_p} b_{m_p-k+1}\,\beta^{-k} \right) \\ = \;& \beta^{m_{p_0}} \left(\beta^{m_p-m_{p_0}} \left(c - \sum_{k=1}^{m_p-m_{p_0}} b_{m_p-k+1}\beta^{-k} \right) - \sum_{k=1}^{m_{p_0}} b_{m_{p_0}-k+1}\beta^{-k} \right) \\ = \;& \beta^{m_{p_0}} \left(c - \sum_{k=1}^{m_{p_0}} b_{m_{p_0}-k+1}\beta^{-k} \right) = C \end{aligned} \tag{80}$$

is constant for every $p \geq p_0$. According to (51), we have

$$y_1^{(t_{n_p})}(0^{n_p}) < C \leq y_1^{(t_{n_{p+1}})}(0^{n_{p+1}}) \tag{81}$$

for every $p \geq p_0$, which is a contradiction. This completes the proof of Theorem 1. □

4. Two Analog Neurons Accept Deterministic Languages

We have proven in Section 3 that one extra analog unit in 1ANNs is not sufficient for recognizing the DCFL $L_{\#} = \{0^n 1^n \mid n \geq 1\}$. In this section, we show that any DPDA can be simulated by a 2ANN with rational weights. Thus, the DCFLs including $L_{\#}$ (see Example 2) are recognized by 2ANNs with two extra analog unit, which means DCFLs $\subset$ 2ANNs. This provides the separation 1ANNs $\subsetneq$ 2ANNs of the second level in the analog neuron hierarchy (see Figure 1).

The main idea of simulating DPDAs by 2ANNs is based on the classical technique of implementing the PDA's stack by two analog neurons, one for the `pop` operation and the other one for `push`, where the stack contents are encoded by analog states using a Cantor-like set (Siegelmann and Sontag, 1995). The technical part of the proof then consists in synchronizing the `swap` operation on the states of analog neurons.

We recall a formal definition of a *deterministic pushdown automaton (DPDA)* which is a septuple $\mathcal{M} = (Q, \Sigma, \Gamma, q_0, Z_0, F, \delta)$ where $Q \neq \emptyset$ is a finite set of states, Σ and Γ are finite sets of input and stack symbols, respectively, which are assumed for simplicity to be the binary alphabet $\Sigma = \Gamma = \{0, 1\}$. In addition, $q_0 \in Q$ is the start state, $Z_0 \in \Gamma$ is the initial stack symbol, and $F \subseteq Q$ is the set of accepting states. Moreover,

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma^*) \tag{82}$$

is a transition function that given a current state $q \in Q$ of $\mathcal{M}$, a next symbol $x \in \Sigma \cup \{\varepsilon\}$ of an input word which is read from left to right (including the empty string ε, which means no symbol is read), and a symbol $Z \in \Gamma$ on the top of the stack, produces either the empty set $\delta(q, x, Z) = \emptyset$ (i.e. $\mathcal{M}$ halts), or a one-element set $\delta(q, x, Z) = \{(q', \gamma)\}$ with a new state $q' \in Q$ and a string $\gamma \in \Gamma^*$ that replaces Z on the top of the stack where the first symbol of γ becomes the top element. In order to ensure that $\mathcal{M}$ is truly deterministic, it is assumed that for any $q \in Q$, $Z \in \Gamma$, if $\delta(q, \varepsilon, Z) \neq \emptyset$, then $\delta(q, x, Z) = \emptyset$ for every $x \in \Sigma$.

An input word $\mathbf{x} \in \Sigma^*$ is accepted by $\mathcal{M}$ if there is a (unique) sequence of transitions of $\mathcal{M}$ defined by δ, from the start state q_0 with the initial symbol Z_0 on the stack, which, by reading $\mathbf{x}$, terminates in an accepting state from F. We say that a language $L \subseteq \Sigma^*$ is accepted by a DPDA $\mathcal{M}$, which is denoted as $L = \mathcal{L}(\mathcal{M})$, if for any input $\mathbf{x} \in \Sigma^*$, $\mathcal{M}$ accepts $\mathbf{x}$ iff $\mathbf{x} \in L$. The class of languages accepted by DPDAs establishes the class of DCFLs.

In addition, we assume without loss of generality that if $\delta(q, x, Z) = \{(q', \gamma)\}$, then the length $|\gamma|$ of string γ is at most 2 and $\gamma = Z'Z$ for some $Z' \in \Gamma$, if $|\gamma| = 2$. Note that there are only finitely many instructions in $\mathcal{M}$ because the domain $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ of δ in (82) is finite. Thus, any instruction with $|\gamma| > 2$ can always be replaced by a unique sequence of new instructions (for which new states are introduced in Q) that push the string γ successively symbol by symbol to the stack.

Theorem 2 *For any deterministic context-free language $L \subseteq \{0,1\}^*$, there is a 2ANN $\mathcal{N}$ with two extra analog units having rational weights, which accepts $L = \mathcal{L}(\mathcal{N})$.*

Proof. Let $L = \mathcal{L}(\mathcal{M})$ be accepted by a DPDA $\mathcal{M} = (Q, \Sigma, \Gamma, q_0, Z_0, F, \delta)$. We will construct a 2ANN $\mathcal{N}$ with two extra analog units that accepts the same language $L = \mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{M})$ by simulating the DPDA $\mathcal{M}$. A scheme of the architecture of $\mathcal{N}$ is depicted in Figure 3 where the directed edges connecting neurons are labeled with the respective weights, while the edges drawn without the originating unit $0 \in V'$ correspond to the biases.

The stack of $\mathcal{M}$ is realized by the two analog neurons $1, 2 \in V$ of $\mathcal{N}$, where the first unit implements the `top` and `push` operations while the `pop` operation is performed by the second analog neuron. The current contents of the stack, $Z_1 \dots Z_p \in \Gamma^p = \{0,1\}^p$ are encoded by the state of an analog neuron,

$$y_k^{\rm cur} = \sum_{i=1}^{p} \frac{2Z_i + 1}{4^i} \in \mathbb{I} \qquad \text{for } k \in \{1, 2\}\,, \tag{83}$$

using a Cantor-like set which allows an efficient neuronal implementation of the stack operations (Siegelmann and Sontag, 1995), producing the new states $y_1^{\rm new}$, $y_2^{\rm new}$ of analog neurons:

$$\texttt{top} = H(2y_1^{\rm cur} - 1) \in \{0, 1\} \tag{84}$$

$$\texttt{push}(Z): \quad y_1^{\rm new} = \sigma\left(\frac{1}{4}y_1^{\rm cur} + \frac{1}{2}Z + \frac{1}{4}\right) \in \mathbb{I} \tag{85}$$

$$\texttt{pop}: \quad y_2^{\rm new} = \sigma(4y_2^{\rm cur} - 2\texttt{top} - 1) \in \mathbb{I} \tag{86}$$

where the activation functions (5) and (4) are used.

The finite control of $\mathcal{M}$ which carries out the state transitions defined by the transition function δ, is implemented by binary neurons. We will describe its functionality while the omitted technical details are ensured using

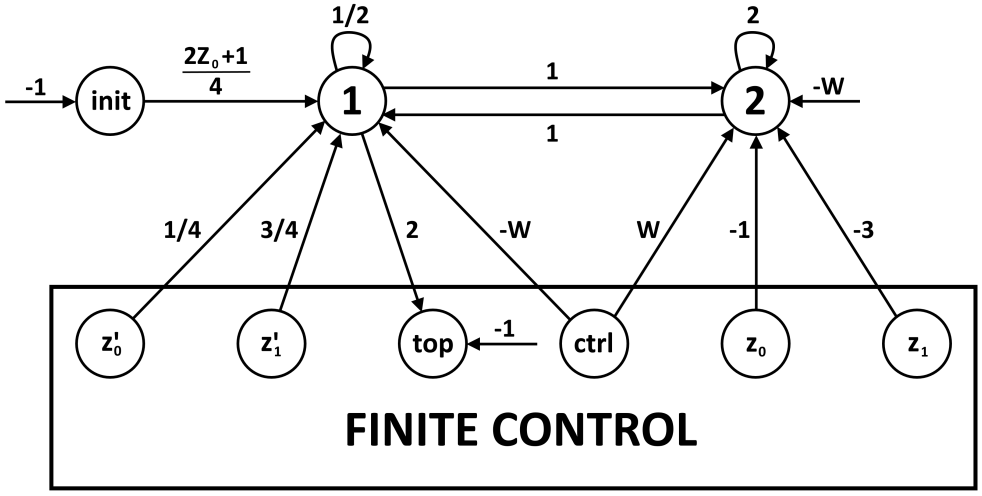

Figure 3: A schema of 2ANNs simulating DPDAs.

known techniques of implementing finite automata by neural networks with integer weights (Horne and Hush, 1996; Indyk, 1995; Minsky, 1967; Šíma and Wiedermann, 1998). At the beginning of the simulation of $\mathcal{M}$ by $\mathcal{N}$, the stack is initialized by the initial stack symbol Z_0. This is implemented by a special binary neuron init $\in V'$ which is only active initially, that is, $y_{\text{init}}^{(t)} = 1$ iff $t = 0$. Thus, init with the bias $w_{\text{init},0} = -1$ is connected to the first analog neuron $1 \in V$ via the weight

$$w_{1,\text{init}} = \frac{2Z_0 + 1}{4}\,, \tag{87}$$

which encodes the stack contents Z_0 by analog state $y_1^{(1)} = (2Z_0 + 1)/4$, according to (83). Then, each transition of $\mathcal{M}$ is realized by one so-called *macrostep* $\tau \geq 1$ which is composed of 12 computational steps of $\mathcal{N}$, starting at the discrete global time instant $t = 12(\tau - 1) + 1$ (including one more step for the stack initialization at $t = 1$). Hereafter, for simplicity, we use the local computational time $t = 0, 1, 2, \ldots, 12$ of $\mathcal{N}$ which is related to the macrostep. The state evolution of relevant neurons during the macrostep is presented in Table 2.

At the beginning of the macrostep when $t = 0$, the state of the first analog neuron $1 \in V$ encodes the current contents of the stack, that is, $y_1^{(0)} = z \in \mathbb{I}$

by (83). The storage of the stack contents alternates between the two analog neurons $1, 2 \in V$ which are connected by the weights

$$w_{12} = w_{21} = 1\,. \tag{88}$$

These unit weights copy the state from the first analog neuron to the second one and back, under the control of binary neuron ctrl $\in V'$. During the macrostep, the output of ctrl produces a sequence of binary states given by the regular expression

$$1(01)^3(110 + 010)(001 + 101) \tag{89}$$

starting with $y^{(0)}_{\rm ctrl} = 1$, where the substrings 110 and 001 deviating from the regular signal $(01)^*$ correspond to the `pop` and `push` operations, respectively, if they occur, as described below. For this purpose, the weights

$$w_{1,\rm ctrl} = -W\,, \qquad w_{2,\rm ctrl} = W\,, \qquad w_{20} = -W \tag{90}$$

are introduced, where $W > 0$ is a sufficiently large positive parameter excluding the influence from other neurons. It follows from (3), (4), and (88)–(90) that

$$z \;=\; y^{(0)}_1 = y^{(1)}_2 = y^{(2)}_1 = y^{(3)}_2 = y^{(4)}_1 = \cdots \tag{91}$$

$$0 \;=\; y^{(0)}_2 = y^{(1)}_1 = y^{(2)}_2 = y^{(3)}_1 = y^{(4)}_2 = \cdots\,, \tag{92}$$

as shown in Table 2.

At the time instant $t = 1$ of the macrostep, the binary neuron top $\in V'$ reads the top element $Z \in \Gamma$ from the stack, that is, $y^{(1)}_{\tt top} = Z \in \{0, 1\}$, which is implemented by the weights

$$w_{\tt top,1} = 2, \qquad w_{\tt top,0} = -1 \tag{93}$$

according to (84). At the time instant $t = 2$, the network $\mathcal{N}$ finds out for this top symbol Z whether $\delta(q, \varepsilon, Z) = \emptyset$, where $q \in Q$ is a current state of $\mathcal{M}$ encoded by some binary neurons in the subnetwork of $\mathcal{N}$ implementing the finite control of $\mathcal{M}$. If $\delta(q, \varepsilon, Z) = \emptyset$, then the input/output protocol (6) and (7) is employed at the time instant $t = 3$, which means $\mathcal{N}$ signals $y^{(3)}_{\rm nxt} = 1$, and $y^{(3)}_{\rm out} = 1$ iff $q \in F$ is an accepting state, while reading the next input symbol $y^{(4)}_{\rm inp} = x \in \{0, 1\}$ at the time instant $t = 4$. Anyway,

t	$y_1^{(t)}$	$y_2^{(t)}$	$y_{\rm ctrl}^{(t)}$	$y_{z_0}^{(t)}$	$y_{z_1}^{(t)}$	$y_{z'_0}^{(t)}$	$y_{z'_1}^{(t)}$	$y_{\texttt{top}}^{(t)}$	$y_{\rm nxt}^{(t)}$	$y_{\rm out}^{(t)}$	$y_{\rm inp}^{(t)}$	operation
0	z	0	1	0	0	0	0		0	0	0	
1	0	z	0	0	0	0	0	Z	0	0	0	
2	z	0	1	0	0	0	0		0	0	0	
3	0	z	0	0	0	0	0		1	$q \in F$	0	`top`
4	z	0	1	0	0	0	0		0	0	x	
5	0	z	0	0	0	0	0		0	0	0	
6	z	0	1	0	0	0	0		0	0	0	
7	0	z	1	0	0	0	0		0	0	0	
8	0	$2z$	1	$Z=0$	$Z=1$	0	0		0	0	0	`pop`
9	0	$z'=4z-2Z-1$	0	0	0	0	0		0	0	0	
10	z'	0	0	0	0	0	0		0	0	0	
11	$\frac{z'}{2}$	0	0	0	0	$Z'=0$	$Z'=1$		0	0	0	`push`
$12 \equiv 0$	$z''=\frac{z'}{4}+\frac{Z'}{2}+\frac{1}{4}$	0	1	0	0	0	0		0	0	0	

Table 2: The macrostep of 2ANN $\mathcal{N}$ simulating one transition of DPDA $\mathcal{M}$ (including the `pop` and `push` operations).

the next two steps $t = 5, 6$ of the macrostep are exploited for evaluating the transition function $\delta(q, x, Z)$ where $x = \varepsilon$ is the empty word if $\delta(q, \varepsilon, Z) \neq \emptyset$. If $\delta(q, x, Z) = \emptyset$, then the simulation by $\mathcal{N}$ terminates (e.g. all the neurons in the network are reset by a large negative weight) since the computation of $\mathcal{M}$ halts.

Thus, assume $\delta(q, x, Z) = \{(q', \gamma)\}$ where $q' \in Q$ is the new state of $\mathcal{M}$, for which the old one encoded by the binary neurons in $\mathcal{N}$, is substituted, and $\gamma \in \Gamma^*$ replaces the top symbol on the stack. If $|\gamma| \leq 1$, then the top symbol Z is popped from the stack during the time instants $t = 7, 8, 9$ of the macrostep. Namely, at the time instant $t = 7$, the current contents of the stack are stored by the second analog neuron as $y_2^{(7)} = z$. The `pop` operation is implemented by the weights

$$w_{22} = 2 \tag{94}$$

$$w_{2,z_0} = -1\,, \qquad w_{2,z_1} = -3 \tag{95}$$

from the binary neurons $z_0, z_1 \in V'$ whose outputs are activated at the time instant $t = 8$ of the macrostep with respect to the top stack element Z so that

$$y_{z_b}^{(8)} = 1 \quad \text{iff} \quad Z = b \in \{0, 1\}\,. \tag{96}$$

Moreover, we know $y_{\rm ctrl}^{(7)} = y_{\rm ctrl}^{(8)} = 1$ and $y_{\rm ctrl}^{(9)} = 0$ by (89) when the `pop`

operation applies (otherwise, $y^{(7)}_{\rm ctrl} = 0$). Hence, $y^{(8)}_2 = 2z$ by (94), and

$$y^{(9)}_2 = 4z - 2\texttt{top} - 1 = z' \in \mathbb{I} \tag{97}$$

due to (94)–(96), which pops the top symbol $Z = \texttt{top}$ from the stack according to (86).

If $|\gamma| \geq 1$, then either $\gamma = Z'$ or $\gamma = Z'Z$, where $Z' \in \Gamma$ is the new top symbol which is pushed to the stack during the time instants $t = 10, 11, 12$ of the macrostep. Namely, at the time instant $t = 10$, the current contents of the stack are stored by the first analog neuron as $y^{(10)}_1 = z'$. The $\texttt{push}(Z')$ operation is implemented by the weights

$$w_{11} = \tfrac{1}{2} \tag{98}$$

$$w_{1,z'_0} = \tfrac{1}{4}\,, \qquad w_{1,z'_1} = \tfrac{3}{4} \tag{99}$$

from the binary neurons $z'_0, z'_1 \in V'$ whose outputs are activated at the time instant $t = 11$ of the macrostep with respect to the new top stack element Z' so that

$$y^{(11)}_{z'_b} = 1 \quad \text{iff} \quad Z' = b \in \{0, 1\}\,. \tag{100}$$

Moreover, we know $y^{(10)}_{\rm ctrl} = y^{(11)}_{\rm ctrl} = 0$ and $y^{(12)}_{\rm ctrl} = 1$ by (89) when the $\texttt{push}$ operation applies (otherwise, $y^{(10)}_{\rm ctrl} = 1$). Hence, $y^{(11)}_1 = \frac{z'}{2}$ by (98), and

$$y^{(12)}_2 = \tfrac{z'}{4} + \tfrac{Z'}{2} + \tfrac{1}{4} = z'' \in \mathbb{I} \tag{101}$$

due to (98)–(100), which pushes the symbol Z' to the stack according to (85).

At the time instant $t = 12$, the macrostep of $\mathcal{N}$ simulating one transition of $\mathcal{M}$ using rational weights is finished while the new contents z'' of the stack are stored by the first analog neuron as required for the next macrostep. This completes the simulation and the proof of Theorem 2. □

Example 2 We illustrate Theorem 2 on an example of a 2ANN $\mathcal{N}_\#$ that accepts the language $\mathcal{L}(\mathcal{N}_\#) = L_\# = \{0^n 1^n \,|\, n \geq 1\}$, which cannot be recognized by any 1ANN according to Theorem 1. For this purpose, we first shortly introduce a simple DPDA $\mathcal{M}_\# = (Q, \Sigma, \Gamma, q_0, Z_0, F, \delta)$ that accepts $\mathcal{L}(\mathcal{M}_\#) = L_\#$, where $Q = \{q_0, q_1, q_2\}$, $\Sigma = \Gamma = \{0, 1\}$, $Z_0 = 1$, $F = \{q_2\}$, while the values of δ not equal to the empty set, are defined as

$$\delta(q_0, 0, 1) = \{(q_0, 01)\}, \quad \delta(q_0, 0, 0) = \{(q_0, 00)\}, \quad \delta(q_0, 1, 0) = \{(q_1, \varepsilon)\}\,, \tag{102}$$

$$\delta(q_1, 1, 0) = \{(q_1, \varepsilon)\}, \quad \delta(q_1, \varepsilon, 1) = \{(q_2, \varepsilon)\}. \tag{103}$$

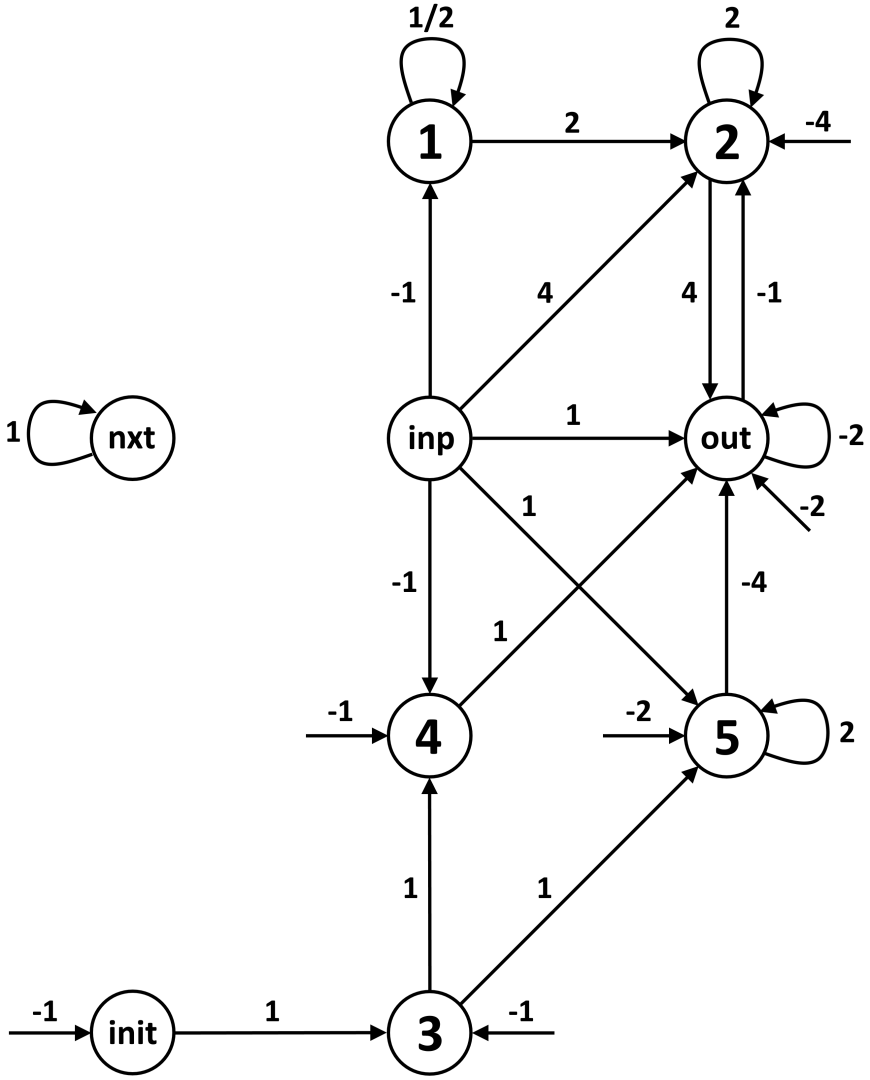

Figure 4: Example of a 2ANN $\mathcal{N}_{\#}$ recognizing $L_{\#} = \{0^n 1^n \,|\, n \geq 1\}$.

The DPDA $\mathcal{M}_{\#}$ starts at the state q_0 with the initial stack symbol $Z_0 = 1$ which serves for detecting the empty stack at the end of its computation. At the start state q_0, $\mathcal{M}_{\#}$ pushes the symbol 0 to the stack as long as it reads the input symbol 0, while $\mathcal{M}_{\#}$ goes to the state q_1 when the symbol 1 first occurs at the input, according to (102). At the state q_1, $\mathcal{M}_{\#}$ pops one

symbol 0 from the stack for each input symbol 1 that is read, by (103). Finally, $\mathcal{M}_{\#}$ accepts the input in the accepting state q_2 iff the stack contains only the initial symbol $Z_0 = 1$ after the input has been read. Clearly, only the input words composed of n zeros followed by the same number n of ones, are accepted by $\mathcal{M}_{\#}$, which means $\mathcal{L}(\mathcal{M}_{\#}) = L_{\#}$.

The general construction of a 2ANN $\mathcal{N}_{\#}$ corresponding to the DPDA $\mathcal{M}_{\#}$ as it is described in the proof of Theorem 2, leads to an unnecessarily complicated architecture of $\mathcal{N}_{\#}$, which would not be too illustrative. Fortunately, this construction can substantially be simplified in the particular case of $L_{\#}$. Note that $\mathcal{M}_{\#}$ employs first a sequence of the `push` operations and only then the `pop` operations are applied. Thus, there is no need for swapping the storage of the stack between the two analog neurons such as (91) and (92). In addition, $\mathcal{M}_{\#}$ exploits, in fact, only one stack symbol 0 whereas the initial stack symbol 1 is used solely for detecting the empty stack which can directly be implemented in $\mathcal{N}_{\#}$. Hence, the unary stack contents 0^p can be encoded only by the negative powers 2^{-p-1} which simplifies the original encoding (83).

The architecture of such a simplified $\mathcal{N}_{\#}$ is depicted in Figure 4. We assume that $\mathcal{N}_{\#}$ starts with the zero initial states except for $y_1^{(0)} = y_{\mathrm{nxt}}^{(0)} = y_{\mathrm{init}}^{(0)} = 1$. Note that the state of binary neuron $\mathrm{nxt} \in V'$ then meets $y_{\mathrm{nxt}}^{(0)} = 1$ for every $t \geq 0$, because of its positive self-loop weight $w_{\mathrm{nxt,nxt}} = 1$. According to the input/output protocol (6) and (7), an input word $\mathbf{x} = x_1 \dots x_n \in \{0,1\}^n$ is thus presented online, bit by bit, at each time instant $\tau_k = k$ for $k = 1 \dots, n$, which means $y_{\mathrm{inp}}^{(0)} = 0$ and $y_{\mathrm{inp}}^{(t)} = x_t$ for $t = 1, \dots, n$, whereas $y_{\mathrm{out}}^{(t+1)} = 1$ iff $x_1 \dots x_t \in L_{\#}$, particularly, $y_{\mathrm{out}}^{(n+1)} = 1$ iff $\mathbf{x} \in L_{\#}$. The four binary neurons $\mathrm{init}, 3, 4, 5 \in V'$ in $\mathcal{N}_{\#}$ serve for processing the rejected inputs that start with 1 and the accepted single input 01. This part of $\mathcal{N}_{\#}$ which is discussed below, illustrates techniques of the finite control design whose details have been omitted in the proof of Theorem 2.

Thus, we first consider the case when an input starts with the prefix 00. For example, Table 3 shows the state evolution of relevant neurons for the input $\mathbf{x} = 0^p 1^p \in L_{\#}$ with $p = 4$, where the input bits and the resulting output are indicated in boldface. At the time instant $t = 1$, the initially active analog neuron $1 \in V$ divides its state by two as $y_1^{(1)} = y_1^{(0)}/2 = \frac{1}{2}$ because of its self-loop weight $w_{11} = \frac{1}{2}$ and $y_{\mathrm{inp}}^{(0)} = 0$. This is further repeated at the time instants $t = 2, 3, \dots, p+1$ when the zero bits from the prefix 0^p of input $\mathbf{x}$ are processed. This simulates pushing the p input bits 0 to

t	$y_{\text{inp}}^{(t)}$	$y_1^{(t)}$	$y_2^{(t)}$	$y_{\text{out}}^{(t)}$
0	0	1	0	0
1	**0**	$\frac{1}{2}$	0	0
2	**0**	$\frac{1}{4}$	0	0
3	**0**	$\frac{1}{8}$	0	0
4	**0**	$\frac{1}{16}$	0	0
5	**1**	$\frac{1}{32}$	0	0
6	**1**	0	$\frac{1}{16}$	0
7	**1**	0	$\frac{1}{8}$	0
8	**1**	0	$\frac{1}{4}$	0
9	0	0	$\frac{1}{2}$	**1**
10	0	0	0	0

Table 3: The accepting computation by the 2ANN $\mathcal{N}_{\#}$ from Figure 4 on the input 00001111.

the stack which results in the analog state $y_1^{(p+1)} = \frac{1}{2^{p+1}}$ encoding the stack contents 0^p. At the time instant $p+2$ when the first input bit 1 is processed, this analog value is multiplied by $w_{21} = 2$ and moved to the second analog neuron $2 \in V$ since the weight $w_{2,\text{inp}} = 4$ cancels its bias $w_{20} = -4$ when $y_{\text{inp}}^{(p+1)} = 1$, that is, $y_1^{(p+2)} = \frac{1}{2^p}$, which simulates popping one 0 from the top of the stack. Note that at the same time, the state of the first analog neuron resets to $y_1^{(p+2)} = 0$ by the negative weight $w_{1,\text{inp}} = -1$, which is further clamped because of its only positive self-loop weight.

The underlying popping is then repeated for each bit 1 of the input suffix 1^p at the time instants $t = p+2, \ldots, 2p$ until the last bit 1 is presented to $\mathcal{N}_{\#}$, when $y_2^{(2p)} = \frac{1}{4}$ encodes the stack contents 0. At the time instant $t = 2p+1$ when the last input bit 1 is processed, the output neuron fires $y_{\text{out}}^{(2p+1)} = 1$ iff $y_1^{(2p)} = 1$ and $y_2^{(2p)} \geq \frac{1}{4}$ due to its weights $w_{\text{out,inp}} = 1$, $w_{\text{out},2} = 4$, and bias $w_{\text{out},0} = -2$, which means iff $\mathcal{N}_{\#}$ reads the last input bit 1 and the stack contains the last symbol 0. This provides the correct result of recognition, accepting the input $\mathbf{x} \in L_{\#}$. In addition, at the next time instant $t = 2p+2$, the active output neuron resets itself and the second

t	$y_{\mathrm{inp}}^{(t)}$	$y_1^{(t)}$	$y_2^{(t)}$	$y_{\mathrm{init}}^{(t)}$	$y_3^{(t)}$	$y_4^{(t)}$	$y_5^{(t)}$	$y_{\mathrm{out}}^{(t)}$
0	0	1	0	1	0	0	0	0
1	$\mathbf{1}$	$\frac{1}{2}$	0	0	1	0	0	0
2	$\mathbf{x_2}$	0	1	0	0	0	1	0
3	$\mathbf{x_3}$	0	x_2	0	0	0	1	0

Table 4: The rejecting computation by the 2ANN $\mathcal{N}_{\#}$ from Figure 4 on an word input starting with 1.

analog neuron as $y_{\mathrm{out}}^{(2p+2)} = y_2^{(2p+2)} = 0$ via the negative weights $w_{\mathrm{out,out}} = -2$ and $w_{2,\mathrm{out}} = -1$, respectively, which stops the simulation of $\mathcal{M}_{\#}$.

Furthermore, Table 4 shows the state evolution of relevant neurons in $\mathcal{N}_{\#}$ for an input word $\mathbf{x} = 1x_2x_3 \ldots x_n \in \{0,1\}^n$ that starts with 1. At the time instant $t = 1$, we have $y_1^{(1)} = \frac{1}{2}$, and the initially active neuron $\mathrm{init} \in V$, which itself resets to $y_{\mathrm{init}}^{(1)} = 0$ by its negative bias $w_{\mathrm{init},0} = -1$, ensures $y_3^{(1)} = 1$ via the weight $w_{3,\mathrm{init}} = 1$ balancing the bias $w_{30} = -1$. At the time instant $t = 2$, the neuron $5 \in V$ fires iff $y_3^{(1)} = 1$ and $y_{\mathrm{inp}}^{(1)} = 1$ because of the weights $w_{53} = w_{5,\mathrm{inp}} = 1$ and bias $w_{50} = -2$, which detects that $\mathbf{x}$ starts with 1. If this is the case, then its state is clamped by the positive self-loop weight $w_{55} = 2$, that is, $y_5^{(t)} = 1$ for every $t \geq 2$, which prevents the output neuron $\mathrm{out} \in V$ from being activated due to the negative weight $w_{\mathrm{out},5} = -4$. In particular, the excitation

$$\begin{aligned} \xi_{\mathrm{out}}^{(2)} &= w_{\mathrm{out},0} + w_{\mathrm{out,inp}} x_2 + w_{\mathrm{out},2} y_2^{(2)} + w_{\mathrm{out},5} y_5^{(2)} \\ &= -2 + 1 \cdot x_2 + 4 \cdot 1 - 4 \cdot 1 < 0\,, \end{aligned} \tag{104}$$

by (2), which implies $y_{\mathrm{out}}^{(2)} = 0$ according to (5). Hence, the input $1x_2x_3 \ldots x_n \notin L_{\#}$ is correctly rejected.

Finally, we check that the single input $01 \in L_{\#}$ is accepted by $\mathcal{N}_{\#}$ whose computation is outlined in Table 5. As in the previous case described in Table 4, we have $y_3^{(1)} = 1$ which now produces $y_4^{(2)} = 1$ via the weight $w_{43} = 1$ balancing its negative bias $w_{40} = -1$ since $y_{\mathrm{inp}}^{(1)} = 0$ cancels the negative weight $w_{4,\mathrm{inp}} = -1$. Hence, $y_{\mathrm{out}}^{(3)} = 1$ because $y_{\mathrm{inp}}^{(2)} = 1$, $w_{\mathrm{out},4} = w_{\mathrm{out,inp}} = 1$, and $w_{\mathrm{out},0} = -2$. Thus, the input $01 \in L_{\#}$ is accepted by $\mathcal{N}_{\#}$. Moreover, $y_{\mathrm{out}}^{(3)} = 1$ resets $\mathcal{N}_{\#}$ for time $t \geq 4$ as in the case presented in Table 3. This completes Example 2.

t	$y_{\text{inp}}^{(t)}$	$y_1^{(t)}$	$y_2^{(t)}$	$y_{\text{init}}^{(t)}$	$y_3^{(t)}$	$y_4^{(t)}$	$y_5^{(t)}$	$y_{\text{out}}^{(t)}$
0	0	1	0	1	0	0	0	0
1	**0**	$\frac{1}{2}$	0	0	1	0	0	0
2	**1**	$\frac{1}{4}$	0	0	0	1	0	0
3	0	0	$\frac{1}{2}$	0	0	0	0	**1**

Table 5: The accepting computation by the 2ANN $\mathcal{N}_{\#}$ from Figure 4 on the input 01.

5. Simulating a Turing Machine Using Three Analog Neurons

In this section, we prove that any TM can be simulated by a 3ANN having rational weights with a linear-time overhead. This means that recursively enumerable languages are accepted by 3ANNs with rational weights. In other words, this model including only three analog neurons is Turing-complete, being able to compute any algorithmically computable function. On the other hand, αANNs with rational weights can in principle be simulated by TMs for any $\alpha \geq 0$, which implies the collapse of the analog neuron hierarchy at the third level: αANNs = 3ANNs = TMs for every $\alpha \geq 3$ (see Figure 1).

The main idea of simulating TMs by 3ANNs is the same as in Section 4, which is based on the technique of implementing the PDA's stack by two analog neurons, one for the `pop` operation and the other one for `push`, where the stack contents are encoded by analog states using a Cantor-like set (Siegelmann and Sontag, 1995). Since two stacks are known to be sufficient for simulating TMs, the technical part of the proof reduces to synchronizing the `swap` operation on the states of analog neurons, which employs the third auxiliary analog neuron.

We recall a formal definition of a *Turing machine (TM)* which is a quintuple $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ where $Q \neq \emptyset$ is a finite set of states and $\Sigma \neq \emptyset$ is a finite set of tape alphabet symbols including the blank symbol $B \in \Sigma$, while $\Sigma \setminus \{B\} \neq \emptyset$ serves as an input alphabet. In addition, $q_0 \in Q$ is the initial state and $F \subseteq Q$ is the set of final or accepting states. Apart from the finite control unit which stores the current state, $\mathcal{M}$ has a tape which is arbitrarily extendable to the left and to the right. We assume without loss of generality that the tape and input alphabets coincide in the binary alphabet $\Sigma = \{0, 1\}$ which is sufficient for encoding the blank symbol B uniquely (e.g. each symbol is encoded by two bits) so that there is the infinite string 0^ω to

the left and to the right of the tape.

At startup, $\mathcal{M}$ begins in the initial state q_0 with an input word $\mathbf{x} = x_1 \dots x_n \in \{0,1\}^n$ written on the tape so that x_1 is under the tape head. Furthermore,

$$\delta : (Q \setminus F) \times \Sigma \longrightarrow Q \times \Sigma \times \{L, R\} \tag{105}$$

is a partial function called the transition function of $\mathcal{M}$ that given its current state $q_{\text{cur}} \in Q$ and a symbol $x \in \Sigma$ under the head, produces

$$\delta(q_{\text{cur}}, x) = (q_{\text{new}}, b, d) \tag{106}$$

(if defined for q_{cur} and x) where $q_{\text{new}} \in Q$ is its new state, $b \in \Sigma$ is a symbol to overwrite x on the tape, and a direction $d \in \{L, R\}$ for the tape head to move, which is either left shift for $d = L$ or right shift for $d = R$. If δ is not defined on the current state and the tape symbol under the head, then $\mathcal{M}$ halts. Finally, the input word $\mathbf{x}$ is accepted if there is a sequence of transitions of $\mathcal{M}$ defined by δ, which terminates in an accepting state from F. We say that a language $L \subseteq \Sigma^*$ is accepted by a TM $\mathcal{M}$, which is denoted as $L = \mathcal{L}(\mathcal{M})$, if for any input $\mathbf{x} \in \Sigma^*$, $\mathcal{M}$ accepts $\mathbf{x}$ iff $\mathbf{x} \in L$. The class of languages accepted by TMs establishes the class of recursively enumerable languages.

The following theorem shows how to simulate a TM by a 3ANN with rational weights and a linear-time overhead.

Theorem 3 *Given a Turing machine $\mathcal{M}$ that accepts a language $L = \mathcal{L}(\mathcal{M})$ in time $T(n) \geq n$, there is a 3ANN $\mathcal{N}$ with rational weights, which accepts the same language $L = \mathcal{L}(\mathcal{N})$ in time $O(T(n))$.*

Proof. Let $L = \mathcal{L}(\mathcal{M})$ be accepted by a TM $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ in time $T(n) \geq n$. We will construct a 3ANN $\mathcal{N}$ with the set of neurons V, simulating the Turing machine $\mathcal{M}$ so that $L = \mathcal{L}(\mathcal{N}) = \mathcal{L}(\mathcal{M})$. For this purpose, we use two stacks s_1 and s_2. One stack holds the contents of the tape to the left of the head of $\mathcal{M}$ while the other stack stores the right part of the tape. We assume that the first stack s_1 implements only the `push`(b) operation adding a given element b to the top of s_1, whereas the second stack s_2 allows only for the `top` and `pop` operation which reads and removes the top element of s_2, respectively. In addition, the top element of s_2 models a symbol currently under the head of $\mathcal{M}$. In order to compensate for these restriction, we introduce the `swap` operation which exchanges the contents of s_1 and s_2,

while the finite control unit remembers in its current state which part of the tape contents $c_{\rm cur} \in \{L, R\}$, either to the left of the head for $c_{\rm cur} = L$ or to the right for $c_{\rm cur} = R$, is stored in the second stack s_2.

We show how to implement one instruction $\delta(q_{\rm cur}, x) = (q_{\rm new}, b, d)$ of $\mathcal{M}$, introduced in (106), by using the two stacks s_1 and s_2 and their operations $\texttt{push}(b)$, $\texttt{top}$, $\texttt{pop}$, and $\texttt{swap}$. The transition from its current state $q_{\rm cur}$ to the new state $q_{\rm new}$ is realized by the finite control unit, while the tape update takes place in the stacks, for which we distinguish two cases, a so-called *short* and *long* instruction.

The short instruction applies when $d = c_{\rm cur}$. In this case, the two operations

$$\texttt{push}(b);\ \texttt{pop} \tag{107}$$

implement the corresponding update of the tape contents so that x under the head of $\mathcal{M}$ is overwritten by b, the head moves to a new symbol which is next in the desired direction d and appears at the top of s_2, while $c_{\rm new} = c_{\rm cur}$ is preserved. For the long instruction when $d \neq c_{\rm cur}$, the following sequence of five operations

$$\texttt{push}(\texttt{top});\ \texttt{pop};\ \texttt{swap};\ \texttt{push}(b);\ \texttt{pop} \tag{108}$$

is employed where the first two operations $\texttt{push}(\texttt{top})$; $\texttt{pop}$ shift the current symbol $x = \texttt{top}$ under the head of $\mathcal{M}$ from the top of s_2 to the top of s_1. Then the $\texttt{swap}$ operation exchanges the contents of s_1 and s_2 so that x is back at the top of s_2. Now, $c_{\rm new} = d \neq c_{\rm cur}$, which ensures the conversion to the previous case, and hence, the last two operations of (108) coincide with the short instruction (107).

The stacks s_1 and s_2 are implemented by the first two analog neurons $1, 2 \in V \setminus V'$, respectively. The contents $\mathbf{a} = a_1 \ldots a_p \in \{0, 1\}^*$ of stack s_k for $k \in \{1, 2\}$, where a_1 is the top element of s_k, are represented by the analog state y_k of neuron $k \in V$, using the encoding $\gamma : \{0, 1\}^* \longrightarrow \mathbb{I}$,

$$y_k = \gamma\,(a_1 \ldots a_p) = \sum_{i=1}^{p} \frac{2^6\,(a_i + 1) - 1}{(2^7)^{i+1}} \in \left[0\,,\,\tfrac{1}{2^7}\right) \subset \mathbb{I}\,. \tag{109}$$

All the possible analog state values generated by the encoding (109) create a Cantor-like set so that two strings with distinct top symbols are represented by two sufficiently separated numbers (Siegelmann and Sontag, 1995). In

particular, for $\mathbf{a} \neq \varepsilon$, we have

$$a_1 = \begin{cases} 0 & \text{if } \gamma(a_1 \ldots a_p) \in \left[\frac{2^6-1}{2^{14}}, \frac{1}{2^8}\right) \\ \\ 1 & \text{if } \gamma(a_1 \ldots a_p) \in \left[\frac{2^7-1}{2^{14}}, \frac{1}{2^7}\right), \end{cases} \tag{110}$$

which can be used for reading the top element from the stack s_2 (i.e. the current tape symbol under the head of $\mathcal{M}$) by a binary neuron employing the Heaviside activation function (5):

$$\texttt{top} = 1 \quad \text{iff} \quad -1 + 2^8 y_2 \geq 0\,. \tag{111}$$

Furthermore, the $\texttt{push}(b)$ and $\texttt{pop}$ operation can be implemented by the analog neuron $1, 2 \in V$, respectively, employing the linear part of the saturated-linear activation function (4), as

$$\begin{aligned} \texttt{push}(b) \;:\; y_1^{\text{new}} = \xi_1^{\text{cur}} &= \frac{2^6(b+1)-1}{2^{14}} + \frac{1}{2^7} \cdot y_1^{\text{cur}} \\ &= \frac{2^6-1}{2^{14}} + \frac{1}{2^8} \cdot b + \frac{1}{2^7} \cdot y_1^{\text{cur}} \in \left[0, \tfrac{1}{2^7}\right) \end{aligned} \tag{112}$$

$$\begin{aligned} \texttt{pop} \;:\; y_2^{\text{new}} = \xi_2^{\text{cur}} &= \frac{1-2^6(\texttt{top}+1)}{2^7} + 2^7 \cdot y_2^{\text{cur}} \\ &= \frac{1-2^6}{2^7} - \frac{1}{2} \cdot \texttt{top} + 2^7 \cdot y_2^{\text{cur}} \in \left[0, \tfrac{1}{2^7}\right) \end{aligned} \tag{113}$$

according to (109), where y_k^{new} and y_k^{cur} (ξ_k^{cur}) for $k \in \{1, 2\}$, denotes the analog state (excitation) of neuron $k \in V$, encoding the new and current contents of stack s_k, respectively.

According to Horne and Hush (1996), one can construct a binary-state (size-optimal) neural network $\mathcal{N}^c$ with integer weights that implements the finite control of Turing machine $\mathcal{M}$ (i.e. a finite automaton). In this way, $\mathcal{N}^c$ is a subnetwork of the 3ANN $\mathcal{N}$ with binary neurons in $V^c \subset V' = V \setminus \{1, 2, 3\}$, which evaluates the transition function δ of $\mathcal{M}$, introduced in (105) and (106), within four time steps by using the method of threshold circuit synthesis due to Lupanov (1973) (cf. Šíma, 2014). Moreover, one can ensure that $\mathcal{N}^c$ operates in the fully parallel mode by using the technique of Orponen (1997). Thus, $\mathcal{N}^c$ holds internally a current state q_{cur} of $\mathcal{M}$ and receives a current symbol $x \in \{0, 1\}$ under the tape head of $\mathcal{M}$ (which is stored at the

top of stack s_2) via the neuron hd $\in V^c$ implementing the `top` operation. Then, $\mathcal{N}^c$ computes $\delta\,(q_{\rm cur}, x) = (q_{\rm new}, b, d)$ within four computational steps, replaces the current state $q_{\rm cur}$ with $q_{\rm new}$, and outputs a symbol $b \in \{0,1\}$ to overwrite x, via the neuron ow $\in V^c$. In addition, $\mathcal{N}^c$ holds a current value of $c_{\rm cur} \in \{L, R\}$ which together with the calculated direction of head move $d \in \{L, R\}$, decides if a short or long instruction applies, depending on whether or not $c_{\rm cur} = d$.

At the beginning of the simulation, $\mathcal{N}^c$ holds the initial state of $\mathcal{M}$ and the stacks s_1, s_2 contain the initial tape contents including an input word $\mathbf{x} = x_1 \ldots x_n \in \{0,1\}^n$, which are encoded by the analog states at the time instant $t_0 > 0$,

$$y_1^{(t_0)} \;=\; \gamma\,(0^\omega) = \sum_{i=1}^{\infty} \frac{2^6-1}{(2^7)^{i+1}} = \frac{2^6-1}{2^7(2^7-1)} = \frac{63}{16256} \in \left[0\,,\,\tfrac{1}{2^7}\right) \tag{114}$$

$$y_2^{(t_0)} \;=\; \gamma\,(\mathbf{x}0^\omega) = \sum_{i=1}^{n} \frac{2^6\,(x_i+1)-1}{(2^7)^{i+1}} + \frac{2^6-1}{2^{n+1}(2^7-1)} \in \left[0\,,\,\tfrac{1}{2^7}\right)\,, \tag{115}$$

according to (109). In accordance with (114) and (115), the first two analog neurons can be initialized with the value $\gamma(0^\omega)$ by the technique introduced in (87). In addition, $\mathcal{N}$ reads the input word $\mathbf{x}$ whose end can be delimited by the blank symbol, bit after bit according to the input protocol (6). Each input bit is pushed to the stack by using (112) so that the initialization of $y_2^{(t_0)}$, which meets (115), is achieved in linear time $t_0 = O(n)$. We omit the technical details of this initialization process whose implementation will be clear from the detailed description of the simulation below.

One computational step of $\mathcal{M}$ is simulated within one *macrostep* of $\mathcal{N}$ which takes 7 computational steps for a short instruction, while a long one consumes 18 steps of $\mathcal{N}$. Hereafter, the computational time t of $\mathcal{N}$ is related to the macrostep. At the beginning of the macrostep when $t = 0$, the states of analog neurons $1, 2 \in V$ encode the stack contents by the rational number $z_k \in \left[0, \frac{1}{2^7}\right)$ according to (109), that is, $y_k^{(0)} = z_k$ for $k \in \{1, 2\}$. Then, $\mathcal{N}^c$ reads the top element of s_2 via the neuron hd $\in V^c$ at the time instant $t = 1$ of the macrostep, which is implemented by the integer weights

$$w_{\rm hd,0} = -1\,, \qquad w_{\rm hd,2} = 2^8\,, \tag{116}$$

implying $y_{\rm hd}^{(1)} =$ `top` by (111). On the other hand, $\mathcal{N}^c$ outputs a symbol $b \in \{0,1\}$ to overwrite the current tape cell under the head via the neuron

	pop_1	pop_2	bias	1	2	3
0	-1	-1	0	0	0	$\frac{1}{4}$
ow	0	0	0	$\frac{1}{2^8}$	0	0
c_1	-2^3	0	0	0	0	0
c_2	0	-2^3	0	0	0	0
c_3	0	0	-1	0	0	-5
c_4	0	0	0	$\frac{2^6-1}{2^{14}}$	$\frac{1-2^6}{2^7}$	0
pop_1	0	0	0	$\frac{1}{2^8}$	$-\frac{1}{2}$	0
pop_2	0	0	0	0	$-\frac{1}{2}$	0
bias	0	0	0	$-\frac{1}{4}$	$\frac{1}{4}$	0
1	0	0	0	$\frac{1}{2}$	0	$-\frac{1}{4}$
2	2^3	2^3	0	0	2	4
3	0	0	0	1	-1	0

Table 6: The weight matrix with w_{ji} in the ith row and jth column for $j \in V \setminus V^c$.

ow $\in V^c$ either at the time instant $t = 6$ for a short instruction (i.e. $y_{\mathrm{ow}}^{(6)} = b$), or at the time instant $t = 17$ for a long one (i.e. $y_{\mathrm{ow}}^{(17)} = b$), whereas the state of ow $\in V^c$ is 0 at other times, thus producing the binary sequence $0^5 b\, 0$ or $0^{16} b\, 0$, respectively.

We further extend $\mathcal{N}^c$ with four control neurons $c_1, c_2, c_3, c_4 \in V^c$ for synchronizing the stack operations. Within each macrostep of $\mathcal{N}$, the control neurons c_1, c_2, c_3, c_4 produce the sequences of binary output values, either 1111111, 1111011, 1111111, 0000010 of length 7 for a short instruction, or $1^4 01^{13}$, $1^{15} 01^2$, $1^6 01^{11}$, $0^5 10^{10} 10$ of length 18 for a long instruction, respectively, which can easily be implemented by a finite automaton and incorporated within $\mathcal{N}^c$.

For realizing the stack operations, the binary neurons $\mathrm{pop}_1, \mathrm{pop}_2, \mathrm{bias} \in V' \setminus V^c$ and the third auxiliary analog unit $3 \in V \setminus V'$ are introduced in $\mathcal{N}$. In Table 6, the incoming rational weights to the relevant neurons in $V \setminus V^c$ are defined in the form of weight matrix with the rational entry $w_{ji} \in \mathbb{Q}$ in the ith row and jth column, where the analog neurons are separated from the

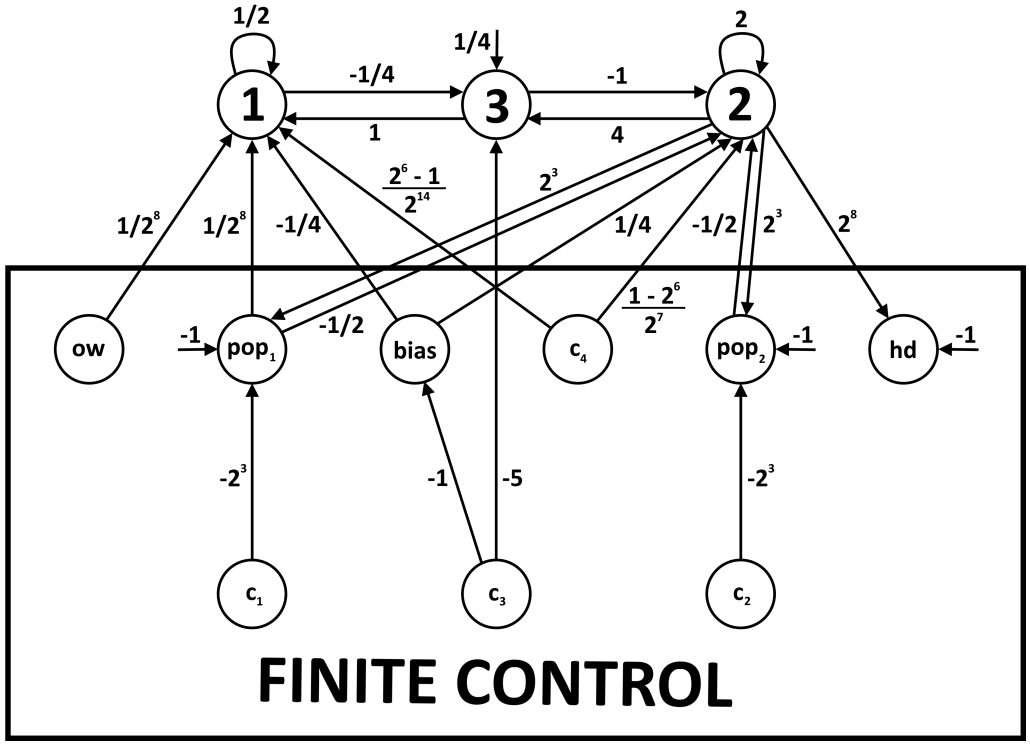

Figure 5: A schema of 3ANNs simulating TMs.

binary ones by the double lines. For example, the weights of the connections from the control neuron $c_1 \in V^c$ and from the analog neurons $2 \in V \setminus V'$ to $\mathrm{pop}_1 \in V' \setminus V^c$ are $w_{\mathtt{pop}_1,c_1} = -2^3$ and $w_{\mathtt{pop}_1,2} = 2^3$, respectively, whereas the bias of pop_1 is $w_{\mathtt{pop}_1,0} = -1$. In addition, a scheme of the architecture of $\mathcal{N}$ is depicted in Figure 5 where the directed edges connecting neurons are labeled with these weights, while the edges drawn without the originating unit $0 \in V^c$ correspond to the biases.

We will verify the implementation of the long instruction including the short one, within one macrostep of $\mathcal{N}$ which is composed of 18 network state updates. The state evolution of relevant neurons during the macrostep is presented in Table 7 which also shows the short instruction when the block bounded by the horizontal double lines corresponding to the time interval from $t = 6$ to $t = 16$ within the long instruction, is skipped. Moreover, alternatives for the short instruction are presented after the slash symbol, e.g.

t	$y_{\rm hd}^{(t)}$	$y_{\rm ow}^{(t)}$	$y_{c_1}^{(t)}$	$y_{c_2}^{(t)}$	$y_{c_3}^{(t)}$	$y_{c_4}^{(t)}$	$y_{\rm pop_1}^{(t)}$	$y_{\rm pop_2}^{(t)}$	$y_{\rm bias}^{(t)}$	$y_1^{(t)}$	$y_2^{(t)}$	$y_3^{(t)}$
0		0	1	1	1	0	0	0	0	z_1	z_2	0
1	top	0	1	1	1	0	0	0	0	$\frac{z_1}{2}$	$2z_2$	0
2		0	1	1	1	0	0	0	0	$\frac{z_1}{2^2}$	2^2z_2	0
3		0	1	1	1	0	0	0	0	$\frac{z_1}{2^3}$	2^3z_2	0
4		0	1	1	1	0	0	0	0	$\frac{z_1}{2^4}$	2^4z_2	0
5		0	0/1	1/0	1	0	0	0	0	$\frac{z_1}{2^5}$	2^5z_2	0
6		0	1	1	1	1	top	0	0	$\frac{z_1}{2^6}$	2^6z_2	0
7		0	1	1	0	0	0	0	0	$z_1'\,(123)$	$z_2'\,(124)$	0
8		0	1	1	1	0	0	0	1	$\frac{z_1'}{2}$	$2z_2'$	$\frac{1}{4}-\frac{z_1'}{4}+4z_2'$
9		0	1	1	1	0	0	0	0	$4z_2'$	$\frac{z_1'}{4}$	0
10		0	1	1	1	0	0	0	0	$2z_2'$	$\frac{z_1'}{2}$	0
11		0	1	1	1	0	0	0	0	z_2'	z_1'	0
12		0	1	1	1	0	0	0	0	$\frac{z_2'}{2}$	$2z_1'$	0
13		0	1	1	1	0	0	0	0	$\frac{z_2'}{2^2}$	$2^2z_1'$	0
14		0	1	1	1	0	0	0	0	$\frac{z_2'}{2^3}$	$2^3z_1'$	0
15		0	1	1	1	0	0	0	0	$\frac{z_2'}{2^4}$	$2^4z_1'$	0
16		0	1	0	1	0	0	0	0	$\frac{z_2'}{2^5}$	$2^5z_1'$	0
17/6		b	1	1	1	1	0	top	0	$\frac{z_2'}{2^6}/\frac{z_1}{2^6}$	$2^6z_1'/2^6z_2$	0
$18/7\equiv 0$		0	1	1	1	0	0	0	0	$z_1''\,(133)$	$z_2''\,(134)$	0

Table 7: The macrostep of 3ANN $\mathcal{N}$ simulating one long/short instruction of TM $\mathcal{M}$.

$t = 17/6$ means the seventeenth/sixth computational step of the long/short instruction within the macrostep.

Observe that for every $t = 1, \ldots, 18$ and $k \in \{1, 2\}$,

$$y_{\rm pop_k}^{(t)} = 0 \quad \text{if} \quad (k = 1 \,\&\, t \neq 6) \text{ or } (k = 2 \,\&\, t \neq 17) \tag{117}$$

since

$$\begin{aligned} \xi_{\rm pop_k}^{(t-1)} &= w_{\mathtt{pop}_k,0} + w_{\mathtt{pop}_k,c_k} y_{c_k}^{(t-1)} + w_{\mathtt{pop}_k,2} y_2^{(t-1)} \\ &= -1 - 2^3 y_{c_k}^{(t-1)} + 2^3 y_2^{(t-1)}\,, \end{aligned} \tag{118}$$

by Table 6, reducing to $\xi_{\rm pop_k}^{(t-1)} = -1 - 2^3 + 2^3 y_2^{(t-1)} < 0$ for $y_{c_k}^{(t-1)} = 1$ which holds for $(k = 1 \,\&\, t \neq 6)$ or $(k = 2 \,\&\, t \neq 17)$. Similarly, we have

$$y_{\rm bias}^{(t)} = 1 \quad \text{iff} \quad y_{c_3}^{(t-1)} = 0 \quad \text{iff} \quad t = 8 \quad \text{for every } t = 1, \ldots, 18\,, \tag{119}$$

because $\xi_{\text{bias}}^{(t-1)} = w_{\text{bias},c_3} y_{c_3}^{(t-1)} = -y_{c_3}^{(t-1)} \geq 0$ iff $y_{c_3}^{(t-1)} = 0$. Furthermore,

$$y_3^{(t)} = 0 \quad \text{if} \quad t \neq 8 \quad \text{for every } t = 1, \ldots, 18\,, \tag{120}$$

since $\xi_3^{(t-1)} = w_{30} + w_{3,c_3} y_{c_3}^{(t-1)} + w_{31} y_1^{(t-1)} + w_{32} y_2^{(t-1)} = \frac{1}{4} - 5y_{c_3}^{(t-1)} - \frac{1}{4} y_1^{(t-1)} + 4y_2^{(t-1)}$ which implies $\xi_3^{(t-1)} < 0$ for $y_{c_3}^{(t-1)} = 1$ holding for $t \neq 8$.

For a given symbol under the head of $\mathcal{M}$ held in $y_{\text{hd}}^{(1)}$ at the time instant $t = 1$ according to (116), the binary-state subnetwork $\mathcal{N}^c$ evaluates the transition function δ of $\mathcal{M}$ during four computational steps for $t = 2, 3, 4, 5$, deciding whether a long or short instruction occurs, which is indicated through the state $y_{c_1}^{(5)}$ of control neuron c_1 at the time instant $t = 5$, that is, $y_{c_1}^{(5)} = 0$ iff a long instruction applies. In the meantime, the state of analog unit $k \in \{1, 2\}$, starting with $y_k^{(0)} = z_k \in \left[0, \frac{1}{2^7}\right)$, is multiplied by its self-loop weight w_{kk} at each time instant $t = 1, \ldots, 6$, producing

$$y_k^{(t)} = w_{kk}^t z_k = \begin{cases} \frac{z_1}{2^t} \in \left[0, \frac{1}{2^{t+7}}\right) & \text{if } k = 1 \\ \\ 2^t z_2 \in \left[0, \frac{1}{2^{7-t}}\right) & \text{if } k = 2 \end{cases} \quad \text{for } t = 0, \ldots, 6\,, \tag{121}$$

since $y_{\text{ow}}^{(t)} = y_{c_4}^{(t)} = y_{\text{pop}_1}^{(t)} = y_{\text{pop}_2}^{(t)} = y_{\text{bias}}^{(t)} = y_3^{(t)} = 0$ for every $t = 0, \ldots, 5$ due to (117), (119), and (120).

For a long instruction, we have $y_{c_1}^{(5)} = 0$ which implies

$$\xi_{\text{pop}_1}^{(5)} = -1 - 2^3 y_{c_1}^{(5)} + 2^3 y_2^{(5)} = -1 + 2^8 z_2 \tag{122}$$

according to (118) and (121). Hence, $y_{\text{pop}_1}^{(6)} = \texttt{top}$ by (111), which gives

$$\begin{aligned} y_1^{(7)} &= w_{1,\text{ow}} y_{\text{ow}}^{(6)} + w_{1,c_4} y_{c_4}^{(6)} + w_{1,\text{pop}_1} y_{\text{pop}_1}^{(6)} + w_{1,\text{bias}} y_{\text{bias}}^{(6)} + w_{11} y_1^{(6)} + w_{13} y_3^{(6)} \\ &= \frac{2^6 - 1}{2^{14}} + \frac{1}{2^8} \cdot \texttt{top} + \frac{z_1}{2^7} = z_1' \in \left[0\,, \frac{1}{2^7}\right) \end{aligned} \tag{123}$$

by Table 6, since $y_{\text{ow}}^{(6)} = y_{\text{bias}}^{(6)} = y_3^{(6)} = 0$, $y_{c_4}^{(6)} = 1$, and $y_1^{(6)} = \frac{z_1}{2^6}$ due to (119), (120), and (121). It follows from (112) and (123) that z_1' encodes the contents of the stack s_1 after the first operation $\texttt{push}(\texttt{top})$ of long instruction (108) has been applied to $\gamma^{-1}(z_1)$. Similarly,

$$\begin{aligned} y_2^{(7)} &= w_{2,c_4} y_{c_4}^{(6)} + w_{2,\text{pop}_1} y_{\text{pop}_1}^{(6)} + w_{2,\text{pop}_2} y_{\text{pop}_2}^{(6)} + w_{22} y_2^{(6)} \\ &= \frac{1 - 2^6}{2^7} - \frac{1}{2} \cdot \texttt{top} + 2^7 z_2 = z_2' \in \left[0\,, \frac{1}{2^7}\right) \end{aligned} \tag{124}$$

due to $y^{(6)}_{\mathrm{pop}_2} = 0$ and $y^{(6)}_2 = 2^6 z_2$ by (117) and (121), respectively. According to (113) and (124), we thus know that z'_2 encodes the contents of the stack s_2 after the second operation `pop` of long instruction (108) has been applied to $\gamma^{-1}(z_2)$.

The `swap` operation starts at the time instant $t = 8$ when

$$y^{(8)}_1 = w_{11} y^{(7)}_1 = \frac{z'_1}{2} \in \left[0\,,\,\tfrac{1}{2^8}\right) \tag{125}$$

$$y^{(8)}_2 = w_{22} y^{(7)}_2 = 2z'_2 \in \left[0\,,\,\tfrac{1}{2^6}\right) \tag{126}$$

$$y^{(8)}_3 = w_{30} + w_{31} y^{(7)}_1 + w_{32} y^{(7)}_2 = \frac{1}{4} - \frac{z'_1}{4} + 4z'_2 \in \left[\tfrac{2^7-1}{2^9}\,,\,\tfrac{2^3+1}{2^5}\right) \tag{127}$$

according to (123), (124), and Table 6, since $y^{(7)}_{\mathrm{ow}} = y^{(7)}_{c_3} = y^{(7)}_{c_4} = y^{(7)}_{\mathrm{pop}_1} = y^{(7)}_{\mathrm{pop}_2} = y^{(7)}_{\mathrm{bias}} = 0$ due to (117) and (119). At the time instant $t = 9$, we have

$$\begin{aligned} y^{(9)}_1 &= w_{1,\mathrm{bias}} y^{(8)}_{\mathrm{bias}} + w_{11} y^{(8)}_1 + w_{13} y^{(8)}_3 \\ &= -\frac{1}{4} + \frac{z'_1}{4} + \frac{1}{4} - \frac{z'_1}{4} + 4z'_2 = 4z'_2 \in \left[0\,,\,\tfrac{1}{2^5}\right) \end{aligned} \tag{128}$$

$$\begin{aligned} y^{(9)}_2 &= w_{2,\mathrm{bias}} y^{(8)}_{\mathrm{bias}} + w_{22} y^{(8)}_2 + w_{23} y^{(8)}_3 \\ &= \frac{1}{4} + 4z'_2 - \frac{1}{4} + \frac{z'_1}{4} - 4z'_2 = \frac{z'_1}{4} \in \left[0\,,\,\tfrac{1}{2^9}\right) \end{aligned} \tag{129}$$

by (125)–(127) and Table 6, since $y^{(8)}_{\mathrm{ow}} = y^{(8)}_{c_4} = y^{(8)}_{\mathrm{pop}_1} = y^{(8)}_{\mathrm{pop}_2} = 0$ and $y^{(8)}_{\mathrm{bias}} = 1$ due to (117) and (119). This means that the respective multiples of z'_1 and z'_2 are exchanged between the analog neurons 1 and 2, cf. (125), (126) and (128), (129), respectively. Analogously to (121), the state of analog unit $k \in \{1, 2\}$, starting with $y^{(9)}_k$ in (128) and (129), respectively, is further multiplied by its self-loop weight w_{kk} at each time instant $t = 10, \ldots, 17$, producing

$$y^{(t)}_k = w^t_{kk} z_k = \begin{cases} \frac{z'_2}{2^{t-11}} \in \left[0, \frac{1}{2^{t-4}}\right) & \text{if } k = 1 \\ 2^{t-11} z'_1 \in \left[0, \frac{1}{2^{18-t}}\right) & \text{if } k = 2 \end{cases} \quad \text{for } t = 9, \ldots, 17\,, \tag{130}$$

since $y^{(t)}_{\mathrm{ow}} = y^{(t)}_{c_4} = y^{(t)}_{\mathrm{pop}_1} = y^{(t)}_{\mathrm{pop}_2} = y^{(t)}_{\mathrm{bias}} = y^{(t)}_3 = 0$ for every $t = 9, \ldots, 16$ due to (117), (119), and (120). Thus, the `swap` operation is finished at the time instant $t = 11$ when $y^{(11)}_1 = z'_2$ and $y^{(11)}_2 = z'_1$.

Similarly to (122), $y^{(16)}_{c_2} = 0$ ensures

$$\xi^{(16)}_{\mathrm{pop}_2} = -1 - 2^3 y^{(16)}_{c_2} + 2^3 y^{(16)}_2 = -1 + 2^8 z'_1 \tag{131}$$

according to (118) and (130), which implies

$$y_{\mathrm{pop}_2}^{(17)} = \mathtt{top} \tag{132}$$

by (111). At the time instant $t = 18$, (123) reads as

$$\begin{aligned} y_1^{(18)} &= w_{1,\mathrm{ow}} y_{\mathrm{ow}}^{(17)} + w_{1,c_4} y_{c_4}^{(17)} + w_{11} y_1^{(17)} \\ &= \frac{2^6 - 1}{2^{14}} + \frac{1}{2^8} \cdot b + \frac{z_2'}{2^7} = z_1'' \in \left[0\,,\,\tfrac{1}{2^7}\right) , \end{aligned} \tag{133}$$

since $y_{\mathrm{pop}_1}^{(17)} = y_{\mathrm{bias}}^{(17)} = y_3^{(17)} = 0$, $y_{\mathrm{ow}}^{(17)} = b$, $y_{c_4}^{(17)} = 1$, and $y_1^{(17)} = \frac{z_2'}{2^6}$ due to (117), (119), (120), and (130). It follows from (112) and (133) that z_1'' encodes the contents of the stack s_1 after the fourth operation $\mathtt{push}(b)$ of long instruction (108) has been applied to $\gamma^{-1}(z_2')$. Similarly to (124),

$$\begin{aligned} y_2^{(18)} &= w_{2,c_4} y_{c_4}^{(17)} + w_{2,\mathrm{pop}_2} y_{\mathrm{pop}_2}^{(17)} + w_{22} y_2^{(17)} \\ &= \frac{1 - 2^6}{2^7} - \frac{1}{2} \cdot \mathtt{top} + 2^7 z_1' = z_2'' \in \left[0\,,\,\tfrac{1}{2^7}\right) \end{aligned} \tag{134}$$

by (130) and (132). According to (113) and (134), we thus know that z_2'' encodes the contents of the stack s_2 after the fifth operation $\mathtt{pop}$ of (108) has been applied to $\gamma^{-1}(z_1')$, which completes the macrostep of $\mathcal{N}$ for a long instruction. For a short instruction when $y_{c_1}^{(5)} = 1$, $y_{c_2}^{(5)} = 0$, $y_1^{(5)} = \frac{z_1}{2^5}$ and $y_2^{(5)} = 2^5 z_2$, which coincides with a long instruction at the time instant $t = 16$, the $\mathtt{push}(b)$ and $\mathtt{pop}$ operations of (107) are implemented analogously.

Finally, if $\mathcal{M}$ halts at the computational time $T(n)$ because the next transition (106) is not defined on the current state q_{cur} and the current tape symbol, then $\mathcal{N}^c$ activates the neuron $\mathrm{nxt} \in V \setminus V^c$, while in the next step the neuron $\mathrm{out} \in V \setminus V^c$ signals whether the input word $\mathbf{x}$ is accepted by $\mathcal{M}$. This means $y_{\mathrm{nxt}}^{(\tau_{n+1}-1)} = 1$, and $\mathbf{x} \in \mathcal{L}(\mathcal{N})$ iff $y_{\mathrm{out}}^{(\tau_{n+1})} = 1$ iff $q_{\mathrm{cur}} \in F$ iff $\mathbf{x} \in \mathcal{L}(\mathcal{M})$, according to the output protocol (7), in which now $\mathcal{N}$ does not recognize the prefixes of the input word $\mathbf{x}$ just as $\mathcal{M}$ does not do it. It follows that $\mathcal{N}$ simulates $\mathcal{M}$ in time $\tau_{n+1} = O(T(n))$ because each macrostep takes only constant number of network's updates, which completes the proof of Theorem 3. □

6. Conclusion

In this paper, we have established the analog neuron hierarchy FAs ≡ 0ANNs $\subsetneqq$ 1ANNs $\subsetneqq$ 2ANNs $\subseteq$ 3ANNs ≡ TMs (see Figure 1) for the model

αANNs of discrete-time binary-state recurrent NNs with the Heaviside activation function, which are extended with α analog neurons employing the saturated-liner activations and rational weights. We have compared this hierarchy to that of Chomsky, which refines the analysis of the computational power of NNs between the binary and analog states, corresponding to FAs (Chomsky level 3) and TMs (Chomsky level 0), respectively. Namely, we have proven that the DCFL $L_{\#} = \{0^n 1^n \,|\, n \geq 1\}$ cannot be recognized by any 1ANN with one analog neuron (Theorem 1), while any DCFL (at Chomsky level 2) including $L_{\#}$ (Example 2) can be accepted by a 2ANN with two analog units (Theorem 2). We have shown that the analog neuron hierarchy collapses to the third level 3ANNs by simulating any TM with three analog neurons and a linear-time overhead (Theorem 3).

We conjecture that Theorem 1 can be generalized so that any non-regular CFL cannot be recognized by 1ANNs, which holds at least in the deterministic case (Šíma and Plátek, 2019). On the other hand, it is an open question whether there is a non-context-sensitive language that can be accepted *offline* by a 1ANN, which does not apply to an online input/output protocol since we know online 1ANNs $\subset$ CSLs. The most important challenge for further research is the separation 2ANNs $\subsetneq$ 3ANNs of the third level in the analog neuron hierarchy.

It appears that the analog neuron hierarchy is only partially comparable to that of Chomsky since 1ANNs and probably also 2ANNs do not coincide with the Chomsky levels although 0ANNs and 3ANNs correspond to FAs and TMs, respectively. In our previous paper (Šíma, 2019b), the class of languages accepted by 1ANNs has been characterized syntactically by so-called cut languages which represent a new type of basis languages defined by NNs that do not have an equivalent in the Chomsky hierarchy. A similar characterization still needs to be done for 2ANNs.

The presented results show what is the role of analogicity in the computational power of NNs. The binary states restrict NNs to a finite domain while the analog values create a potentially infinite state space which can be exploited for recognizing more complex languages in the Chomsky hierarchy. This is not only an issue of increasing precision of rational-number parameters in NNs but also of functional limitations of one or two analog units for decoding an information from rational states as well as for synchronizing the storage operations. An important open problem thus concerns the generalization of the hierarchy to other types of analog neurons used in practical deep networks such as LSTM, GRU, or ReLU units (Korsky and Berwick,

2019; Merrill et al., 2020). Clearly, the degree of analogicity represent another computational resource that can simply be measured by the number of analog units while a possible tradeoff with computational time can also be explored.

Nevertheless, the ultimate goal is to prove a proper "natural" hierarchy of NNs between integer and rational weights similarly as it is known between rational and real weights (Balcázar et al., 1997) and possibly, map it to known hierarchies of regular/context-free languages. This problem is related to a more general issue of finding suitable complexity measures of realistic NNs establishing the complexity hierarchies, which could be employed in practical neurocomputing, e.g. the precision of weight parameters (Weiss et al., 2018), energy complexity (Šíma, 2014), temporal coding etc.

Yet another important issue concerns grammatical inference. For a given PDA or TM, the constructions of equivalent αANNs presented in the proofs of Theorems 2 and 3, respectively, can be generated automatically by a computer program although they do not provide learning algorithms that would infer a language from training data. Nevertheless, the underlying theorems establish the principal limits (lower and upper bounds) for a few analog units to recognize more complex languages. For example, we now know that one analog unit is not Turing-complete since it cannot accept even some simple CFLs. In other words, any learning algorithm has to employ a sufficient number of analog units to be able to infer more complex grammars.

Acknowledgment

The presentation of this paper benefited from valuable suggestions of anonymous reviewers. The research was done with institutional support RVO: 67985807 and partially supported by the grant of the Czech Science Foundation No. GA19-05704S.

References

Alon, N., Dewdney, A. K., Ott, T. J., 1991. Efficient simulation of finite automata by neural nets. Journal of the ACM 38 (2), 495–514.

Balcázar, J. L., Gavaldà, R., Siegelmann, H. T., 1997. Computational power of neural networks: A characterization in terms of Kolmogorov complexity. IEEE Transactions on Information Theory 43 (4), 1175–1183.

Horne, B. G., Hush, D. R., 1996. Bounds on the complexity of recurrent neural network implementations of finite state machines. Neural Networks 9 (2), 243–252.

Indyk, P., 1995. Optimal simulation of automata by neural nets. In: Proceedings of the STACS 1995 Twelfth Annual Symposium on Theoretical Aspects of Computer Science. Vol. 900 of LNCS. pp. 337–348.

Kilian, J., Siegelmann, H. T., 1996. The dynamic universality of sigmoidal neural networks. Information and Computation 128 (1), 48–56.

Koiran, P., 1996. A family of universal recurrent networks. Theoretical Computer Science 168 (2), 473–480.

Korsky, S. A., Berwick, R. C., 2019. On the computational power of RNNs. arXiv:1906.06349.

Lupanov, O. B., 1973. On the synthesis of threshold circuits. Problemy Kibernetiki 26, 109–140.

Merrill, W., 2019. Sequential neural networks as automata. arXiv:1906.01615.

Merrill, W., Weiss, G., Goldberg, Y., Schwartz, R., Smith, N. A., Yahav, E., 2020. A formal hierarchy of RNN architectures. arXiv:2004.08500.

Minsky, M., 1967. Computations: Finite and Infinite Machines. Prentice-Hall, Englewood Cliffs.

Orponen, P., 1997. Computing with truly asynchronous threshold logic networks. Theoretical Computer Science 174 (1-2), 123–136.

Schmidhuber, J., 2015. Deep learning in neural networks: An overview. Neural Networks 61, 85–117.

Siegelmann, H. T., 1996. Recurrent neural networks and finite automata. Journal of Computational Intelligence 12 (4), 567–574.

Siegelmann, H. T., 1999. Neural Networks and Analog Computation: Beyond the Turing Limit. Birkhäuser, Boston.

Siegelmann, H. T., Sontag, E. D., 1994. Analog computation via neural networks. Theoretical Computer Science 131 (2), 331–360.

Siegelmann, H. T., Sontag, E. D., 1995. On the computational power of neural nets. Journal of Computer System Science 50 (1), 132–150.

Šíma, J., 1997. Analog stable simulation of discrete neural networks. Neural Network World 7 (6), 679–686.

Šíma, J., 2014. Energy complexity of recurrent neural networks. Neural Computation 26 (5), 953–973.

Šíma, J., 2018. Three analog units are Turing universal. In: Proceedings of the TPNC 2018 Seventh International Conference on Theory and Practice of Natural Computing. Vol. 11324 of LNCS. pp. 460–472.

Šíma, J., 2019a. Counting with analog neurons. In: Proceedings of the ICANN 2019 Twenty-Eighth International Conference on Artificial Neural Networks, Part I. Vol. 11727 of LNCS. pp. 389–400.

Šíma, J., 2019b. Subrecursive neural networks. Neural Networks 116, 208–223.

Šíma, J., Orponen, P., 2003. General-purpose computation with neural networks: A survey of complexity theoretic results. Neural Computation 15 (12), 2727–2778.

Šíma, J., Plátek, M., 2019. One analog neuron cannot recognize deterministic context-free languages. In: Proceedings of the ICONIP 2019 Twenty-Sixth International Conference on Neural Information Processing of the Asia-Pacific Neural Network Society, Part III. Vol. 11955 of LNCS. pp. 77–89.

Šíma, J., Savický, P., 2018. Quasi-periodic β-expansions and cut languages. Theoretical Computer Science 720, 1–23.

Šíma, J., Wiedermann, J., 1998. Theory of neuromata. Journal of the ACM 45 (1), 155–178.

Šorel, M., Šíma, J., 2004. Robust RBF finite automata. Neurocomputing 62, 93–110.

Weiss, G., Goldberg, Y., Yahav, E., 2018. On the practical computational power of finite precision RNNs for language recognition. arXiv:1805.04908.