One Analog Neuron Cannot Recognize Deterministic Context-Free Languages*

Jiří Šíma and Martin Plátek

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

Abstract. We analyze the computational power of discrete-time recurrent neural networks (NNs) with the saturated-linear activation function within the Chomsky hierarchy. This model restricted to integer weights coincides with binary-state NNs with the Heaviside activation function, which are equivalent to finite automata (Chomsky level 3), while rational weights make this model Turing complete even for three analog-state units (Chomsky level 0). For an intermediate model αANN of a binary-state NN that is extended with $\alpha \geq 0$ extra analog-state neurons with rational weights, we have established the analog neuron hierarchy 0ANNs $\subset$ 1ANNs $\subset$ 2ANNs $\subseteq$ 3ANNs. The separation 1ANNs $\subsetneq$ 2ANNs has been witnessed by the deterministic context-free language (DCFL) $L_{\#} = \{0^n 1^n \mid n \geq 1\}$ which cannot be recognized by any 1ANN even with real weights, while any DCFL (Chomsky level 2) is accepted by a 2ANN with rational weights. In this paper, we generalize this result by showing that any non-regular DCFL cannot be recognized by 1ANNs with real weights, which means (DCFLs $\setminus$ REG) $\subset$ (2ANNs $\setminus$ 1ANNs), implying 0ANNs = 1ANNs $\cap$ DCFLs. For this purpose, we show that $L_{\#}$ is the simplest non-regular DCFL by reducing $L_{\#}$ to any language in this class, which is by itself an interesting achievement in computability theory.

Keywords: Neural computing · Analog neuron hierarchy · Deterministic context-free language · Restart automaton · Chomsky hierarchy.

1 The Analog Neuron Hierarchy

The computational power of discrete-time recurrent neural networks (NNs) with the saturated-linear activation function[1] depends on the descriptive complexity of their weight parameters [13, 21]. NNs with *integer* weights, corresponding to binary-state (shortly binary) networks (with Boolean outputs 0 or 1), coincide with finite automata (FAs) recognizing regular languages (REG) [1, 3, 4, 9, 17, 23]. *Rational* weights make the analog-state (shortly analog) NNs (with real-valued outputs in the interval $[0, 1]$) computationally equivalent to Turing

* Research was done with institutional support RVO: 67985807 and partially supported by the grant of the Czech Science Foundation GA19-05704S.

[1] The results are partially valid for more general classes of activation functions [8, 12, 16, 24] including the logistic function [7].

machines (TMs) [4, 15], and thus (by a real-time simulation [15]) polynomial-time computations of such networks are characterized by the fundamental complexity class P. Moreover, NNs with arbitrary *real* weights can even derive "super-Turing" computational capabilities [13]. In particular, their polynomial-time computations correspond to the nonuniform complexity class P/poly while any input/output mapping (including undecidable problems) can be computed within exponential time [14]. In addition, 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 [2].

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. Although this study has been inspired by theoretical issues, NNs with different types of units/layers are widely used in practical applications, e.g. in deep learning [11], and they thus require a detailed mathematical analysis.

In our previous work [20], 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* [22] which are combined in a certain way by usual operations on languages. By using this syntactic characterization of 1ANNs we have proven a sufficient condition when a 1ANN recognizes only a regular language (Chomsky level 3), which is based on the *quasi-periodicity* [22] of some parameters derived from its real weights. In particular, a 1ANN with weights from the smallest field extension $\mathbb{Q}(\beta)$ over the rational numbers $\mathbb{Q}$ including a *Pisot number* $\beta > 1$, 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, it follows that any 1ANN with rational weights such that $w = 1/n$, accepts a regular language. 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* ρ or the *golden ratio* φ, respectively, which are Pisot numbers.

On the other hand, we have introduced examples of languages accepted by 1ANNs with rational weights that are not context-free (CFLs) [20], and they are thus above Chomsky level 2, while we have proven that any language accepted online[2] by this model is context-sensitive (CSL) at Chomsky level 1. For

[2] 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.

instance, the CSL $L_1 = \left\{ x_1 \dots x_n \in \{0,1\}^* \;\middle|\; \sum_{k=1}^{n} x_{n-k+1} \left(\frac{216}{125}\right)^{-k} < 1 \right\}$ which is not in CFLs, can be recognized by a 1ANN. In other words, the computational power of binary-state networks having integer weights can increase from REG (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 this does not bring any additional power even for real weights, was formulated.

Furthermore, we have established an analog neuron hierarchy of classes of languages recognized by binary αANNs with α extra analog units having rational weights, for $\alpha = 0, 1, 2, 3, \dots$, 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 there are non-context-free languages such as L_1 accepted by 1ANNs [20]. In contrast, we have proven 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[2] by any 1ANN with arbitrary real weights [19]. We thus know that 1ANNs are not Turing complete.

Nevertheless, we have shown that any DCFL included in Chomsky level 2 can be recognized by a 2ANN with two extra analog neurons having rational weights, by simulating a corresponding deterministic pushdown automaton (DPDA) [19]. This provides the separation 1ANNs $\subsetneq$ 2ANNs since the DCFL $L_{\#}$ is not accepted by any 1ANN. In addition, we have proven that any TM can be simulated by a 3ANN having rational weights with a linear-time overhead [18]. It follows that recursively enumerable languages at the highest 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:

$$\text{FAs} \equiv \text{0ANNs} \subsetneq \text{1ANNs} \subsetneq \text{2ANNs} \subseteq \text{3ANNs} = \text{4ANNs} = \ldots \equiv \text{TMs},$$

which is schematically depicted in Figure 1. It appears that the analog neuron hierarchy is only partially comparable to that of Chomsky.

In this paper, we further study the relation between the analog neuron hierarchy and the Chomsky hierarchy. We show that any non-regular DCFL cannot be recognized online by 1ANNs with real weights, which provides the stronger separation (DCFLs $\setminus$ REG) $\subset$ (2ANNs $\setminus$ 1ANNs), implying REG = 0ANNs = 1ANNs $\cap$ DCFLs. Thus, the class of non-regular DCFLs is contained in 2ANNs with rational weights, having the empty intersection with 1ANNs, as depicted in Figure 1. In order to prove this lower bound on the computational power of 1ANNs, we show that $L_{\#}$ is the simplest non-regular DCFL by reducing $L_{\#}$ to any language in DCFLs $\setminus$ REG. Namely, for any non-regular DCFL L, we can recognize the language $L_{\#}$ by a FA that is allowed to call an online subroutine for solving L, which represents a kind of Turing reduction known in computability

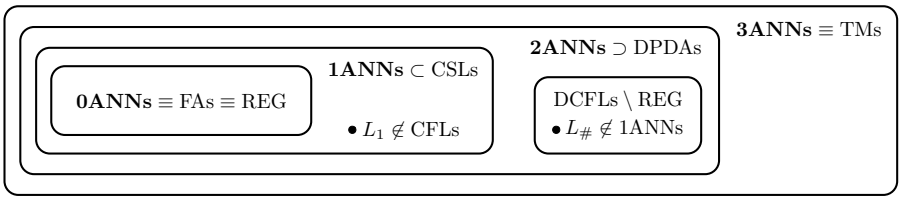

Fig. 1. The analog neuron hierarchy.

theory. Now if the language L is accepted by a 1ANN, then we could recognize $L_{\#}$ by a 1ANN, which is a contradiction, implying that L cannot be accepted by any 1ANN even with real weights. The proof exploits the technical representation of DCFLs by so-called deterministic monotonic restarting automata [5, 6].

Note that the Turing-like reduction from $L_{\#}$ to any non-regular DCFL is by itself an interesting achievement in formal language theory, providing the simplest non-regular DCFL which any language in DCFLs \ REG must include. This is somewhat opposite to the usual hardness results in computational complexity theory where all problems in a class are usually reduced to its hardest problem such as in NP-completeness. Our result can thus open a new direction of research in computability theory aiming towards the existence of the simplest problems in traditional complexity classes and their mutual reductions.

The paper is organized as follows. In Section 2, we introduce basic definitions concerning the language acceptors based on 1ANNs. In Section 3, we present the theorem that reduces the languages $L_{\#}$ to any non-regular DCFL. For this purpose we use the formalism of deterministic monotonic restarting automata, which is shortly recalled. Section 4 shows that one extra analog neuron is not sufficient for recognizing any non-regular DCFL. Finally, we summarize the results and list some open problems in Section 5.

2 Neural Language Acceptors with One Analog Unit

We specify a computational model of a *discrete-time binary-state recurrent neural network with one extra analog unit* (shortly, 1ANN), $\mathcal{N}$, which will be used as a formal language acceptor. The network $\mathcal{N}$ consists of $s \geq 1$ *units (neurons)*, indexed as $V = \{1, \ldots, s\}$. All the units in $\mathcal{N}$ are assumed to be binary-state (shortly *binary*) neurons (i.e. *perceptrons, threshold gates)* except for the last sth neuron which is an analog-state (shortly *analog*) unit. 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* $w(i, j) = w_{ji} \in \mathbb{R}$. 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 states $y_j^{(t)}$ of the first

$s-1$ binary neurons $j \in V' = V \setminus \{s\}$ are Boolean values 0 or 1, whereas the output $y_s^{(t)}$ from analog unit s is a real number from the unit interval $\mathbb{I} = [0,1]$. This establishes the *network state* $\mathbf{y}^{(t)} = \left(y_1^{(t)}, \ldots, y_{s-1}^{(t)}, y_s^{(t)}\right) \in \{0,1\}^{s-1} \times \mathbb{I}$ at each discrete time instant $t \geq 0$.

For notational simplicity, we assume a synchronous fully parallel mode without loss of efficiency [10]. At the beginning of a computation, the 1ANN $\mathcal{N}$ is placed in an *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 defined as $\xi_j^{(t)} = \sum_{i=0}^{s} w_{ji} y_i^{(t)}$, including a real *bias* value $w_{j0} \in \mathbb{R}$ which can be viewed as the weight $w(0,j)$ from a formal constant unit input $y_0^{(t)} \equiv 1$ for every $t \geq 0$ (i.e. formally $0 \in V'$). 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, $y_j^{(t+1)} = \sigma_j\left(\xi_j^{(t)}\right)$ for $j \in V$. For the neurons $j \in V'$ with binary states $y_j \in \{0,1\}$, the *Heaviside* activation function $\sigma_j(\xi) = H(\xi)$ is used where $H(\xi) = 1$ for $\xi \geq 0$ and $H(\xi) = 0$ for $\xi < 0$, while the analog unit $s \in V$ with real output $y_s \in \mathbb{I}$ employs the *saturated-linear* function $\sigma_s(\xi) = \sigma(\xi)$ where $\sigma(\xi) = \xi$ for $0 \leq \xi \leq 1$, whereas $\sigma(\xi) = 1$ for $\xi > 1$, and $\sigma(\xi) = 0$ for $\xi < 0$. In this way, the new network state $\mathbf{y}^{(t+1)} \in \{0,1\}^{s-1} \times \mathbb{I}$ is determined at time $t+1$.

The computational power of NNs has been studied analogously to the traditional models of computations [21] so that the networks are exploited as acceptors of formal languages $L \subseteq \Sigma^*$ over a finite alphabet $\Sigma = \{\lambda_1, \ldots \lambda_p\}$ composed of p letters (symbols). For a finite 1ANN $\mathcal{N}$, we use the following *online* input/output protocol employing its special neurons $\mathrm{nxt}, \mathrm{out} \in V'$. An input word (string) $\mathbf{x} = x_1 \ldots x_n \in \Sigma^n$ of arbitrary length $n \geq 0$, is sequentially presented to the network, symbol after symbol, via the first $p < s$ so-called *input neurons* $X = \{1, \ldots, p\} \subset V'$, at the time instants $0 < \tau_1 < \tau_2 < \cdots < \tau_n$ when queried by $\mathcal{N}$, where $\tau_{k+1} - \tau_k$ is bounded by a constant for every $k = 1, \ldots, n$. Thus, once the prefix $x_1, \ldots, x_{k-1}$ of $\mathbf{x}$ for $1 \leq k \leq n$, has been read, the next input symbol $x_k \in \Sigma$ is presented to $\mathcal{N}$ one computational step after $\mathcal{N}$ activates the special neuron $\mathrm{nxt} \in V'$. This means that $\mathcal{N}$ signals $y_{\mathrm{nxt}}^{(t-1)} = 1$ if $t = \tau_k$ whereas $y_{\mathrm{nxt}}^{(t-1)} = 0$ otherwise, for every $k = 1, \ldots, n$.

We employ the popular *one-hot encoding* of alphabet Σ where each letter $\lambda_i \in \Sigma$ is represented by one input neuron $i \in X$ which is activated when symbol λ_i is being read. The states of input neurons $i \in X$, which represent a current input symbol x_k at the time instant τ_k, are thus externally set as $y_i^{(t)} = 1$ if $x_k = \lambda_i \in \Sigma$ and $t = \tau_k$, whereas $y_i^{(t)} = 0$ otherwise. At the same time, $\mathcal{N}$ carries its computation 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 neuron nxt is active, i.e. $y_{\mathrm{out}}^{(\tau_{k+1}-1)} = 1$ if $x_1 \ldots x_k \in L$, and $y_{\mathrm{out}}^{(\tau_{k+1}-1)} = 0$ if $x_1 \ldots x_k \notin L$, where $\tau_{n+1} > \tau_n$ is the time instant when the input word $\mathbf{x}$ is decided. We say that a language $L \subseteq \Sigma^*$ is *accepted (recognized)* by 1ANN $\mathcal{N}$, which is denoted as $L = \mathcal{L}(\mathcal{N})$, if for any input word $\mathbf{x} \in \Sigma^*$, $\mathcal{N}$ accepts $\mathbf{x}$ iff $\mathbf{x} \in L$.

3 The Simplest Non-Regular Deterministic Language

In this section, we show that in some sense, any non-regular DCFL includes the language $L_{\#} = \{0^n 1^n \mid n \geq 1\}$ which is thus the simplest problem in the class DCFLs $\setminus$ REG. This fact will be used in Section 4 for proving that any non-regular DCFL cannot be recognized by 1ANNs since we know that $L_{\#}$ is not accepted by 1ANNs [19]. For the proof, we employ *deterministic monotonic restarting automata* (shortly, *det-mon-R-automata*) which have been shown to recognize exactly the class of DCFLs [5, 6].

Recall a det-mon-R-automaton $\mathcal{A} = (Q, \Sigma, k, I, q_0, Q_A, Q_R)$ has a finite-state control unit and one head moving on an input while possibly erasing some symbols. A finite set of its states, Q, includes the initial (start) state $q_0 \in Q$ and two disjoint subsets of accepting and rejecting states, $Q_A, Q_R \subseteq Q$, respectively. Moreover, the finite input alphabet Σ is extended with two new special symbols $¢, \$$ (originally not contained in Σ), which always start and end any input to $\mathcal{A}$, respectively, and serve as sentinels which cannot be erased. The head of $\mathcal{A}$ scans a 'window' of $k \geq 1$ consecutive symbols of the input from its position to the right (or the remaining symbols to the end of the input if the distance of the head position to the right endmarker \$ is less than k).

For an input word $s \in \Sigma^*$, the det-mon-R-automaton $\mathcal{A}$ starts in the initial state q_0, while its head position is on the left endmarker ¢ of input $¢s\$$. Then $\mathcal{A}$ carries out the computation by performing instructions from a finite set I, which are of the following two types:

1. the *move* instruction $(q, w) \longrightarrow q'$
2. the *restart* instruction $(q, w) \longrightarrow v$

The left-hand side (q, w) of an instruction determines when it is applicable, namely, the current state of $\mathcal{A}$ is $q \in Q$ and its head scans the string $w \in \Sigma^*$ composed of $k = |w|$ consecutive symbols of the input from the head position to the right (or $|w| < k$ if w ends with \$). We assume that $\mathcal{A}$ is *deterministic* which means there are no two instructions in I with the same left-hand side (q, w). The right-hand side of an instruction describes the activity to be performed. In a move instruction, $\mathcal{A}$ changes its current state to $q' \in Q$ and the head moves one symbol to the right. In a restart instruction, some of the symbols (excluding $¢, \$$) in the string w are deleted which means the scanned part w of the input, is replaced with a shorter string $v \in \Sigma^*$ where $|v| < |w|$, which is a proper subsequence of w, and $\mathcal{A}$ restarts in the initial state q_0, while its head position is again on the left endmarker ¢ of this modified input. Moreover, we assume that $\mathcal{A}$ is *monotonic* which means that the whole string v which has replaced w, will be scanned by the head before the next restart instruction is applied. This ensures that the positions of deleted symbols do not increase their distances from the right endmarker \$.

Furthermore, the subset of so-called halting states in which no instruction from I is applicable, coincide with $Q_A \cup Q_R \subseteq Q$. Thus, an input word $s \in \Sigma^*$ is accepted (recognized) by $\mathcal{A}$ if its computation on s (bounded by $¢, \$$) halts in an accepting state from Q_A. Such input words form the language $\mathcal{L}(\mathcal{A})$. For any

$s_1, s_2 \in \Sigma^*$, the notation $s_1 \Rightarrow s_2$ means that if $\mathcal{A}$ starts in the initial states q_0 with the input s_1, then this input is rewritten to s_2 when $\mathcal{A}$ finds in q_0 for the next time, while $\Rightarrow^*$ denotes the reflexive and transitive closure of the relation $\Rightarrow$. In addition, the det-mon-R-automata satisfy the *correctness preserving property* [5, 6] which guarantees that for every $s_1, s_2 \in \Sigma^*$ if $s_1 \Rightarrow^* s_2$, then $s_1 \in \mathcal{L}(\mathcal{A})$ iff $s_2 \in \mathcal{L}(\mathcal{A})$.

Theorem 1. *For any non-regular deterministic context-free language $L \subset \Sigma^*$ over a finite alphabet $\Sigma \neq \emptyset$, there exist words $u, w, z \in \Sigma^*$, nonempty strings $x, y \in \Sigma^+$, an integer $\kappa \geq 0$, and languages $L_k \in \{L, \overline{L}\}$ for $k \in K = \{-\kappa, \ldots, -1, 0, 1, \ldots, \kappa\}$, such that for every pair of integers, $m \geq 0$ and $n \geq \kappa$,*

$$\left(ux^m wy^{n+k} z \in L_k \ \textit{ for all } k \in K\right) \quad \textit{iff} \quad m = n\,. \tag{1}$$

Proof. (Sketch.) Let $L \subset \Sigma^*$ be a non-regular DCFL and assume $\mathcal{A} = (Q, \Sigma, k, I, q_0, Q_A, Q_R)$ is a det-mon-R-automaton that accepts $L = \mathcal{L}(\mathcal{A})$. It follows that $\mathcal{A}$ employs at least one restart instruction for some input since otherwise $\mathcal{A}$ would reduce to a finite automaton implying L is regular. Let $s \in \Sigma^*$ be an input presented to $\mathcal{A}$. We mark all the symbols in s (including $\text{¢}, \$$) that are scanned by the head at least once at a time instant when some restart instruction is applied in the course of computation of $\mathcal{A}$ on the input s. These marked symbols form contiguous segments in s called the *marked substrings* $\sigma_1, \ldots, \sigma_\ell \in \Sigma^*$, which are separated by non-marked symbols and have length at least k since the head of $\mathcal{A}$ scans a window of length k. Observe that the number of marked strings in s' which is derived from $s \Rightarrow^* s'$ does not increases (i.e. is at most ℓ) because $\mathcal{A}$ is monotonic, which guarantees that the windows scanned by the head in the consecutive restarts, overlap when the distance of the restart head position to the left endmarker ¢ shortens.

Suppose that the length of marked substrings could be bounded by a constant, say $k \leq |\sigma_i| \leq c$ for every $i = 1, \ldots, \ell$ and $s \in \Sigma^*$. In such a case, $\mathcal{A}$ could be modified to an equivalent $\mathcal{A}'$ recognizing the same language $L = \mathcal{L}(\mathcal{A}')$, which employs only the move instructions while the original restart operations of $\mathcal{A}$ are implemented by the finite-state control unit of $\mathcal{A}'$ when the length of the head window is extended to c symbols. This would imply that L is regular. Hence, there are inputs to $\mathcal{A}$ with marked substrings of unbounded length. In our analysis of input s, we can focus on only one marked substring σ of unbounded length since the other marked substrings in s can be eliminated by $\mathcal{A}$ through restart operations because $\mathcal{A}$ is monotonic and the marked substrings are separated by non-marked symbols.

It follows that there exists an infinite sequence of inputs $s_n \in \Sigma^*$ for $n \geq 1$, each with marked substring $\sigma_n \in \Sigma^*$, such that $s_{n+1} \Rightarrow^* s_n$ (i.e. $|s_{n+1}| > |s_n|$), and $(q, w') \longrightarrow v$ from I is the first restart instruction that is applicable when the input s_n is presented to $\mathcal{A}$, which can be assumed to be the same for every $n \geq 1$, since both the set of states Q and the set of instructions I are finite. This also ensures that the number of restart instructions that are applied in order to rewrite s_{n+1} to s_n, is bounded by a constant. Hence, $s_n = u'\sigma_n z'$ for some strings $u', z' \in \Sigma^*$ composed of symbols that are not marked in s_n, and $\sigma_n = x_n w' y_n$ for some

$x_n, y_n \in \Sigma^*$ such that the head of $\mathcal{A}$ scans $w' \in \Sigma^*$ (following x_n) in state $q \in Q$, which implies $s_{n+1} = u'x_{n+1}w'y_{n+1}z' \Rightarrow u'x_{n+1}vy_{n+1}z' \Rightarrow^* u'x_nw'y_nz' = s_n$.

Since $\mathcal{A}$ is monotonic, we have $y_{n+1} = y'_n y_n$ for some $y'_n \in \Sigma^*$, while $x_{n+1} = \chi_n x'_n$ and $x_n = \chi_n \chi'_n$ for some $\chi_n, \chi'_n, x'_n \in \Sigma^*$. The length of x'_n and y'_n is bounded due to the number of restart operations rewriting s_{n+1} to s_n is bounded, which ensures the set $\{x'_n y'_n \,|\, n \geq 1\}$ is finite. Hence, there are $x', y \in \Sigma^*$ such that $x'_n = x'$ and $y'_n = y$ for infinitely many n. By pruning the sequence (s_n), we can assume without loss of generality that $x'_n = x'$ and $y'_n = y$ are the same for every $n \geq 1$. Thus, we have $s_{n+1} = u'\chi_n x'w'yy_nz' \Rightarrow^* u'\chi_n\chi'_n w'y_nz' = s_n = u'\chi_{n-1}x'w'yy_{n-1}z'$ where $|x'y| > |\chi'_n|$ due to any restart operation deletes at least one symbol.

Suppose that $\chi'_n = v_n x'$ where $v_n \in \Sigma^*$, for all but finitely many $n \geq 1$, which implies $y \neq \varepsilon$. In this case, the derivation $s_{n+1} = u'\chi_n x'w'yy_nz' \Rightarrow^* u'\chi_n v_n x'w'y_nz' = s_n$ only swaps the substrings $x'w'$ of bounded length and y, while rewriting y with v_n through a finite number of restart instructions, for every $n \geq 1$. If this is the only type of operations in any sequence (s_n) (apart from a finite number of n such that χ'_n is a suffix of x'), then $\mathcal{A}$ would accept only a regular language, which is a contradiction. Thus, by pruning the sequence (s_n), we can assume without loss of generality that for every $n \geq 1$, $x' = x\chi'_n$ for some nonempty $x \in \Sigma^+$, which means $\chi_n = \chi_{n-1}x = \chi_1 x^{n-1}$ due to χ'_n is the same for every $n \geq 1$, defining $u = u'\chi_1$ and $w = \chi'_n w'$. We obtain $s_{n+1} = ux^n wyy_nz' \Rightarrow^* ux^{n-1}wy_nz' = s_n$.

Furthermore, suppose that for any choice of considered (s_n), the string $y = \varepsilon$ is empty, which also ensure $y_n = \varepsilon$. Thus the derivation $s_{n+1} = ux^n wz' \Rightarrow^* ux^{n-1}wz' = s_n$ only deletes x, confirming that L is regular, which is a contradiction. Hence, $y \in \Sigma^+$ and we have $y_n = yy_{n-1} = y^{n-1}y_1$, defining $z = y_1 z'$. This results in $s_{n+1} = ux^n wy^n z \Rightarrow^* ux^{n-1}wy^{n-1}z = s_n$, for every $n \geq 1$.

It also follows that $ux^{n+k}wy^n z \Rightarrow^* ux^k wz$ and $ux^n wy^{n+k}z \Rightarrow^* uwy^k z$ for every $n \geq 1$ and $k \geq 0$. We show that $L_x = L \cap \{ux^k wz \,|\, k \geq 0\}$ is regular, while a similar argument proves $L_y = L \cap \{uwy^k z \,|\, k \geq 0\}$ to be regular. Consider the inputs $ux^k wz$ to $\mathcal{A}$ for any $k \geq 1$. If $\mathcal{A}$ applies only finitely many restart instructions to these inputs, then the restarts can be implemented by the finite-state control unit while the head possibly scans a wider window, which implies L_x is regular. Thus, suppose that the number of restarts that $\mathcal{A}$ applies to $ux^k wz$ is unbounded. We know the first restart instruction can possibly be applied first when the head scans w' in $ux^k wz = u'\chi_1 x^k \chi'_n w'y_1 z'$ where recall χ'_n is constant for every $n \geq 1$. According to the previous analysis, there is a repeated cycle of restart operations rewriting and shortening the sequence of substrings x^k from the right by a regular rule, since Q, k, and $|x|$ are finite. We conclude that L_x and L_y are regular languages.

Let $K_x = \{k \geq 0 \,|\, ux^k wz \in L_x\}$ which meets $K_x = \varrho_x \cup \{aq + r \,|\, a \geq 1, r \in R_x\}$ for some integer $q \geq 1$, and sets $\varrho_x, R_x \subseteq \{0, \ldots, q-1\}$, because L_x is regular. Note that we assume without loss of generality that the period q described by R_x equals the length of preperiodic part determined by ϱ_x, since we can align the preperiodic part to a multiple of the periods while shifting this new

multiple period. Similarly, $K_y = \{k \geq 0 \,|\, uwy^k z \in L_y\} = \varrho_y \cup \{aq + r \,|\, a \geq 1, r \in R_y\}$ for $\varrho_y, R_y \subseteq \{0, \ldots, q-1\}$. Without loss of generality, we employ the same periods q in K_x and K_y by taking their least common multiple. For simplicity, we consider only the case when $R_x \neq \emptyset$ and $R_y \neq \emptyset$. If $\varrho_x = \varrho_y$ and $R_x = R_y$ for any choice of sequence (s_n), then L would be regular. Hence, either $\varrho_x \neq \varrho_y$ or $R_x \neq R_y$.

Now set $\kappa = 2q - 1$, and for every $k = 0, 1, \ldots, \kappa$, define $K_{-k} = L$ if $k \in K_x$ and $K_{-k} = \overline{L}$ if $k \notin K_x$, while $K_k = L$ if $k \in K_y$ and $L_k = \overline{L}$ if $k \notin K_y$. It follows that $ux^k wz \in L_{-k}$ and $uwy^k z \in L_k$ for every $k = 0, \ldots, \kappa$. Let $m \geq 0$ and $n \geq \kappa$. First assume that $m = n$ which implies $ux^m wy^{n-k} z \Rightarrow^* ux^k wz$ for $0 \leq k \leq n$ and $ux^m wy^{n+k} z \Rightarrow^* uwy^k z$ for every $k \geq 0$. Hence, $ux^m wy^{n+k} z \in L_k$ for every $k \in K = \{-\kappa, \ldots, -1, 0, 1, \ldots, \kappa\}$, which proves the right-to-left implication in (1).

Further assume that $m > n$, while the argument for $m < n$ is analogous. In addition, we consider the special case without preperiodic parts, which means $\varrho_x = R_x$ and $\varrho_y = R_y$, whereas the general case can be handled similarly. On the contrary, suppose that $ux^m wy^{n+k} z \in L_k$ for all $k \in K$. Denote $\delta = m - n > 0$. Let $d \in \{0, \ldots, \kappa\}$ be the remainder after dividing δ by $2q$, and $b \geq 1$ be the greatest common divisor of d and $2q$. We have $ux^m wy^{n-k} z \Rightarrow^* ux^{\delta+k} wz$ which implies $ux^{\delta+k} wz \in L_{-k}$ for $0 \leq k \leq \kappa$. Hence, $L_k = L_{k-d}$ for $k = -\kappa + d, \ldots, 0$ whereas $L_k = L_{k-d+\kappa+1}$ for $k = -\kappa, \ldots, -\kappa + d - 1$, which is resolved as $L_i = L_{i-bj}$ for every $i = -b+1, \ldots, 0$ and $j = 1, \ldots, \frac{2q}{b} - 1$. Similarly, $ux^m wy^{n+k} z \Rightarrow^* ux^{\delta-k} wz \in L_k$ for $0 \leq k \leq \min(\delta, \kappa)$, and $ux^m wy^{n+k} z \Rightarrow^* uwy^{k-\delta} z \in L_k$ for $\delta + 1 \leq k \leq \kappa$. Hence, $L_k = L_{k-d}$ for every $k = 0, \ldots, \kappa$, which imposes $L_i = L_{i+bj}$ for every $i = 0, \ldots, b-1$ and $j = 1, \ldots, \frac{2q}{b} - 1$. It follows that $R_x = R_y$ which is a contradiction, completing the proof of the left-to-right implication in (1). □

Example 1. We illustrate Theorem 1 on a simple example of the non-regular deterministic context free language L over the binary alphabet $\Sigma = \{0, 1\}$ that is composed of words containing more zeros then ones. For this language L, Theorem 1 provides the empty words $u = w = z = \varepsilon$, the non-empty strings $x = 0 \in \Sigma^+$, $y = 1 \in \Sigma^+$, the integer $\kappa = 1$, and the languages $L_{-1} = L$ and $L_0 = L_1 = \overline{L}$ such that for every pair of integers, $m \geq 0$ and $n \geq 1$, condition (1) holds:

$$\begin{aligned}\big(0^m 1^{n-1} \in L_{-1} = L \;\&\; 0^m 1^n \in L_0 = \overline{L} \;\&\; 0^m 1^{n+1} \in L_1 = \overline{L}\big) \;\text{ iff}\\ (m > n-1 \;\&\; m \leq n \;\&\; m \leq n+1) \;\text{ iff }\; m = n\,.\end{aligned}$$

4 One Analog Unit Doesn't Accept Non-Regular DCFLs

In this section we show the main result that any non-regular DCFL cannot be recognized online by a binary-state 1ANN extended with one extra analog unit, which gives the stronger separation (DCFLs \ REG) $\subset$ (2ANNs \ 1ANNs) in the analog neuron hierarchy, implying REG = 0ANNs = 1ANNs $\cap$ DCFLs. For this purpose, we exploit the fact that the DCFL $L_\#$ is not accepted by any 1ANN:

Theorem 2. *[19, Theorem 1] The deterministic context-free language* $L_{\#} = \{0^n 1^n \mid n \geq 1\}$ *cannot be recognized by any 1ANN with one extra analog unit having real weights.*

According to Theorem 1, the language $L_{\#}$ reduces to any non-regular DCFL, which can be implemented by a binary NN, providing the following theorem.

Theorem 3. *Any non-regular deterministic context-free language* L *cannot be recognized online by any 1ANN with one extra analog unit having real weights.*

Proof. (Sketch.) Let $L \subset \Sigma^*$ be a non-regular deterministic context-free language over a finite alphabet Σ including $p > 0$ symbols. On the contrary assume that there is a 1ANN $\mathcal{N}$ with the set of neurons V, that accepts $L = \mathcal{L}(\mathcal{N})$. We will outline a construction of a bigger 1ANN $\mathcal{N}_{\#}$ with the set of neurons $V_{\#} \supset V$, recognizing the language $L_{\#} = \{0^n 1^n \mid n \geq 1\}$ over the binary alphabet $\{0, 1\}$, which incorporates $\mathcal{N}$ as its subnetwork. Let $u, w, z \in \Sigma^*$ and $x, y \in \Sigma^+$ be the strings, $\kappa \geq 0$ be the integer, and L_k for $k \in K = \{-\kappa, \ldots, -1, 0, 1, \ldots, \kappa\}$ be the languages guaranteed by Theorem 1 for L. Since the class of languages accepted by 1ANNs is clearly closed under intersection with regular languages, we can confine ourselves only to strings $0^m 1^n$ for sufficiently large integers $m, n \geq \kappa$, which represent the inputs to $\mathcal{N}_{\#}$. Any such input $0^m 1^n$ is transformed to the strings $ux^m wy^{n+k} z \in \Sigma^*$ for all $k \in K$, which are presented as inputs to $\mathcal{N}$.

We first transform $\mathcal{N}$ to a 1ANN $\mathcal{N}'$ so that the output neuron out $\in V$ of $\mathcal{N}'$ decides about each prefix of an input string that has been read so far according to the input/output protocol (see Section 2), as if this prefix extended with the string z is presented to $\mathcal{N}$. The idea of building $\mathcal{N}'$ issues from the representation theorem [20, Theorem 4] characterizing syntactically the languages accepted by 1ANNs. According to this theorem, the state domain $\mathbb{I}$ of the only analog unit $s \in V$ can be partitioned into a finite number of subintervals so that the binary states $y_1^{(t+1)}, \ldots, y_{s-1}^{(t+1)} \in \{0, 1\}$ at the next time instant $t + 1$ are uniquely determined by an index of the subinterval to which the real state $y_s^{(t)} \in \mathbb{I}$ belongs, apart from the current binary states $y_1^{(t)}, \ldots, y_{s-1}^{(t)} \in \{0, 1\}$. The partition of $\mathbb{I}$ can further be refined so that the membership in its subintervals, which can easily be tested by threshold gates, determines the output from out $\in V$ as if the tail z is already read. This refinement can be achieved by continuing the computation of $\mathcal{N}$ with the state of analog unit replaced by the end-points of the original subintervals until z is read which takes a constant number of computational steps in the online input/output protocol, producing a finite partition.

We introduce an input buffer $B_1 \subset V_{\#} \setminus V$ for the subnetwork $\mathcal{N}'$, which is implemented by p parallel oriented paths of binary neurons. These disjoint paths have the same length $b \geq |uw| + (\kappa + 1)|xy|$ and each p neurons in the same distance from the first units form a layer which encodes one symbol from Σ using the one-hot encoding. Thus, B_1 stores a string from Σ^* of length at most b, which is clamped by self-loop weights. Its last layer feeds the p input neurons $X \subset V$ encoding an input symbol from Σ, at the request of $\mathcal{N}'$ which is indicated by nxt $\in V$, according to the input protocol for $\mathcal{N}'$. At the same time,

the neurons in each layer send their outputs to the subsequent layer so that the string is shifted in B_1 by one symbol forward after its last symbol is read by $\mathcal{N}'$.

On the other hand, B_1 is being filled so that it always contains an input symbol when queried by $\mathcal{N}'$. The initial states of $\mathcal{N}_\#$ ensures that B_1 contains $u \in \Sigma^*$ (and possibly some initial copies of $x \in \Sigma^+$). Furthermore, B_1 is being replenished by copies of x whose count m equals exactly the number of 0s which are being read by $\mathcal{N}_\#$ through its input neurons $X_\# \subset V_\#$ on request of $\mathrm{nxt}_\# \in V_\#$ when there is a free space in B_1. If $\mathcal{N}_\#$ reads the first 1 following the input sequence 0^m, then the string $w \in \Sigma^*$ is pushed into B_1 which is further being filled by copies of $y \in \Sigma^+$ whose count n equals exactly the number of 1s being read by $\mathcal{N}_\#$, increased by κ.

Furthermore, the subnetwork $\mathcal{N}'$ has also its output buffer B_2 for storing the last $2\kappa + 1$ states of the output neuron $\mathrm{out} \in V$ from $\mathcal{N}'$, which are recorded at the time instants when $\mathrm{nxt} \in V$ fires, according to the output protocol for $\mathcal{N}'$. This is synchronized with the input string 0^m1^n which has already been read by $\mathcal{N}_\#$ so that B_2 contains the results of whether $ux^mwy^{n+k}z$ belongs to L for every $k \in K$. The neural acceptor $\mathcal{N}_\#$ rejects the input 0^m1^n for $n < \kappa$ through the output neuron $\mathrm{out}_\# \in V_\#$ which is activated when $\mathrm{nxt}_\# \in V_\#$ fires. For $n \geq \kappa$, the network $\mathcal{N}_\#$ accepts 0^m1^n iff $ux^mwy^{n+k}z \in L_k$ for all $k \in K$, which can be determined from the contents of buffer B_2. According to Theorem 1, this happens if and only if $m = n$, which ensures $\mathcal{N}_\#$ recognizes $L_\#$. This contradicts Theorem 2 and completes the proof of Theorem 3. □

5 Conclusion

In this paper, we have refined the analysis of the computational power of discrete-time binary-state recurrent neural networks αANNs extended with α analog-state neurons by proving a stronger separation 1ANNs $\subsetneqq$ 2ANNs in the analog neuron hierarchy. Namely, we have shown that the class of non-regular deterministic context-free languages is contained in 2ANNs $\setminus$ 1ANNs, which implies 0ANNs = 1ANNs $\cap$ DCFLs. For this purpose, we have reduced the deterministic language $L_\# = \{0^n1^n \mid n \geq 1\}$, which is known to be not in 1ANNs [19], to any non-regular DCFL. This means that in some sense, $L_\#$ is the simplest problem in the class of non-regular DCFLs. This is by itself an interesting new achievement in computability theory, which can open a new direction of research aiming towards the existence of the simplest problems in traditional complexity classes as a counterpart to the hardest problems such as NP-complete problems. We conjecture that our result can be generalized to *nondeterministic* context-free languages. Another challenge for future research is an open question whether there is a non-context-sensitive language that can be accepted *offline* by a 1ANN or whether the separation 2ANNs $\subsetneqq$ 3ANNs holds.

References

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

2. Balcázar, J.L., Gavaldà, R., Siegelmann, H.T.: Computational power of neural networks: A characterization in terms of Kolmogorov complexity. IEEE Transactions on Information Theory **43**(4), 1175–1183 (1997)
3. Horne, B.G., Hush, D.R.: Bounds on the complexity of recurrent neural network implementations of finite state machines. Neural Networks **9**(2), 243–252 (1996)
4. Indyk, P.: Optimal simulation of automata by neural nets. In: Proceedings of the STACS 1995 Twelfth Annual Symposium on Theoretical Aspects of Computer Science. LNCS, vol. 900, pp. 337–348 (1995)
5. Jančar, P., Mráz, F., Plátek, M., Vogel, J.: Restarting automata. In: Proceedings of the FCT 1995 Tenth International Symposium on Fundamentals of Computation Theory. LNCS, vol. 965, pp. 283–292 (1995)
6. Jančar, P., Mráz, F., Plátek, M., Vogel, J.: On monotonic automata with a restart operation. Journal of Automata, Languages and Combinatorics **4**(4), 287–311 (1999)
7. Kilian, J., Siegelmann, H.T.: The dynamic universality of sigmoidal neural networks. Information and Computation **128**(1), 48–56 (1996)
8. Koiran, P.: A family of universal recurrent networks. Theoretical Computer Science **168**(2), 473–480 (1996)
9. Minsky, M.: Computations: Finite and Infinite Machines. Prentice-Hall, Englewood Cliffs (1967)
10. Orponen, P.: Computing with truly asynchronous threshold logic networks. Theoretical Computer Science **174**(1-2), 123–136 (1997)
11. Schmidhuber, J.: Deep learning in neural networks: An overview. Neural Networks **61**, 85–117 (2015)
12. Siegelmann, H.T.: Recurrent neural networks and finite automata. Journal of Computational Intelligence **12**(4), 567–574 (1996)
13. Siegelmann, H.T.: Neural Networks and Analog Computation: Beyond the Turing Limit. Birkhäuser, Boston (1999)
14. Siegelmann, H.T., Sontag, E.D.: Analog computation via neural networks. Theoretical Computer Science **131**(2), 331–360 (1994)
15. Siegelmann, H.T., Sontag, E.D.: On the computational power of neural nets. Journal of Computer System Science **50**(1), 132–150 (1995)
16. Šíma, J.: Analog stable simulation of discrete neural networks. Neural Network World **7**(6), 679–686 (1997)
17. Šíma, J.: Energy complexity of recurrent neural networks. Neural Computation **26**(5), 953–973 (2014)
18. Šíma, J.: Three analog units are Turing universal. In: Proceedings of the TPNC 2018 Seventh International Conference on Theory and Practice of Natural Computing. LNCS, vol. 11324, pp. 460–472 (2018)
19. Šíma, J.: Counting with analog neurons. In: Proceedings of the ICANN 2019 Twenty-Eighth International Conference on Artificial Neural Networks, Part I. LNCS, vol. 11727, pp. 389–400 (2019)
20. Šíma, J.: Subrecursive neural networks. Neural Networks **116**, 208–223 (2019)
21. Šíma, J., Orponen, P.: General-purpose computation with neural networks: A survey of complexity theoretic results. Neural Computation **15**(12), 2727–2778 (2003)
22. Šíma, J., Savický, P.: Quasi-periodic β-expansions and cut languages. Theoretical Computer Science **720**, 1–23 (2018)
23. Šíma, J., Wiedermann, J.: Theory of neuromata. Journal of the ACM **45**(1), 155–178 (1998)
24. Šorel, M., Šíma, J.: Robust RBF finite automata. Neurocomputing **62**, 93–110 (2004)