Neural Networks Between Integer and Rational Weights

Jiří Šíma
Institute of Computer Science, The Czech Academy of Sciences,
P. O. Box 5, 18207 Prague 8, Czech Republic, Email: sima@cs.cas.cz

***Abstract*—The analysis of the computational power of neural networks with the weight parameters between integer and rational numbers is refined. We study an intermediate model of binary-state neural networks with integer weights, corresponding to finite automata, which is extended with an extra analog unit with rational weights, as already two additional analog units allow for Turing universality. We characterize the languages that are accepted by this model in terms of so-called cut languages which are combined in a certain way by usual string operations. We employ this characterization for proving that the languages accepted by neural networks with an analog unit are context-sensitive and we present an explicit example of such non-context-free languages. In addition, we formulate a sufficient condition when these networks accept only regular languages in terms of quasi-periodicity of parameters derived from their weights.**

I. Introduction

The computational power of neural networks with the saturated-linear activation function[1] depends on the descriptive complexity of their weight parameters [6], [7]. Neural nets with *integer* weights, corresponding to binary-state networks, coincide with finite automata [8], [9], [10], [11], [12], [13]. *Rational* weights make the analog-state networks computationally equivalent to Turing machines [10], [14], and thus (by a real-time simulation [14]) polynomial-time computations of such networks are characterized by the complexity class P. Moreover, neural nets with arbitrary *real* weights can even derive "super-Turing" computational capabilities [6], [15]. 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. In addition, a proper hierarchy of nonuniform complexity classes between P and P/poly has been established for polynomial-time computations of neural nets with increasing Kolmogorov complexity of real weights [16].

As can be seen, our understanding of the computational power of neural networks 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 to recursively enumerable languages in the Chomsky hierarchy. It appears that a neural network that contains *two* analog-state units with rational weights, can implement two stacks of pushdown automata, a model equivalent to Turing machines [14]. A natural question arises: what is the computational power of binary-state networks including one extra analog neuron with rational weights? Such a model has been shown to be computationally equivalent to so-called finite automata with a register (FAR) whose domain is partitioned into a finite number of intervals, each associated with a local state-transition function [17].

In this paper, we characterize the class of languages that are accepted by binary-state neural networks with an extra analog unit (NN1A) in terms of so-called *cut languages* [18] which are combined in a certain way by usual operations such as complementation, intersection, union, concatenation, Kleene star, the largest prefix-closed subset, and a letter-to-letter morphism. A cut language $L_{<c}$ contains the representations of numbers in a rational base $1/a$ (where $0 < |a| < 1$) using rational digits from a finite set B [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], that are less than a given real threshold c. It has been proven [18] that a cut language $L_{<c}$ is regular iff any such a representation of c is eventually quasi-periodic, while $L_{<c}$ is not even context-free if it is not the case. Nevertheless, any cut language $L_{<c}$ with rational threshold c is context-sensitive.

By the present characterization of neural networks with an analog neuron we derive a sufficient condition when a NN1A recognizes a regular language, in terms of quasi-periodicity of some parameters depending on its weights. Furthermore, we show examples of languages accepted by NN1A that are not context-free while we prove that any language accepted by NN1A is context-sensitive. These results refine the analysis of the computational power of neural networks 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 to that between context-free and context-sensitive languages, when an extra analog unit with rational weights is added, while a condition when this does not bring any additional power is formulated.

The paper is organized as follows. In Section II, we give a brief review of basic definitions concerning the language acceptors based on NN1A. In Section III, we recall the definition of FAR which is known to be computationally equivalent to NN1A. The main technical result is presented in Section IV, which provides a characterization of languages accepted by FAR in terms of cut languages. As a consequence of this characterization we formulate a sufficient condition in Section V when a language accepted by NN1A is regular. Section VI shows a lower bound on the computational power of NN1A by an explicit example of non-context-free languages

[1]The results are valid for more general classes of activation functions [1], [2], [3], [4] including the logistic function [5].

that are recognized by NN1A, while any language accepted by NN1A proves to be context-sensitive, which represents a corresponding upper bound. Finally, we summarize the results and present some open problems in Section VII.

II. Neural Language Acceptors With an Extra Analog Unit

In this section, we will specify a computational model of a *binary-state neural network with an extra analog unit* (shortly, NN1A), N, which will be used as a formal language acceptor. In terms of computational power, NN1A stands between the binary-state neural networks with integer weights, corresponding to finite automata, and the Turing-universal analog-state networks with rational weights.

The network consists of s *units (neurons)*, indexed as $V = \{1, \ldots, s\}$. All the units in N are assumed to be binary-state *perceptrons (i.e. threshold gates)* except for the last sth neuron which is an *analog unit*. The neurons are connected into a directed graph representing an *architecture* of N, in which each edge (i, j) leading from unit i to j is labeled with a rational *weight* $w(i, j) = w_{ji} \in \mathbb{Q}$ which can be assumed to be an integer[2] for $j \in V \setminus \{s\}$. The absence of a connection within the architecture corresponds to a zero weight between the respective neurons, and vice versa.

The *computational dynamics* of 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 $s - 1$ perceptrons $j \in V \setminus \{s\}$ are binary values from $\{0, 1\}$, whereas the output $y_s^{(t)}$ from analog unit s is a rational number from the unit interval $\mathbb{I} = [0, 1] \cap \mathbb{Q}$. This establishes the *network state* $\mathbf{y}^{(t)} = (y_1^{(t)}, \ldots, y_s^{(t)}) \in \{0, 1\}^{s-1} \times \mathbb{I}$ at each discrete time instant $t \geq 0$.

At the beginning of a computation, the neural network N is placed in an *initial state* $\mathbf{y}^{(0)}$ which may also include an external input, while, for simplicity, we assume $y_s^{(0)} = 0$. At discrete time instant $t \geq 0$, an *excitation* of any neuron $j \in V$ is defined as

$$\xi_j^{(t)} = \sum_{i=0}^{s} w_{ji} y_i^{(t)} \,, \tag{1}$$

including a rational *bias* value $w_{j0} \in \mathbb{Q}$ ($w_{j0} \in \mathbb{Z}$ for $j \neq s$) which can be viewed as the weight $w(0, j)$ from a formal constant unit input $y_0^{(t)} \equiv 1$. At the next instant $t + 1$, the neurons $j \in \alpha_{t+1}$ from a selected subset $\alpha_{t+1} \subseteq V$ compute their new outputs $y_j^{(t+1)}$ in parallel by applying an *activation function* $\sigma_j : \mathbb{R} \longrightarrow \mathbb{R}$ to $\xi_j^{(t)}$, whereas the remaining units $j \notin \alpha_{t+1}$ do not update their states, that is,

$$y_j^{(t+1)} = \begin{cases} \sigma_j\left(\xi_j^{(t)}\right) & \text{for } j \in \alpha_{t+1} \\ y_j^{(t)} & \text{for } j \in V \setminus \alpha_{t+1} \,. \end{cases} \tag{2}$$

[2]Rational weights (i.e fractions) can always be replaced with integers in a binary-state perceptron by multiplying with the absolute value of their common denominator, while its function is preserved [11].

For perceptron units $j \in V \setminus \{s\}$ with binary states $y_j \in \{0, 1\}$ the *Heaviside* activation function $\sigma_j(\xi) = \sigma_H(\xi)$ is used where

$$\sigma_H(\xi) = \begin{cases} 1 & \text{for } \xi \geq 0 \\ 0 & \text{for } \xi < 0 \,, \end{cases} \tag{3}$$

while the analog-state unit $s \in V$ employs the *saturated-linear* function $\sigma_s(\xi) = \sigma_L(\xi)$ where

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

In this way, the new network state $\mathbf{y}^{(t+1)}$ at time $t + 1$ is determined.

Note that the model of NN1A can be considered as the rational-weighted neural network whose every but one activation function are the Heaviside functions σ_H while the last one is saturated-linear σ_L (cf. Footnote 2). In complementary words, NN1A is the neural network with the saturated-linear activation function σ_L, whose weights to every but one neuron are integers while the last one has rational weights.

Without loss of efficiency [32] we assume synchronous computations for which the sets α_t that define the computational dynamics of N according to (2), are predestined deterministically. Usually, sets α_t correspond to layers in the architecture of N which are updated one by one (e.g., a feedforward subnetwork). In particular, we use a systematic periodic choice of α_t so that $\alpha_{t+d} = \alpha_t$ for any $t \geq 0$ where an integer parameter $d \geq 1$ represents the number of updates within one *macroscopic time* step (e.g., d is the number of layers). We assume that the analog unit $s \in V$ is updated exactly once in every macroscopic time step, say $s \in \alpha_{d\tau}$ for every $\tau \geq 1$.

The computational power of neural networks 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 Σ (e.g., the binary alphabet $\Sigma = \{0, 1\}$). For the finite networks the following input/output protocol has been used [8], [16], [9], [10], [2], [6], [15], [14], [7], [13]. 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 so-called *input neurons* $X \subset V \setminus \{s\}$.

In particular, each input symbol $x_\tau \in \Sigma$ is encoded by the states $y_j^{(d(\tau-1)+k)}$ of input neurons $j \in X$, which are externally set (and clamped) for every $k = 0 \ldots, d - 1$ at macroscopic time instants $\tau = 1, \ldots, n$, regardless of any influence from the remaining neurons in the network. An integer $d \geq 1$ is the *time overhead* for processing a single input symbol which coincides with the macroscopic time step. Then, a so-called *output neuron* out $\in V \setminus \{s\}$ signals at macroscopic time instant n whether the input word belongs to the underlying language L, that is,

$$y_{\text{out}}^{(dn)} = \begin{cases} 1 & \text{for } \mathbf{x} \in L \\ 0 & \text{for } \mathbf{x} \notin L \,. \end{cases} \tag{5}$$

Thus, a language $L \subseteq \Sigma^*$ is *accepted (recognized) by NN1A* N, which is denoted by $L = L(N)$, if for any input word $\mathbf{x} \in \Sigma^*$, $\mathbf{x}$ is accepted by N iff $\mathbf{x} \in L$.

III. Finite Automata With a Register

The model of NN1A introduced in Section II has been shown to be computationally equivalent to a formally simpler (deterministic) *finite automaton with a register* (shortly, FAR) [17] which is reminiscent of today's already classical definition of finite automaton with multiplication [33]. In the following, we will thus use the FAR model for analyzing the computational power of NN1A.

In particular, a FAR is formally a nine-tuple $A = (Q, \Sigma, \{I_1, \ldots, I_p\}, a, (\Delta_1, \ldots, \Delta_p), \delta, q_0, z_0, F)$ where, as usual, Q is a finite set of *states* including a *start (initial) state* $q_0 \in Q$ and a subset $F \subseteq Q$ of *accept (final) states*, and Σ is a finite *input alphabet*. In addition, the automaton is augmented with a register which stores a rational number $z \in \mathbb{I} = [0,1] \cap \mathbb{Q}$. Domain $\mathbb{I}$ is partitioned into a finite number of intervals $I_1, \ldots, I_p$, possibly of different types: open, closed, half-closed, or degenerate (i.e. containing a single point) bounded intervals with rational endpoints. Each such an interval I_r is associated with a usual local state-transition function $\delta_r : Q \times \Sigma \longrightarrow Q$ which is employed if the current register value z falls into this interval I_r.

Furthermore, we have a rational *shift* function $\Delta_r : Q \times \Sigma \longrightarrow \mathbb{Q}$ for each interval I_r, $r = 1, \ldots, p$. The register is initialized to a *start (initial)* value $z_0 \in \mathbb{I}$, and during each state transition, its value $z \in \mathbb{I}$ is updated to $\sigma_L(az + \Delta_r(q,x)) \in \mathbb{I}$ by applying a linear mapping with saturation (4) having a fixed slope $a \in \mathbb{Q}$ called *multiplier* and an y-intercept $\Delta_r(q,x) \in \mathbb{Q}$ given by the shift function Δ_r for $z \in I_r$, which depends on current state $q \in Q$ and input symbol $x \in \Sigma$. In summary, for current state $q \in Q$, register value $z \in \mathbb{I}$, and input symbol $x \in \Sigma$, the global *state-transition function* $\delta : Q \times \mathbb{I} \times \Sigma \longrightarrow Q \times \mathbb{I}$ produces the new state and the new register value of automaton A as follows:

$$\delta(q, z, x) = (\delta_r(q,x), \sigma_L(az + \Delta_r(q,x))) \quad \text{if } z \in I_r\,. \quad (6)$$

An input word $\mathbf{x} \in \Sigma^*$ is accepted by A if automaton A, starting at initial state q_0 with start register value z_0, reaches a final state $q \in F$ by a sequence of state transitions according to (6), while reading the input $\mathbf{x}$ from left to right. A language $L \subseteq \Sigma^*$ is *accepted (recognized) by FAR* A, which is denoted by $L = L(A)$, if for any input word $\mathbf{x} \in \Sigma^*$, $\mathbf{x}$ is accepted by A iff $\mathbf{x} \in L$.

The following theorem shows that FAR is computationally equivalent to the NN1A introduced in Section II.

Theorem 1: [17, Theorems 1 and 2][3] Let $L \subseteq \Sigma^*$ be a language over a finite alphabet Σ. There is a binary-state neural network with an analog unit N that accepts $L = L(N)$ iff there exists a finite automaton with a register A such that $L = L(A)$.

Theorem 1 has been proven by mutual simulations [17]. In particular, given a neural network with an analog unit, N, the rational weights of N determine the parameters of a finite automaton with a register, A, that simulates N so that $L(N) = L(A)$. For example, the rational endpoints of $I_1, \ldots, I_p$ are taken from the set

$$C = \left\{ -\sum_{i=0}^{s-1} \frac{w_{ji}}{w_{js}}\, y_i \;\middle|\; j \in V \setminus (X \cup \{s\}) \text{ s.t. } w_{js} \neq 0\,, \; y_1, \ldots, y_{s-1} \in \{0,1\} \right\} \cup \{0,1\}\,, \quad (7)$$

the multiplier is given as

$$a = w_{ss}\,, \quad (8)$$

and the rational values of the shift functions are from the set $B = \bigcup_{r=1}^{p} \Delta_r(Q \times \Sigma)$ such that

$$B = \left\{ \sum_{i=0}^{s-1} w_{si} y_i \;\middle|\; y_1, \ldots, y_{s-1} \in \{0,1\} \right\}\,, \quad (9)$$

while $z_0 = y_s^{(0)} = 0$.

IV. The Characterization of FAR Languages

In this section we characterize the class of languages accepted by FAR which we know by Theorem 1 to be computationally equivalent to NN1A. In particular, we prove in Theorem 2 that any language accepted by FAR can roughly be written as a morphism applied to an intersection of a regular language with an iteration of a core language. The core language can be obtained as the largest prefix-closed subset of a union of interval languages which can easily be defined in terms of so-called cut languages (see Section V). In addition, a partial converse to Theorem 2 is shown in Theorem 3 that a language written in this form excluding the morphism can be recognized by FAR. This characterization of FAR languages is employed for analyzing the computational power of NN1A in Sections V and VI. We first start with an auxiliary lemma.

Lemma 1: For any finite automaton with a register, $A = (Q, \Sigma, \{I_1, \ldots, I_p\}, a, (\Delta_1, \ldots, \Delta_p), \delta, q_0, z_0, F)$, for any initial register value z_0', and for any refinement $\{I_1' \ldots, I_{p'}'\}$ of partition $\{I_1, \ldots, I_p\}$, there exists an equivalent automaton $A' = (Q', \Sigma, \{I_1', \ldots, I_{p'}'\}, a, (\Delta_1', \ldots, \Delta_p'), \delta', q_0', z_0', F')$ such that $L(A) = L(A')$.

Proof: The set of states, $Q' = Q \cup \{q_0'\}$, and possibly the final states, $F' = F \cup \{q_0' \,|\, q_0 \in F\}$, are extended with a new one-use start state q_0'. For each fixed $r \in \{1, \ldots, p'\}$, the local transition function and shift function are defined for any $q \in Q'$ and $x \in \Sigma$ as

$$\delta_r'(q,x) = \begin{cases} \delta_s(q,x) & \text{if } q \in Q \;\&\; I_r' \subseteq I_s \\ \delta_s(q_0,x) & \\ \qquad \text{if } q = q_0'\,,\; z_0' \in I_r'\,,\; \&\; z_0 \in I_s & \end{cases} \quad (10)$$

$$\Delta_r'(q,x) = \begin{cases} \Delta_s(q,x) \quad \text{if } q \in Q \;\&\; I_r' \subseteq I_s \\ a(z_0 - z_0') + \Delta_s(q_0,x) & \\ \qquad \text{if } q = q_0'\,,\; z_0' \in I_r'\,,\;\; \&\;\; z_0 \in I_s\,. & \end{cases} \quad (11)$$

Obviously, $L(A) = L(A')$. ∎

[3] The underlying theorems have actually been proven for the binary alphabet $\{0,1\}$ in [17] but their generalization to any finite alphabet is straightforward.

Theorem 2: Any language $L = L(A)$ that is accepted by a finite automaton with a register, $A = (Q, \Sigma, \{I_1, \ldots, I_p\}, a, (\Delta_1, \ldots, \Delta_p), \delta, q_0, z_0, F)$, where, without loss of generality (Lemma 1), $I_1 = [0,0]$, $I_p = [1,1]$, and $z_0 = 0$, can be written as

$$L = h\left((\mathcal{L} \cap R_0)^* \cdot \mathcal{L} \cap R\right) \tag{12}$$

where

- $h : \Gamma^* \longrightarrow \Sigma^*$ is a letter-to-letter morphism (i.e. $h(\Gamma) \subseteq \Sigma$) from a set of strings over a finite alphabet Γ which is partitioned into $\Gamma_1, \ldots, \Gamma_p$
- $R \subseteq \Gamma^*$ is a regular language
- $R_0 = \Gamma_\lambda^* \cdot \Gamma_\sigma$ where $\Gamma_\lambda = \Gamma \setminus \Gamma_\sigma$ and $\Gamma_\sigma = \Gamma_1 \cup \Gamma_p$
- language $\mathcal{L}$ is defined as

$$\mathcal{L} = \left(\bigcup_{r=1}^{p} L_r \cdot \Gamma_r \cup \Gamma \cup \{\varepsilon\} \right)^{Pref} \tag{13}$$

where S^{Pref} denotes the largest prefix-closed subset of S, ε is the empty string,

$$L_r = \left\{ y_1 \ldots y_k \in \Gamma_\lambda^+ \,\middle|\, \textstyle\sum_{i=0}^{k-1} b(y_{k-i}) a^i \in I'_r \right\}, \tag{14}$$

$$I'_r = \begin{cases} (-\infty, 0] & \text{if } r = 1 \\ I_r & \text{if } 1 < r < p \\ [1, \infty) & \text{if } r = p\,, \end{cases} \tag{15}$$

for $r = 1, \ldots, p$, and $b : \Gamma_\lambda \longrightarrow \mathbb{Q}$ is a mapping.

Proof: Let $A = (Q, \Sigma, \{I_1, \ldots, I_p\}, a, (\Delta_1, \ldots, \Delta_p), \delta, q_0, z_0, F)$ be a finite automaton with a register satisfying $I_1 = [0,0]$, $I_p = [1,1]$, and $z_0 = 0$. Consider a computation by A on input $x_1 \ldots x_n \in \Sigma^n$ which traverses states $q_0 = q_1, \ldots, q_n, q_{n+1} \in Q$ where $q_{n+1} \in F$ iff $x_1 \ldots x_n \in L$, with corresponding register values $0 = z_0 = z_1, \ldots, z_n \in \mathbb{I}$, respectively, such that $z_k \in I_{r_k}$ for $k = 1, \ldots, n$. Observe that $r_1 = 1$ due to $z_1 \in I_1$.

We will encode this computation by using a string $y_1 \ldots y_n \in \Gamma^*$ over finite alphabet $\Gamma = \Gamma' \cup \Gamma''$ which consists of *basic* letters

$$\Gamma' = \bigcup_{r=1}^{p} \Gamma'_r \quad \text{where} \quad \Gamma'_r = Q \times \Sigma \times \{r\} \tag{16}$$

for $r = 1, \ldots, p$, and so-called *contextual* symbols

$$\Gamma'' = (\Gamma' \setminus \Gamma_\sigma) \times \Gamma_\sigma \quad \text{where} \quad \Gamma_\sigma = \Gamma'_1 \cup \Gamma'_p\,. \tag{17}$$

In particular, the underlying string is composed of

$$y_k = \begin{cases} (q_1, x_1, r_1) = (q_0, x_1, 1) \in \Gamma' & \text{if } k = 1 \\ (q_k, x_k, r_k) \in \Gamma' & \\ \qquad \text{if } r_{k-1} \notin \{1, p\} \text{ or } r_k \in \{1, p\} & \\ ((q_k, x_k, r_k), (q_{k-1}, x_{k-1}, r_{k-1})) \in \Gamma'' & \\ \qquad \text{if } r_{k-1} \in \{1, p\} \;\&\; r_k \notin \{1, p\} & \end{cases} \tag{18}$$

for $k = 1, \ldots, p$. Note that actual register value $z_k \in \mathbb{I}$ is replaced by corresponding index $r_k \in \{1, \ldots, p\}$ of the interval I_{r_k} to which z_k belongs. In addition, we define $\Gamma_\lambda = \Gamma \setminus \Gamma_\sigma$ and partition Γ into

$$\Gamma_r = \begin{cases} \Gamma'_r & \text{if } r \in \{1, p\} \\ \Gamma'_r \cup (\Gamma'_r \times \Gamma_\sigma) & \text{if } r \in \{2, \ldots, p-1\} \end{cases} \tag{19}$$

for $r = 1, \ldots, p$.

Let $1 = k_1 < k_2 < \cdots < k_s \leq n$ be all the indices such that $y_{k_j} \in \Gamma_\sigma$, which implies $r_{k_j} \in \{1, p\}$ and $z_{k_j} \in \{0, 1\}$, for $j = 1, \ldots, s$, and formally denote $k_0 = 0$ and $k_{s+1} = n + 1$. Thus, for each $j \in \{1, \ldots, s\}$ such that $k_j + 1 < k_{j+1}$, and for every $k = k_j, \ldots, k_{j+1} - 2$, we know $y_{k+1} \in \Gamma_\lambda$ which ensures

$$0 < z_{k+1} = a z_k + \Delta_{r_k}(q_k, x_k) < 1\,, \tag{20}$$

implying

$$z_{k+1} = \sum_{i=0}^{k-k_j} \Delta_{r_{k-i}}(q_{k-i}, x_{k-i})\, a^i \;+\; a^{k-k_j+1} z_{k_j}\,. \tag{21}$$

In addition, formula (21) for $k = k_{j+1} - 1$ reads

$$\sum_{i=0}^{k_{j+1}-k_j-1} \Delta_{r_{k_{j+1}-i-1}}(q_{k_{j+1}-i-1}, x_{k_{j+1}-i-1})\, a^i + a^{k_{j+1}-k_j} z_{k_j} \begin{cases} \leq 0 & \text{if } y_{k_{j+1}} \in \Gamma_1 \\ \geq 1 & \text{if } y_{k_{j+1}} \in \Gamma_p \end{cases} \tag{22}$$

for every $j = 1, \ldots, s-1$.

Moreover, define mapping $b : \Gamma_\lambda \longrightarrow \mathbb{Q}$ for any $y \in \Gamma_\lambda$ as

$$b(y) = \begin{cases} \Delta_r(q, x) \quad \text{if } y = (q, x, r) \in \Gamma' & \\ a\Delta_1(q', x') + \Delta_r(q, x) & \\ \qquad \text{if } y = ((q, x, r), (q', x', 1)) \in \Gamma'' & \\ a^2 + a\Delta_p(q', x') + \Delta_r(q, x) & \\ \qquad \text{if } y = ((q, x, r), (q', x', p)) \in \Gamma'' & \end{cases} \tag{23}$$

where $r \notin \{1, p\}$. For each $j \in \{1, \ldots, s\}$ such that $k_j + 1 < k_{j+1}$, and for every $k = k_j + 1, \ldots, k_{j+1} - 1$, we know $y_{k_j+1} \ldots y_k \in \Gamma_\lambda^+$ where $y_{k_j+1} \in \Gamma''$ while $y_{k_j+2} \ldots y_k \in \Gamma'^*$. According to (14), $y_{k_j+1} \ldots y_k \in L_r$ iff $\sum_{i=0}^{k-k_j-1} b(y_{k-i}) a^i \in I'_r$, which can be rewritten as

$$\begin{aligned} &\sum_{i=0}^{k-k_j-2} b(q_{k-i}, x_{k-i}, r_{k-i}) a^i \\ &\quad + b((q_{k_j+1}, x_{k_j+1}, r_{k_j+1}), (q_{k_j}, x_{k_j}, r_{k_j})) a^{k-k_j-1} \\ &= \sum_{i=0}^{k-k_j} \Delta_{r_{k-i}}(q_{k-i}, x_{k-i})\, a^i \;+\; a^{k-k_j+1} z_{k_j} \in I'_r \end{aligned} \tag{24}$$

by definition (23). For every $k = k_j + 1, \ldots, k_{j+1} - 2$, formula in (24) coincides with (21), that is, $y_{k_j+1} \ldots y_k \in L_r$ iff $z_{k+1} \in I'_r$ iff $z_{k+1} \in I_r$ due to (20), iff $r_{k+1} = r$ iff $y_{k+1} \in \Gamma_r$. Similarly, for $k = k_{j+1} - 1 < n$, condition (24) agrees with (22), that is, $y_{k_j+1} \ldots y_{k_{j+1}-1} \in L_r$ iff $y_{k_{j+1}} \in \Gamma_r$. Hence, substring $y_{k_j+1} \ldots y_{k_{j+1}} \in \mathcal{L} \cap R_0 = (\bigcup_{r=1}^{p} L_r \cdot \Gamma_r \cup \Gamma \cup \{\varepsilon\})^{Pref} \cap \Gamma_\lambda^* \cdot \Gamma_\sigma$, for every $j = 1, \ldots, s-1$, since any of its prefix $y_{k_j+1} \ldots y_k \in L_{r_{k+1}} \subseteq \Gamma_\lambda^*$ (for $k_j + 1 \leq k < k_{j+1}$) is followed by $y_{k+1} \in \Gamma_{r_{k+1}}$, including $y_{k_{j+1}} \in \Gamma_\sigma$ for $k = k_{j+1} - 1$. Analogously, $y_{k_s+1} \ldots y_n \in \mathcal{L}$. In addition, for any $j \in \{0, \ldots, s-1\}$ such that $k_j + 1 = k_{j+1}$, also $y_{k_{j+1}} \in \Gamma_\sigma \subseteq \mathcal{L} \cap R_0$. It follows that any computation by A is encoded by $y_1 \ldots y_n = (\prod_{j=0}^{s-1} y_{k_j+1} \ldots y_{k_{j+1}}) \cdot y_{k_s+1} \ldots y_n \in (\mathcal{L} \cap R_0)^* \cdot \mathcal{L}$.

The role of language $R \subseteq \Gamma^*$ in (12) is to restrict strings $y_1 \dots y_n \in \mathcal{L}^*$ only to those encoding valid accepting computations of A, mainly with respect to its local transition functions $\delta_r : Q \times \Sigma \longrightarrow Q$ for $r = 1, \dots, p$, and to the consistency of symbols from Γ_σ and Γ''. In particular, these strings (if nonempty) must start with an initial letter $y_1 = (q_1, x_1, r_1) \in \Gamma'$ such that $q_1 = q_0$ is the start state of A and $r_1 = 1$ since $z_0 = 0 \in I_1$. Any subsequent letter $y_k \in \Gamma$, for $2 \leq k \leq n$, has to be either basic symbol $y_k = (q_k, x_k, r_k) \in \Gamma'$ if $r_{k-1} \notin \{1, p\}$ or $r_k \in \{1, p\}$, or contextual symbol $y_k = ((q_k, x_k, r_k), (q_{k-1}, x_{k-1}, r_{k-1})) \in \Gamma''$ which comes after letter $y_{k-1} = (q_{k-1}, x_{k-1}, r_{k-1}) \in \Gamma_\sigma$, if $r_{k-1} \in \{1, p\}$ and $r_k \notin \{1, p\}$. Each symbol y_k, for $1 \leq k < n$, must be followed by $(q_{k+1}, x_{k+1}, r_{k+1}) \in \Gamma'$ or $((q_{k+1}, x_{k+1}, r_{k+1})(q_k, x_k, r_k)) \in \Gamma''$ such that $\delta_{r_k}(q_k, x_k) = q_{k+1}$, and y_n terminates the string so that $\delta_{r_n}(q_n, x_n) \in F$ is a final state of A. In addition, $\varepsilon \in R$ if $q_0 \in F$. Furthermore, for any $j \in \{1, \dots, s-1\}$ such that $k_j + 1 = k_{j+1}$, which ensures $y_{k_j} = (q_{k_j}, x_{k_j}, r_{k_j}) \in \Gamma_\sigma$ and $y_{k_j+1} = y_{k_{j+1}} = (q_{k_{j+1}}, x_{k_{j+1}}, r_{k_{j+1}}) \in \Gamma_\sigma$ with $r_{k_j}, r_{k_{j+1}} \in \{1, p\}$, the valid computation must satisfy

$$r_{k_{j+1}} = \begin{cases} 1 & \text{if } az_{k_j} + \Delta_{r_{k_j}}(q_{k_j}, x_{k_j}) \leq 0 \\ 0 & \text{if } az_{k_j} + \Delta_{r_{k_j}}(q_{k_j}, x_{k_j}) \geq 1 \end{cases} \tag{25}$$

where $z_{k_j} = 0$ if $r_{k_j} = 1$, whereas $z_{k_j} = 1$ if $r_{k_j} = p$. Obviously, language R can be recognized by a finite automaton and hence it is regular.

Finally, the letter-to-letter morphism $h : \Gamma^* \longrightarrow \Sigma^*$ is defined as $h(y) = x$ for $y = (q, x, r) \in \Gamma'$ or for $y = ((q, x, r), (q', x', r')) \in \Gamma''$, which extracts the input strings accepted by A. This completes the proof that L can be written as (12). ∎

Since it is unclear whether the languages accepted by FAR are closed under morphism, the implication in Theorem 2 can only be partially reversed:

Theorem 3: Assume the notation as in Theorem 2. Any language $L \subseteq \Gamma^*$ that can be written as

$$L = (\mathcal{L} \cap R_0)^* \cdot \mathcal{L} \cap R \tag{26}$$

can be recognized by a finite automaton with a register, A, that is, $L = L(A)$.

Proof: Let $L \subseteq \Gamma^*$ be a language that can be written as (26). We will construct a finite automaton with a register, A, such that $L = L(A)$. First we show how to construct an automaton $A_1 = (Q, \Gamma, \{I_1, \dots, I_p\}, a, (\Delta_1, \dots, \Delta_p), \delta, q_0, z_0, F)$ that accepts language $(\mathcal{L} \cap R_0)^* \cdot \mathcal{L}$ over alphabet Γ partitioned into $\Gamma_1, \dots, \Gamma_p$, which is specified by a partition $[0, 0] = I_1, \dots, I_p = [1, 1]$ of $\mathbb{I}$, multiplier a, and mapping $b : \Gamma_\lambda \longrightarrow \mathbb{Q}$. Define $Q = \{q_0, q_1, q_2\}$, $F = \{q_0, q_1\}$, and $z_0 = 0$. For each fixed $r \in \{1, \dots, p\}$, we introduce the local transition function $\delta_r : Q \times \Gamma \longrightarrow Q$ as

$$\delta_r(q_0, y) = \begin{cases} q_0 & \text{if } y \in \Gamma_\sigma \\ q_1 & \text{if } y \in \Gamma_\lambda\,, \end{cases} \tag{27}$$

$$\delta_r(q_1, y) = \begin{cases} q_0 & \text{if } y \in \Gamma_r \ \&\ r \in \{1, p\} \\ q_1 & \text{if } y \in \Gamma_r \ \&\ 1 < r < p \\ q_2 & \text{if } y \notin \Gamma_r\,, \end{cases} \tag{28}$$

$$\delta_r(q_2, y) = q_2\,, \tag{29}$$

for any $y \in \Gamma$, and the shift function $\Delta_r : Q \times \Gamma \longrightarrow \mathbb{Q}$ as

$$\Delta_r(q, y) = \begin{cases} b(y) & \text{if } y \in \Gamma_\lambda \\ 0 & \text{if } y \in \Gamma_1 \\ -a & \text{if } y \in \Gamma_p\,, \end{cases} \tag{30}$$

for any $q \in Q$ and $y \in \Gamma$.

Let $y_1 \dots y_n \in \Gamma^*$ be an input string to A_1. Denote by $k_1 < k_2 < \cdots < k_s$ all the indices such that $y_{k_j} \in \Gamma_\sigma$, and formally define $k_0 = 0$ and $k_{s+1} = n$. String $y_1 \dots y_n$ can be split into $\prod_{j=0}^{s} y_{k_j+1} \dots y_{k_{j+1}} = (\prod_{j=0}^{s-1} y_{k_j+1} \dots y_{k_{j+1}}) \cdot y_{k_s+1} \dots y_n$ where the tail $y_{k_s+1} \dots y_n$ reduces to the empty string when $k_s = n$. Thus, input $y_1 \dots y_n \in \Gamma^*$ is in $(\mathcal{L} \cap R_0)^* \cdot \mathcal{L}$ iff for each $j \in \{0, \dots, s\}$ such that $k_j + 1 < k_{j+1}$, substring $y_{k_j+1} \cdots y_{k_{j+1}} \in \Gamma_\lambda^+ \cdot \Gamma_\sigma$, for $j < s$, or $y_{k_s+1} \dots y_n \in \Gamma_\lambda^+$, for $j = s$, belongs to $\mathcal{L}$, since $y_{k_{j+1}} \in \Gamma_\sigma \subseteq \mathcal{L}$ for $j \in \{0, \dots, s\}$ such that $k_j + 1 = k_{j+1}$, and $\varepsilon \in \mathcal{L}$ for $k_s = n$. Moreover, $y_{k_j+1} \cdots y_{k_{j+1}} \in \mathcal{L} = (\bigcup_{r=1}^{p} L_r \cdot \Gamma_r \cup \Gamma \cup \{\varepsilon\})^{Pref}$ for $k_j + 1 < k_{j+1}$ iff $y_{k_j+1} \cdots y_k \in L_r$ and $y_{k+1} \in \Gamma_r$ for every $k = k_j + 1, \dots, k_{j+1} - 1$, that is,

$$\sum_{i=0}^{k-k_j-1} b(y_{k-i}) a^i \in I'_r \quad \& \quad y_{k+1} \in \Gamma_r\,, \tag{31}$$

according to (14).

Consider automaton A_1 finds in state q_0 with register value $z_{k_j} = 0$ (e.g. at the beginning of computation when $j = 0$). If $k_j + 1 = k_{j+1}$, then an input symbol $y_{k_{j+1}} \in \Gamma_\sigma$ keeps A_1 in state q_0, due to (27), with register value $z_{k_j+1} = 0$ as $az_{k_j} + \Delta_1(q_0, y_{k_{j+1}}) \leq 0$, according to (30). If $k_j + 1 < k_{j+1}$, then $y_{k_j+1} \in \Gamma_\lambda$ moves A_1 to state q_1, by (27), which is shown below to check condition (31) for $k = k_j + 1, \dots, k_{j+1} - 1$, while reading the next input symbols $y_{k_j+2} \dots y_{k_{j+1}} \in \Gamma_\lambda^* \cdot \Gamma_\sigma$. In particular, assume that the register values satisfy

$$0 < az_\ell + \Delta_r(q, y_{\ell+1}) < 1 \tag{32}$$

for every $\ell = k_j, \dots, k-2$ ($k > k_j + 1$), $r \in \{1, \dots, p\}$, and $q \in Q$, where the shifts $\Delta_r(q, y_{\ell+1}) = b(y_{\ell+1})$ depend only on input symbols $y_{\ell+1} \in \Gamma_\lambda$, according to (30). If inequality (32) still holds for $\ell = k-1$, then $z_k = \sum_{i=0}^{k-k_j-1} b(y_{k-i}) a^i \in I_r = I'_r$ for some $r \in \{2, \dots, p-1\}$. According to (28), only $y_{k+1} \in \Gamma_r$ for this r keeps A_1 in state q_1, preserving (32), while A_1 gets stuck in reject state q_2 for $y_{k+1} \notin \Gamma_r$, which agrees with condition (31). Similarly, if inequality (32) is broken for $\ell = k - 1$, then $\sum_{i=0}^{k-k_j-1} b(y_{k-i}) a^i \in I'_r$ for some $r \in \{1, p\}$ (i.e. $z_k \in \{0, 1\}$), which requires $y_{k+1} \in \Gamma_r \subseteq \Gamma_\sigma$ (i.e. $k + 1 = k_{j+1}$) in order to move A_1 to state q_0, according to (28), while A_1 ends up in reject state q_2 for $y_{k+1} \notin \Gamma_r$, which is consistent with condition (31). Moreover,

$$z_{k_{j+1}} = az_{k_{j+1}-1} + \Delta_r(q_1, y_{k_{j+1}}) = 0 \tag{33}$$

for both $z_{k_{j+1}-1} = 0$, $y_{k_{j+1}} \in \Gamma_1$ and $z_{k_{j+1}-1} = 1$, $y_{k_{j+1}} \in \Gamma_p$, according to (30), which ensures the zero register value in state q_0.

It follows that A_1 accepts input $y_1 \dots y_n$ iff it is from $(\mathcal{L} \cap R_0)^* \cdot \mathcal{L}$, that is, $L(A_1) = (\mathcal{L} \cap R_0)^* \cdot \mathcal{L}$. The proof of Theorem 3 is completed by the following lemma. ■

Lemma 2: The languages that are accepted by finite automata with a register are closed under intersection with a regular language.

Proof: Let $A_1 = (Q, \Gamma, \{I_1, \dots, I_p\}, a, (\Delta_1, \dots, \Delta_p), \delta, q_0, z_0, F)$ be a finite automaton with a register, and $A' = (Q', \Gamma, \delta', q'_0, F')$ be an ordinary deterministic finite automaton. We define a finite automaton with a register, $A = (Q_2, \Gamma, \{I_1, \dots, I_p\}, a, (\Delta_1, \dots, \Delta_p), \gamma, (q_0, q'_0), z_0, F \times F')$, having $Q_2 = Q \times Q'$ and local transition functions $\gamma_r : Q_2 \times \Gamma \longrightarrow Q_2$ for $r = 1, \dots, p$ such that $\gamma_r((q, q'), y) = (\delta_r(q, y), \delta'(q', y))$ for any $q \in Q$, $q' \in Q'$, and $y \in \Gamma$. Obviously, $L(A) = L(A_1) \cap L(A')$. ■

V. NN1A and Regular Cut Languages

In this section, we prove a sufficient condition when a NN1A recognizes a regular language. For this purpose, we exploit the characterization of FAR languages presented in Section IV. We first recall the notion of cut languages and their relation to quasi-periodic numbers [18].

A so-called cut language contains the representations of numbers in a rational base that are less than a given threshold. Hereafter, let a be a rational number such that $0 < |a| < 1$, which is the inverse of a base (radix) $1/a$ where $|1/a| > 1$, and let $B \subset \mathbb{Q}$ be a finite set of rational digits. We say that $L \subseteq \Gamma^*$ is a *cut language* over a finite alphabet Γ if there is a mapping $b : \Gamma \longrightarrow B$ and a real threshold c such that

$$L = L_{<c} = \left\{ y_1 \dots y_k \in \Gamma^* \;\middle|\; \sum_{i=0}^{k-1} b(y_{k-i}) a^i < c \right\}. \quad (34)$$

A cut language $L_{>c}$ with the greater-than symbol is defined analogously.

Furthermore, we say that a power series $\sum_{k=0}^{\infty} b_k a^k$ with coefficients $b_k \in B$ is *eventually quasi-periodic* with *period sum* P if there is an increasing infinite sequence of its term indices $0 \le k_1 < k_2 < k_3 < \cdots$ such that for every $i \ge 1$,

$$\frac{\sum_{k=0}^{m_i - 1} b_{k_i + k}\, a^k}{1 - a^{m_i}} = P \quad (35)$$

where $m_i = k_{i+1} - k_i > 0$ is the length of *quasi-repetend* $b_{k_i}, \dots, b_{k_{i+1}-1}$, while k_1 is the length of *preperiodic part* $b_0, \dots, b_{k_1 - 1}$. One can calculate the sum of any eventually quasi-periodic power series as

$$\sum_{k=0}^{\infty} b_k a^k = \sum_{k=0}^{k_1 - 1} b_k a^k + a^{k_1} P \quad (36)$$

which does not change if any quasi-repetend is removed from associated sequence $(b_k)_{k=1}^{\infty}$ or if it is inserted in between two other quasi-repetends. This means that the quasi-repetends can be permuted arbitrarily.

We say that a real number c is *a-quasi-periodic within B* if any power series $\sum_{k=0}^{\infty} b_k a^k = c$ with $b_k \in B$ for all $k \ge 0$, is eventually quasi-periodic. Note that a number is also considered formally to be a-quasi-periodic when it cannot be written as a respective power series at all. For example, the numbers from the complement of the Cantor set are formally $(1/3)$-quasi-periodic within $\{0, 2\}$.

Example 1: We present a non-trivial example of a number c that is a-quasi-periodic within $B = \{\beta_1, \beta_2, \beta_3\}$ where $0 < a < \frac{1}{2}$, $\beta_1 = (1 - a^2)c$, $\beta_2 = a(1 - a)c$, and $\beta_3 = 0$. One can show [18, Examples 1 and 6] that $\beta_1, \beta_2^n, \beta_3$ where β_2^n means β_2 repeated n times, is a quasi-repetend of length $n + 2$ for every integer $n \ge 0$, satisfying (35) for $P = c$, and that any power series $\sum_{k=0}^{\infty} b_k a^k = c$ with $b_k \in B$ for all $k \ge 0$, is eventually quasi-periodic.

The class of regular cut languages is completely characterized by the following theorem.

Theorem 4: [18, Theorem 11] A cut language $L_{<c}$ over alphabet Γ with mapping $b : \Gamma \longrightarrow B$ and threshold $c \in \mathbb{R}$, is regular iff c is a-quasi-periodic within B.

Now, we formulate a sufficient condition when a NN1A accepts only a regular language.

Theorem 5: Let N be a neural network with an analog unit and assume the feedback weight of analog neuron s satisfies $0 < |w_{ss}| < 1$. Define $C \subset \mathbb{Q}$, $a \in \mathbb{Q}$, and $B \subset \mathbb{Q}$ according to (7), (8), and (9), respectively, where the weights of N are employed. If every $c \in C$ is a-quasi-periodic within $B' = B \cup \{0, 1\}$, then the language $L = L(N)$ accepted by N is regular.

Proof: [4] We know from Theorem 1 that there is a finite automaton with a register, $A = (Q, \Sigma, \{I_1, \dots, I_p\}, a, (\Delta_1, \dots, \Delta_p), \delta, q_0, z_0, F)$ with multiplier (8), such that $L = L(A)$ and the rational endpoints of $I_1, \dots, I_p$ are from C while $B = \bigcup_{r=1}^{p} \Delta_r(Q \times \Sigma)$. According to Theorem 2, language L accepted by A can be written as (12) where each language L_r over alphabet Γ_λ, for $1 < r < p$, associated with interval $I'_r = I_r$ by (14), can be expressed as an intersection of two cut languages (34) or their complements, for example, $L_r = L_{>c_r} \cap \overline{L_{>c_{r+1}}}$ for half-closed interval $I'_r = I_r = (c_r, c_{r+1}]$. In addition, $L_1 = \overline{L_{>0}} \setminus \{\varepsilon\}$ and $L_p = \overline{L_{<1}}$ as $0, 1 \in C$.

Note that mapping $b : \Gamma_\lambda \longrightarrow \mathbb{Q}$, defined by (23), in fact, assigns to the first letter $y_{k_{j+1}} \in \Gamma_\lambda$ of input substring $h(y_{k_j+1} \dots y_{k_{j+1}}) \in \Sigma^*$ the contents of register after *two* transitions of automaton A which starts with register value $z \in \{0, 1\}$ and reads two input symbols $h(y_{k_j} y_{k_j+1})$ where $y_{k_j} \in \Gamma_\sigma$. Thus, the image $b'(\Gamma) = B' = B \cup \{0, 1\}$ of an extended mapping $b' : \Gamma \longrightarrow B'$ used in (34) includes the initial register values $0, 1$ in addition to the shift function values $\Delta_r(q, x)$ for $q \in Q$, $x \in \Sigma$, and $r \in \{1, \dots, p\}$, according to (23).

Since all the endpoints of intervals $I_1, \dots, I_p$ are assumed to be a-quasi-periodic within B', it follows from Theorem 4 that

[4] A direct construction of an ordinary finite automaton that simulates a NN1A satisfying the assumption of Theorem 5 was presented already in [17, Theorem 3]. However, the construction was based on a stronger definition of quasi-periodicity assuming a bounded length of quasi-repetends (cf. Example 1). Moreover, the characterization of FAR languages in Theorem 2 simplifies the proof of Theorem 5 substantially.

L_r is a regular language for every $r = 1, \ldots, p$, because regular languages are closed under complementation, intersection, and difference. Furthermore, regular languages are known to be closed under concatenation, union, Kleene star, and string homomorphism. In addition, if S is regular, then its largest prefix-closed subset S^{Pref} is also regular as a corresponding finite automaton A_1 recognizing $S = L(A_1)$ can be reduced to A_2 such that $S^{Pref} = L(A_2)$, by eliminating all the non-final states in A_1. Thus, $\mathcal{L}$ in (13) is regular and it follows from Theorem 2 that language $L = L(N)$ is regular. ■

VI. The Computational Power of NN1A

In this section we present a lower and upper bound on the computational power of NN1A. In particular, we show in Theorem 7 that they are languages accepted by NN1A which are not context-free while Theorem 9 proves any NN1A language to be context-sensitive.

Example 2: We first present an example of numbers c that are not a-quasi-periodic within B. Let $B = \{0, 1\}$ and assume that $a = \alpha_1/\alpha_2 \in \mathbb{Q}$, $c = \gamma_1/\gamma_2 \in \mathbb{Q}$ are irreducible fractions where $\alpha_1, \alpha_2, \gamma_1, \gamma_2 \in \mathbb{N}$ and $c < 1$, such that $\alpha_1\gamma_2$ and $\alpha_2\gamma_1$ are coprime. Suppose that $c = \sum_{k=0}^{\infty} b_k a^k$ with $b_k \in B$ for all $k \geq 0$, and denote by $0 < k_1 < k_2 < \cdots$ all the indices such that $b_{k_i} = 1$ for $i \geq 1$, which satisfies

$$c_n = \sum_{k=0}^{\infty} b_{k_n+k} a^k - 1 = \frac{c - \sum_{i=1}^{n} a^{k_i}}{a^{k_n}} \tag{37}$$

for $n \geq 1$. By plugging $a = \alpha_1/\alpha_2$, $c = \gamma_1/\gamma_2$ into (37), we obtain

$$c_n = \frac{\gamma_1 \alpha_2^{k_n} - \gamma_2 \alpha_1 \sum_{i=1}^{n} \alpha_1^{k_i - 1} \alpha_2^{k_n - k_i}}{\gamma_2 \alpha_1^{k_n}} \tag{38}$$

which is an irreducible fraction since both γ_2 and α_1 are not factors of $\gamma_1\alpha_2$. Hence, for any natural n_1, n_2 such that $0 < n_1 < n_2$ we know $c_{n_1} \neq c_{n_2}$, which implies that $\sum_{k=0}^{\infty} b_k a^k$ is not an eventually quasi-periodic series [18, Theorem 3].

Furthermore, Theorem 6 states that there are non-context-free cut languages while Lemma 3 proves that the cut languages can in fact be recognized by FAR.

Theorem 6: [18, Theorem 13] Let $B \subset \mathbb{Q}$, Γ is a finite alphabet, and $b : \Gamma \longrightarrow B$ is a mapping. If c is not a-quasi-periodic within B, then the cut language $L_{<c}$ over Γ is not context-free.

Lemma 3: Let Γ be a finite alphabet, $b : \Gamma \longrightarrow B$ is a mapping, and $c \in \mathbb{Q}$. In addition, assume

$$\mu = \inf_{\substack{y_1, \ldots, y_k \in \Gamma \\ k \geq 0}} \sum_{i=0}^{k-1} b(y_{k-i}) a^i \geq 0 \,. \tag{39}$$

Then language $L = L_{<c} \cdot \Gamma$ where $L_{<c}$ is a cut language over alphabet Γ, can be recognized by a finite automaton with a register.

Proof: Denote

$$\nu = \sup_{\substack{y_1, \ldots, y_k \in \Gamma \\ k \geq 0}} \sum_{i=0}^{k-1} b(y_{k-i}) a^i \tag{40}$$

which is finite due to $|a| < 1$. Further assume $0 \leq \mu < c \leq \nu$ since otherwise $L_{<c} = \emptyset$ or $L_{<c} = \Gamma^*$ which can trivially be recognized by FAR.

We introduce a finite automaton with a register, $A = (Q, \Gamma, \{I_1, I_2\}, a, (\Delta_1, \Delta_2), \delta, q_1, z_0, F)$, that recognizes language $L(A) = L_{<c} \cdot \Gamma$. In particular, $Q = \{q_1, q_2\}$, $I_1 = [0, c/\nu)$, $I_2 = [c/\nu, 1]$, $z_0 = 0$, and $F = \{q_1\}$. Moreover, we define $\delta_r : Q \times \Sigma \longrightarrow Q$, and $\Delta_r : Q \times \Sigma \longrightarrow \mathbb{Q}$, for $r \in \{1, 2\}$, so that

$$\delta_r(q, y) = q_r \tag{41}$$

$$\Delta_r(q, y) = \frac{b(y)}{\nu} \,, \tag{42}$$

for any $q \in Q$ and $y \in \Gamma$.

It follows from (42) that automaton A, after reading an input string $y_1 \ldots y_n \in \Gamma^n$, stores $z_n = \sum_{i=0}^{n-1} b(y_{n-i}) a^i/\nu$ in its register, which satisfies $0 \leq z_k \leq 1$ for every $k = 0, \ldots, n$. Thus, $y_1 \ldots y_n \in L_{<c}$ iff $z_n \in I_1$ iff A moves to final state $q_1 \in F$ via δ_1 defined by (41), while reading an extra input symbol $\gamma \in \Gamma$, that is, $y_1 \ldots y_n \gamma \in L(A)$. Hence, A accepts $L_{<c} \cdot \Gamma$. ■

The preceding results are summarized in the following theorem:

Theorem 7: There is a language L accepted by a neural network with an analog unit, which is not context-free.

Proof: Example 2 ensures that any power series $\sum_{k=0}^{\infty} b_k a^k = c = 5/7 < 1$ with $b_k \in B = \{0, 1\}$ for all $k \geq 0$, is not $(2/3)$-quasi-periodic, since the greatest common divisor of $2 \cdot 7$ and $3 \cdot 5$ is 1. Let $\Gamma = B$ and $b : \Gamma \longrightarrow B$ be the identity. According to Theorem 6, the cut language $L_{<c}$ over alphabet Γ is not context-free. It follows that the same holds for $L = L_{<c} \cdot \Gamma$ since $L_{<c} = \{\mathbf{y} \in \Gamma^* \,|\, \mathbf{y}0 \in L\}$ and the context-free languages are closed under GSM (generalized sequential machine) mapping. On the other hand, L can be recognized by FAR according to Lemma 3, because (39) follows from $a > 0$ and $b(y) = y \geq 0$ for $y \in \Gamma = \{0, 1\}$. Hence, Theorem 1 guarantees that the non-context-free language L can be recognized by NN1A. ■

On the other hand, Theorem 9 provides an upper bound on the computational power of NN1A which follows from the fact that the cut languages are context-sensitive for rational thresholds.

Theorem 8: [18, Theorem 15] Every cut language $L_{<c}$ with threshold $c \in \mathbb{Q}$ is context-sensitive.

Theorem 9: Any language accepted by a neural network with an analog unit is context-sensitive.

Proof: The argument is analogous to the proof of Theorem 5. In particular, Theorem 1 is employed to transform a given NN1A, N, to a computationally equivalent FAR, A, so that $L = L(N) = L(A)$, while Theorem 2 reduces L to (12). Since context-sensitive languages are closed under complementation, intersection, and difference, Theorem 8 ensures that L_r used in (12) is a context-sensitive[5] for every $r = 1, \ldots, p$.

[5]Theorem 8 assumes formally that $0 < |a| < 1$ which, in general, need not be met for $a = w_{ss}$ from (8). Nevertheless, the proof of this theorem in [18, Theorem 15] is valid for any $a \in \mathbb{Q}$.

Furthermore, context-sensitive languages are known to be closed under concatenation, union, Kleene star, and ε-free homomorphism. In addition, if S is context-sensitive, then its largest prefix-closed subset S^{Pref} is also context-sensitive as a nondeterministic linear bounded automaton (LBA) M^{Pref} that recognizes $S^{Pref} = L(M^{Pref})$ runs successively LBA M for $S = L(M)$ on every prefix of an input which can be stored within linear space, and S^{Pref} accepts if all these runs of M are accepting computations. Thus, it follows from (12) and (13) that language L is context-sensitive. ■

VII. Conclusion

In this paper we have characterized the class of languages that are accepted by binary-state neural networks with an extra analog unit, which is an intermediate computational model between neural networks with integer weights, corresponding to finite automata, and that with rational weights which are Turing universal. By using this characterization we have shown that the computational power of such networks is between context-sensitive and context-free languages. In addition, we have formulated a sufficient condition when these networks accept only regular languages in terms of quasi-periodicity of their weight parameters. The question of whether this condition is also necessary remains open.

Another challenge for further research is to generalize the result to other domains of the feedback weight w_{ss} associated with analog unit s such as $w_{ss} \in \mathbb{R}$ or $|w_{ss}| > 1$. Moreover, it would be very interesting to study the possibility of having a proper hierarchy of classes NN1As of increasing complexity, according to the nature of their unique rational weights (as it is done in the real-weighted analog case, for weights of increasing Kolmogorov complexity [16]).

Acknowledgment

Research was done with institutional support RVO: 67985807 and partially supported by the grant of the Czech Science Foundation No. P202/12/G061.

References

[1] P. Koiran, "A family of universal recurrent networks," *Theoretical Computer Science*, vol. 168, no. 2, pp. 473–480, 1996.

[2] H. T. Siegelmann, "Recurrent neural networks and finite automata," *Journal of Computational Intelligence*, vol. 12, no. 4, pp. 567–574, 1996.

[3] J. Šíma, "Analog stable simulation of discrete neural networks," *Neural Network World*, vol. 7, no. 6, pp. 679–686, 1997.

[4] M. Šorel and J. Šíma, "Robust RBF finite automata," *Neurocomputing*, vol. 62, pp. 93–110, 2004.

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

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

[7] J. Šíma and P. Orponen, "General-purpose computation with neural networks: A survey of complexity theoretic results," *Neural Computation*, vol. 15, no. 12, pp. 2727–2778, 2003.

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

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

[10] P. Indyk, "Optimal simulation of automata by neural nets," in *Proceedings of the STACS 1995 Twelfth Annual Symposium on Theoretical Aspects of Computer Science*, ser. LNCS, vol. 900, 1995, pp. 337–348.

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

[12] J. Šíma, "Energy complexity of recurrent neural networks," *Neural Computation*, vol. 26, no. 5, pp. 953–973, 2014.

[13] J. Šíma and J. Wiedermann, "Theory of neuromata," *Journal of the ACM*, vol. 45, no. 1, pp. 155–178, 1998.

[14] H. T. Siegelmann and E. D. Sontag, "On the computational power of neural nets," *Journal of Computer System Science*, vol. 50, no. 1, pp. 132–150, 1995.

[15] ——, "Analog computation via neural networks," *Theoretical Computer Science*, vol. 131, no. 2, pp. 331–360, 1994.

[16] J. L. Balcázar, R. Gavaldà, and H. T. Siegelmann, "Computational power of neural networks: A characterization in terms of Kolmogorov complexity," *IEEE Transactions on Information Theory*, vol. 43, no. 4, pp. 1175–1183, 1997.

[17] J. Šíma, "The power of extra analog neuron," in *Proceedings of the TPNC 2014 Third International Conference on Theory and Practice of Natural Computing*, ser. LNCS, vol. 8890, 2014, pp. 243–254.

[18] J. Šíma and P. Savický, "Cut languages in rational bases," in *Proceedings of the LATA 2017 Eleventh International Conference on Language and Automata Theory and Applications*, ser. LNCS, vol. 10168, 2017.

[19] J.-P. Allouche, M. Clarke, and N. Sidorov, "Periodic unique beta-expansions: The Sharkovskiĭ ordering," *Ergodic Theory and Dynamical Systems*, vol. 29, no. 4, pp. 1055–1074, 2009.

[20] S. Baker, "On small bases which admit countably many expansions," *Journal of Number Theory*, vol. 147, pp. 515–532, 2015.

[21] S. Baker, Z. Masáková, E. Pelantová, and T. Vávra, "On periodic representations in non-Pisot bases," arXiv:1604.03354v1 [math.NT], 2016.

[22] D. Dombek, Z. Masáková, and E. Pelantová, "Number representation using generalized $(-\beta)$-transformation," *Theoretical Computer Science*, vol. 412, no. 48, pp. 6653–6665, 2011.

[23] P. Erdös, I. Joó, and V. Komornik, "Characterization of the unique expansions $1 = \sum_{i=1}^{\infty} q^{-n_i}$ and related problems," *Bulletin de la Société Mathématique de France*, vol. 118, no. 3, pp. 377–390, 1990.

[24] P. Glendinning and N. Sidorov, "Unique representations of real numbers in non-integer bases," *Mathematical Research Letters*, vol. 8, no. 4, pp. 535–543, 2001.

[25] K. G. Hare, "Beta-expansions of Pisot and Salem numbers," in *Proceedings of the Waterloo Workshop in Computer Algebra 2006: Latest Advances in Symbolic Algorithms*. World Scientific, 2007, pp. 67–84.

[26] W. Parry, "On the β-expansions of real numbers," *Acta Mathematica Hungarica*, vol. 11, no. 3, pp. 401–416, 1960.

[27] M. Pedicini, "Greedy expansions and sets with deleted digits," *Theoretical Computer Science*, vol. 332, no. 1-3, pp. 313–336, 2005.

[28] A. Rényi, "Representations for real numbers and their ergodic properties," *Acta Mathematica Academiae Scientiarum Hungaricae*, vol. 8, no. 3-4, pp. 477–493, 1957.

[29] K. Schmidt, "On periodic expansions of Pisot numbers and Salem numbers," *Bulletin of the London Mathematical Society*, vol. 12, no. 4, pp. 269–278, 1980.

[30] N. Sidorov, "Almost every number has a continuum of β-expansions," *The American Mathematical Monthly*, vol. 110, no. 9, pp. 838–842, 2003.

[31] ——, "Expansions in non-integer bases: Lower, middle and top orders," *Journal of Number Theory*, vol. 129, no. 4, pp. 741–754, 2009.

[32] P. Orponen, "Computing with truly asynchronous threshold logic networks," *Theoretical Computer Science*, vol. 174, no. 1-2, pp. 123–136, 1997.

[33] O. H. Ibarra, S. Sahni, and C. E. Kim, "Finite automata with multiplication," *Theoretical Computer Science*, vol. 2, no. 3, pp. 271–294, 1976.