Towards Logical Representations of Recurrent Neural Networks

Sandro Preto[1]

[1]Center for Mathematics, Computing and Cognition
Federal University of ABC – Santo André – SP – Brazil

sandro.preto@ufabc.edu.br

Abstract. *Representations of neural networks in the formal language of logical systems may be used to enhance their interpretability and as a step in the formal verification of their properties. This work builds on the literature regarding the representation of feedforward neural networks that compute rational McNaughton functions in the language of Łukasiewicz logic. Thus, we propose a technique for representing recurrent computations, enabling the representation of recurrent neural networks. This is an initial investigation conducted within the specific context of simple recurrent networks (SRNs) whose layers compute rational McNaughton functions.*

1. Introduction

The logical representation of neural networks is a step toward advancing the understanding of these artificial intelligence systems. Formalizing a network within a logical framework can enhance the interpretability of the model. Furthermore, such formalization might be used in the formal verification of neural networks, improving the security of these models when performing critical tasks.

Techniques for representing neural networks within the language of Łukasiewicz logic are already discussed in the literature. Specifically, feedforward neural networks that compute rational McNaughton functions have been explored [Preto and Finger 2024, Preto and Finger 2022, Preto and Finger 2020]. These representations were used to formally encode and verify properties of binary classification neural networks through the framework of Łukasiewicz logic [Preto and Finger 2023a, Preto and Finger 2023b, Preto et al. 2023].

The aim of this work is to propose a path for advancing this line of research towards logical representations of recurrent neural networks. To achieve this, we introduce a mechanism for representing the computation of recurrent layers within Łukasiewicz logic. This technique is explored in the context of *simple recurrent networks* whose layers compute rational McNaughton functions.

The rest of this work is organized as follows. In Section 2, we introduce simple recurrent networks. In Section 3, we introduce Łukasiewicz logic and functional representations as a means of representing neural networks. In Section 4, we propose a technique for representing recurrent computations and simple recurrent networks. Finally, in Section 5, we provide some final remarks.

2. Recurrent Neural Networks

While *neural networks* are computational models defined by connections between their computing units—commonly organized in layers—, *recurrent neural networks* are a specific type of these models that include cycles in their connections. These cycles enable the model to establish context when processing sequential data, which is essential for tasks involving data such as text. In this work, we focus on a simple kind of recurrent neural networks called *Elman networks* or *simple recurrent networks (SRN)* [Elman 1990]. The following presentation of SRNs is based on [Jurafsky and Martin 2025, Chapter 8].

Let the input data for an SRN, such as written text or recorded speech, be represented through (column) vectors in $\mathbb{R}^{d_{in}}$. These vectors might be, for instance, word embeddings, which are representations of words in a natural language. Then, a sequence of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbb{R}^{d_{in}}$ might stand for a sequence of words in a text excerpt. An SRN processes sequences of vectors, each element of a sequence one at a time, from the first to the last. The further along a vector is in the sequence, the more context the SRN is expected to accumulate. The computation path of an SRN, depicted in Figure 1, comprehends:

- An input layer, that stores the input data given by a vector $\mathbf{x}_t \in \mathbb{R}^{d_{in}}$ in a sequence of vectors;
- A hidden layer, that receives as input both the data in the input layer and its own previous output through a recurrent connection; and
- An output layer, that receives as input the output from the hidden layer.

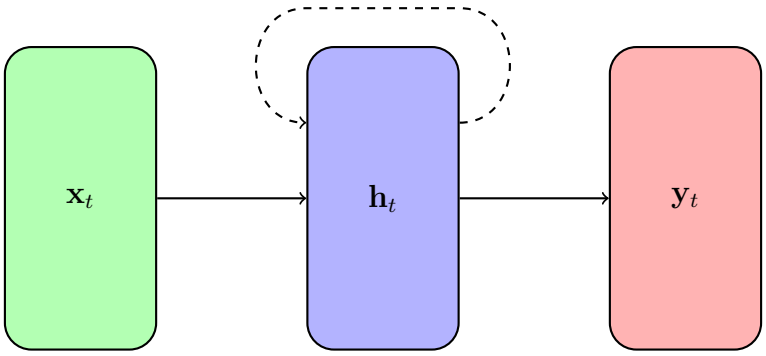

Figure 1. The computation path in an SRN

An SRN is determined by matrices $\mathbf{W} \in \mathbb{R}^{d_h \times d_{in}}$, $\mathbf{U} \in \mathbb{R}^{d_h \times d_h}$ and $\mathbf{V} \in \mathbb{R}^{d_{out} \times d_h}$, where d_{in} is its input dimension, d_{out} is its output dimension and d_h is its hidden layer dimension. Let $\mathbf{x}_t \in \mathbb{R}^{d_{in}}$ be an element of a sequence of vectors that is given as input to the SRN; the computation of this element by the SRN is given by equations:

$$\mathbf{h}_t = g(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1}); \tag{1}$$

$$\mathbf{y}_t = f(\mathbf{V}\mathbf{h}_t). \tag{2}$$

Equation (1) describes the computation in the hidden layer and equation (2) describes the computation in the output layer.

Vector $\mathbf{h}_{t-1} \in \mathbb{R}^{d_h}$ is the output of the hidden layer from the computation of the previous element $\mathbf{x}_{t-1}$ in the sequence, and is reused as input for the hidden layer in

the computation of the current element $\mathbf{x}_t$. This mechanism provides context acquired from $\mathbf{x}_1$ up to $\mathbf{x}_{t-1}$ when processing $\mathbf{x}_t$. Then, $\mathbf{h}_t$ is the output of the hidden layer in the processing of $\mathbf{x}_t$; it is used to calculate the SRN output $\mathbf{y}_t$, through the output layer, and is given as input for the hidden layer when processing a possible next element $\mathbf{x}_{t+1}$ in the sequence.

Note that, for processing the first element $\mathbf{x}_1$ in a sequence of vectors, a predefined standard initial vector $\mathbf{h}_0$ is necessary; in this work, we assume it to be the zero vector. Also note that, in addition to linear operations between matrices and vectors, nonlinearity is introduced in (1) and (2) through the *activation functions* f and g. Function g must map vectors in $\mathbb{R}^{d_h}$ to vectors still in $\mathbb{R}^{d_h}$.

In this work, we consider the particular case where activation functions f and g transform an argument vector by applying the *truncated identity function* $\mathrm{TId} : \mathbb{R} \to \mathbb{R}$ to each of its entries, where $\mathrm{TId}(x) = \max(0, \min(1, x))$. We also assume that vectors $\mathbf{x}_t$ in an input sequence are constrained to $[0, 1]^{d_{in}} \subseteq \mathbb{R}^{d_{in}}$.

As an example of application, assume that the elements of an input sequence $\mathbf{x}_1, \ldots, \mathbf{x}_k \in [0, 1]^{d_{in}}$ are word embeddings of a text. Then, each output $\mathbf{y}_t$, for $t \in \{1, \ldots, k\}$, might classify the text encoded by the subsequence $\mathbf{x}_1, \ldots, \mathbf{x}_t$ into d_{out} topics, for instance, such as news, sports, technology, etc. The final output $\mathbf{y}_k$ determines a classification for the entire encoding sequence. For more details on recurrent neural networks, including applications and training, we refer the reader to [Jurafsky and Martin 2025].

In order to achieve our aim of representing an SRN by logical formulas, it is convenient to rewrite equation (1) as follows:

$$\mathbf{h}_t = g\left(\begin{bmatrix}\mathbf{W} & \mathbf{U}\end{bmatrix}\begin{bmatrix}\mathbf{x}_t \\ \mathbf{h}_{t-1}\end{bmatrix}\right). \tag{3}$$

Note that equations (1) and (3) are equivalent.

3. Łukasiewicz Logic for Functional and Neural Representations

Łukasiewicz logic (Ł) is defined from a basic language $\mathcal{L}$ containing freely generated formulas from a countable infinite set of propositional variables $\mathbb{P}$, a binary disjunction operator ($\oplus$) and a unary negation operator ($\neg$). Let us denote propositional variables by uppercase Latin letters, possibly with subscripts and superscripts.

A Ł-valuation is a function $v : \mathcal{L} \to [0, 1]$ that satisfies, for $\varphi, \psi \in \mathcal{L}$:

$$v(\varphi \oplus \psi) = \min(1, v(\varphi) + v(\psi)); \tag{4}$$
$$v(\neg\varphi) = 1 - v(\varphi). \tag{5}$$

Let $\mathbf{Val}$ be the set of all valuations. A formula $\varphi \in \mathcal{L}$ is *satisfiable* if there exists a Ł-valuation $v \in \mathbf{Val}$ such that $v(\varphi) = 1$; a set of formulas $\Phi \subseteq \mathcal{L}$ is *satisfiable* if there exists a Ł-valuation $v \in \mathbf{Val}$ such that $v(\varphi) = 1$, for all $\varphi \in \Phi$; otherwise (sets of) formulas are *unsatisfiable*. Let $\mathbf{Val}_\Phi$ be the set of all Ł-valuations $v \in \mathbf{Val}$ that satisfy a set of formulas $\Phi \subseteq \mathcal{L}$; we call such a restricted set of valuations a *semantics modulo satisfiability*. Note that $\mathbf{Val}_\varnothing = \mathbf{Val}$.

For $\varphi, \psi \in \mathcal{L}$, derived operators from disjunction and negation are defined as follows.

Conjunction: $\varphi \odot \psi \doteq \neg(\neg\varphi \oplus \neg\psi)$	$v(\varphi \odot \psi) = \max(0, v(\varphi) + v(\psi) - 1)$
Implication: $\varphi \to \psi \doteq \neg\varphi \oplus \psi$	$v(\varphi \to \psi) = \min(1, 1 - v(\varphi) + v(\psi))$
Maximum: $\varphi \vee \psi \doteq \neg(\neg\varphi \oplus \psi) \oplus \psi$	$v(\varphi \vee \psi) = \max(v(\varphi), v(\psi))$
Minimum: $\varphi \wedge \psi \doteq \neg(\neg\varphi \vee \neg\psi)$	$v(\varphi \wedge \psi) = \min(v(\varphi), v(\psi))$
Bi-implication: $\varphi \leftrightarrow \psi \doteq (\varphi \to \psi) \wedge (\psi \to \varphi)$	$v(\varphi \leftrightarrow \psi) = 1 - \lvert v(\varphi) - v(\psi)\rvert$
Truncated subtraction: $\varphi \ominus \psi \doteq \neg(\varphi \to \psi)$	$v(\varphi \ominus \psi) = \max(0, v(\varphi) - v(\psi))$

Note that $v(\varphi \to \psi) = 1$ iff $v(\varphi) \leq v(\psi)$ and, similarly, $v(\varphi \leftrightarrow \psi) = 1$ iff $v(\varphi) = v(\psi)$. Let X be a propositional variable; as we have $v(X \odot \neg X) = 0$, for any $v \in \mathbf{Val}$, we define constant $\mathbf{0}$ by $X \odot \neg X$. We also define $0\varphi \doteq \mathbf{0}$ and $n\varphi \doteq \varphi \oplus \cdots \oplus \varphi$, n times, for $n \in \mathbb{N}^*$; and $\bigoplus_{i \in \varnothing} \varphi_i \doteq \mathbf{0}$.

A *rational McNaughton function* $f : [0,1]^n \to [0,1]$ is a function that satisfies:

- f is continuous with respect to the usual topology of $[0,1]$ real number interval;
- There are linear functions $p_1, \ldots, p_m$ over $[0,1]^n$ with rational coefficients such that, for each point $\mathbf{x} \in [0,1]^n$, there is an index $i \in \{1, \ldots, m\}$ with $f(\mathbf{x}) = p_i(\mathbf{x})$. Polynomials $p_1, \ldots, p_m$ are the *linear pieces* of f.

A *McNaughton function* is a rational McNaughton function whose linear pieces coefficients are constrained to integer values.

Formulas of Łukasiewicz logic represent McNaughton functions. That is, for a formula $\varphi \in \mathcal{L}$ whose propositional variables are $X_1, \ldots, X_n \in \mathbb{P}$, function $f_\varphi : [0,1]^n \to [0,1]$, given by

$$f_\varphi(x_1, \ldots, x_n) = v(\varphi), \tag{6}$$

for a Ł-valuation $v \in \mathbf{Val}$ such that $v(X_1) = x_1, \ldots, v(X_n) = x_n$, is a McNaughton function. Conversely, for any McNaughton function $f : [0,1]^n \to [0,1]$, there is a formula $\varphi_f \in \mathcal{L}$ with n distinct propositional variables, which may be identified with $X_1, \ldots, X_n$, satisfying an expression analogous to (6). Such identification of formulas and McNaughton functions is established by the McNaughton's Theorem [McNaughton 1951, Mundici 1994].

Although formulas of Ł only represent (integer) McNaughton functions, an implicit kind of representation may be used to overcome such constraint [Finger and Preto 2020, Preto and Finger 2022, Preto and Finger 2020]. Let $f : [0,1]^n \to [0,1]$ be a function, $\varphi \in \mathcal{L}$ a formula and $\Phi \subseteq \mathcal{L}$ a set of formulas. The pair $\langle \varphi, \Phi \rangle$ is a *representation (modulo satisfiability)* of f if:

- For all $\langle x_1, \ldots, x_n \rangle \in [0,1]^n$, there exists a Ł-valuation $v \in \mathbf{Val}_\Phi$, such that $v(X_\iota) = x_\iota$, for $\iota = 1, \ldots, n$; and
- $f(v(X_1), \ldots, v(X_n)) = v(\varphi)$, for all Ł-valuation $v \in \mathbf{Val}_\Phi$.

Every rational McNaughton function has a representation modulo satisfiability [Preto and Finger 2022]. From this point onward, whenever we refer to function representations in this text, it is understood that we are speaking of representations modulo satisfiability.

For example, constant functions $f : [0,1]^n \to [0,1]$, given by $f(\mathbf{x}) = c$, where $c \in [0,1] \cap \mathbb{Q}$, are rational McNaughton functions. Let $c = 1/b$, where $b \in \mathbb{N}^*$. Formula

$$\varphi_{1/b} \doteq \neg(b-1)Z_{1/b} \leftrightarrow Z_{1/b}$$

has the property that any Ł-valuation v satisfying $\varphi_{1/b}$ is such that $v(Z_{1/b}) = 1/b$. Then, the pair $\langle Z_{1/b}, \{\varphi_{1/b}\}\rangle$ is a representation of f. In case $c = 0$, the pair $\langle \mathbf{0}, \varnothing\rangle$ represents f. And, in case $c = a/b$, where $a \in \mathbb{N}$, $b \in \mathbb{N}^*$ and $a \leq b$, the pair $\langle aZ_{1/b}, \{\varphi_{1/b}\}\rangle$ is a representation of f.

Neural networks may be represented in Łukasiewicz logic by encoding the functions they compute. Techniques exist for constructing such representations for certain non-recurrent (feedforward) neural networks that compute rational McNaughton functions [Preto and Finger 2024].

Both the hidden and output layers of an SRN compute rational McNaughton functions if matrices $\mathbf{W}$, $\mathbf{U}$ and $\mathbf{V}$ have rational entries, their inputs are restricted to vectors in $[0,1]^{d_{in}}$ and activation functions f and g are elementwise applications of function TId, as agreed upon in the previous section. Let us refer to such a neural network as a *McNaughton simple recurrent network (McN-SRN)*.

In this case, by Equation (3), each of the d_h entries in the output vector $\mathbf{h}_t$ of the hidden layer is given by a rational McNaughton function of the input vector $[\mathbf{x}_t^T \ \mathbf{h}_{t-1}^T]^T \in [0,1]^{d_{in}+d_h}$, where T is the transposition operator; let $\langle \varphi_i, \Phi_i\rangle$ be the representation of each such function, for $i \in \{1, \ldots, d_h\}$. And, by Equation (2), each of the d_{out} entries in the output vector $\mathbf{y}_t$ of the output layer is given by a rational McNaughton function of the vector $\mathbf{h}_t \in [0,1]^{d_h}$; let $\langle \psi_j, \Psi_j\rangle$ be the representation of each such function, for $j \in \{1, \ldots, d_{out}\}$.

For example, consider a McN-SRN, where $d_{in} = d_h = 2$ and $d_{out} = 1$, given by:

$$\mathbf{W} = \begin{bmatrix} 0 & -1 \\ 1/2 & 0 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{V} = \begin{bmatrix} 1 & 1 \end{bmatrix}. \tag{7}$$

The first and second entries in the output $\mathbf{h}_t \in [0,1]^2$ of the hidden layer are given by rational McNaughton functions $f_1, f_2 : [0,1]^4 \to [0,1]$, respectively defined by:

$$f_1(x_1, x_2, x_3, x_4) = \max(\ 0,\ \min(1,\ -x_2 + x_3)\ );$$
$$f_2(x_1, x_2, x_3, x_4) = \max(\ 0,\ \min(1,\ x_1/2 + x_4)\ ).$$

And the output $\mathbf{y}_t \in [0,1]$ of the output layer is given by rational McNaughton function $g : [0,1]^2 \to [0,1]$, defined by:

$$g(x_1, x_2) = \max(\ 0,\ \min(1,\ x_1 + x_2)\ ).$$

Functions f_1, f_2 and g are represented, respectively, by $\langle \varphi_1, \Phi_1\rangle$, $\langle \varphi_2, \Phi_2\rangle$ and $\langle \psi, \Psi\rangle$, which are defined in the following.

$$\begin{array}{ll} \varphi_1 \doteq X_3 \ominus X_2 & \Phi_1 \doteq \varnothing \\ \varphi_2 \doteq Z \oplus X_4 & \Phi_2 \doteq \{X_1 \leftrightarrow Z \oplus Z, \quad Z_{1/2} \leftrightarrow \neg Z_{1/2}, \quad Z \to Z_{1/2}\} \\ \psi \doteq X_1 \oplus X_2 & \Psi \doteq \varnothing \end{array}$$

4. Representing Recurrent Neural Networks

In the previous section, we saw that the hidden and output layers of the class of McN-SRNs (whose layers compute rational McNaughton functions) may be represented in Łukasiewicz logic. We now turn to the problem of constructing logical representations for the entire neural networks.

Let a McN-SRN be given by the matrices $\mathbf{W} \in \mathbb{Q}^{d_h \times d_{in}}$, $\mathbf{U} \in \mathbb{Q}^{d_h \times d_h}$ and $\mathbf{V} \in \mathbb{Q}^{d_{out} \times d_h}$. Following the previous section, let the representations of the rational McNaughton functions computed by each entry in the output vector of the hidden layer be given by $\langle \varphi_i, \Phi_i \rangle$, for $i \in \{1, \ldots, d_h\}$. And let the representations of the rational McNaughton functions computed by each entry in the output vector of the output layer be given by $\langle \psi_j, \Psi_j \rangle$, for $j \in \{1, \ldots, d_{out}\}$.

Let $\mathbf{x}_1, \ldots, \mathbf{x}_k \in [0,1]^{d_{in}}$ be an input sequence of vectors to the McN-SRN, and $\mathbf{y}_k \in [0,1]^{d_{out}}$ its output relative to $\mathbf{x}_k$. Also, let $X_1^t, \ldots, X_{d_{in}}^t, X_{d_{in}+1}^t, \ldots, X_{d_{in}+d_h}^t \in \mathbb{P}$ be distinguished propositional variables, for $t \in \{1, \ldots, k\}$. Our aim is to establish, for each $k \in \mathbb{N}^*$ and $j \in \{1, \ldots, d_{out}\}$, a representation $\langle \xi_j^k, \Xi_j^k \rangle$ such that, for any Ł-valuation $v \in \mathbf{Val}_{\Xi_j^k}$, if

$$\begin{aligned} v(X_1^1) = \mathbf{x}_{1,1}, \quad \ldots, \quad v(X_{d_{in}}^1) = \mathbf{x}_{1,d_{in}}, \\ \vdots \\ v(X_1^k) = \mathbf{x}_{k,1}, \quad \ldots, \quad v(X_{d_{in}}^k) = \mathbf{x}_{k,d_{in}}, \end{aligned}$$

then,

$$v(\xi_j^k) = \mathbf{y}_{k,j}.$$

Thus, representing a McN-SRN means to represent functions $f_j^k : [0,1]^{k \cdot d_{in}} \to [0,1]$ by pairs $\langle \xi_j^k, \Xi_j^k \rangle$, for sizes $k \in \mathbb{N}^*$ of input sequences of vectors and indices $j \in \{1, \ldots, d_{out}\}$ of the McN-SRN output vector. For each such k and j, function f_j^k has:

- As arguments, the $k \cdot d_{in}$ entries in the k vectors in an input sequence for the McN-SRN; and
- As output, the j-th value of the McN-SRN output vector.

To proceed with the construction, we represent the recurrent computation performed by McN-SRNs. For that, for each $i \in \{1, \ldots, d_h\}$, define k new representations $\langle \varphi_i^t, \Phi_i^t \rangle$, for $t \in \{1, \ldots, k\}$, by replacing all propositional variables in $\langle \varphi_i, \Phi_i \rangle$ with fresh ones, ensuring that no propositional variable is shared among the k new representations. Let us agree that, in a pair $\langle \varphi_i^t, \Phi_i^t \rangle$, propositional variables $X_\iota^t \in \mathbb{P}$ are the replacements of $X_\iota \in \mathbb{P}$ in the original functional representation $\langle \varphi_i, \Phi_i \rangle$, for $\iota \in \{1, \ldots, d_{in} + d_h\}$. Pairs $\langle \varphi_i^t, \Phi_i^t \rangle$ are meant to represent computations in distinct recurrences of the SRN, for $t \in \{1, \ldots, k\}$.

Moreover, we need to ensure that, for fixed $t \in \{1, \ldots, k\}$, the only shared variables among pairs $\langle \varphi_i^t, \Phi_i^t \rangle$, for $i \in \{1, \ldots, d_h\}$, are $X_1^t, \ldots, X_{d_{in}}^t, X_{d_{in}+1}^t, \ldots, X_{d_{in}+d_h}^t \in \mathbb{P}$. Also, if $i' \neq i''$ and $t' \neq t''$, $\langle \varphi_{i'}^{t'}, \Phi_{i'}^{t'} \rangle$ does not share any propositional variable with $\langle \varphi_{i''}^{t''}, \Phi_{i''}^{t''} \rangle$. As a last agreement, pairs $\langle \psi_j, \Psi_j \rangle$, for $j \in \{1, \ldots, d_{out}\}$, only share propositional variables $X_1, \ldots, X_{d_h}$ among themselves and do not share any propositional variable with any pair $\langle \varphi_i^t, \Phi_i^t \rangle$, whatever are i and t.

Then, we are able to define representation $\langle \xi_j^k, \Xi_j^k \rangle$ by

$$\xi_j^k \doteq \psi_j$$

and

$$\Xi_j^k \doteq \bigcup_{i=1}^{d_h} \bigcup_{t=1}^{k} \Phi_i^t \cup \Psi_j \cup \bigcup_{\iota=d_{in}+1}^{d_{in}+d_h} \Big\{ X_\iota^1 \leftrightarrow \mathbf{0} \Big\} \cup \bigcup_{t=2}^{k} \bigcup_{\iota=d_{in}+1}^{d_{in}+d_h} \Big\{ X_\iota^t \leftrightarrow \varphi_{\iota-d_{in}}^{t-1} \Big\} \cup \bigcup_{\iota=1}^{d_h} \Big\{ X_\iota \leftrightarrow \varphi_\iota^k \Big\}.$$

A representation for the neural network given in previous section by matrices (7), for $k = 2$, is established by pair $\langle \xi^k, \Xi^k \rangle$, where

$$\xi^k \doteq X_1 \oplus X_2$$

and

$$\begin{aligned} \Xi^k \doteq \Big\{ & X_1^1 \leftrightarrow Z^1 \oplus Z^1, \quad Z_{1/2}^1 \leftrightarrow \neg Z_{1/2}^1, \quad Z^1 \to Z_{1/2}^1, \\ & X_1^2 \leftrightarrow Z^2 \oplus Z^2, \quad Z_{1/2}^2 \leftrightarrow \neg Z_{1/2}^2, \quad Z^2 \to Z_{1/2}^2, \\ & X_3^1 \leftrightarrow \mathbf{0}, \quad X_4^1 \leftrightarrow \mathbf{0}, \\ & X_3^2 \leftrightarrow X_3^1 \ominus X_2^1, \quad X_4^2 \leftrightarrow Z^1 \oplus X_4^1, \\ & X_1 \leftrightarrow X_3^2 \ominus X_2^2, \quad X_2 \leftrightarrow Z^2 \oplus X_4^2 \Big\}. \end{aligned}$$

5. Final Remarks

In this work, we delineated the first steps towards the logical representations of recurrent neural networks. We devised techniques to represent recurrent computations from representations of layer computations in the language of Łukasiewicz logic. These techniques were applied to the specific case of a class of simple recurrent networks (SRN), which we called McNaughton simple recurrent networks (McN-SRN).

In order to achieve more general techniques for representing recurrent neural networks in Łukasiewicz logic, many extensions of the approach presented here might be explored in future research. One may pursue techniques to account for activation functions other than the TId function aiming at representations of layers that compute functions beyond rational McNaughton functions. Ultimately, this line of research might culminate in the logical representation of recurrent neural networks widely used today, such as long short-term memory (LSTM) and gated recurrent unit (GRU) neural networks [Hochreiter and Schmidhuber 1997, Cho et al. 2014].

The logical representation of recurrent neural networks is part of a broader research effort to develop formal representations of neural networks in general. Among the potential applications of this endeavor, we highlight the formal verification of neural networks. Thus, another direction for future research is to identify critical properties of recurrent neural networks, such as specific features desired in natural language models, and formalize these properties in the language of Łukasiewicz logic.

Funding

This work was carried out at the Center for Artificial Intelligence (C4AI-USP), with support by the São Paulo Research Foundation (FAPESP) [grant #2019/07665-4] and by the IBM Corporation.

References

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211.

Finger, M. and Preto, S. (2020). Probably partially true: Satisfiability for Łukasiewicz infinitely-valued probabilistic logic and related topics. *Journal of Automated Reasoning*, 64(7):1269–1286.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Jurafsky, D. and Martin, J. H. (2025). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd edition. Online manuscript released January 12, 2025.

McNaughton, R. (1951). A theorem about infinite-valued sentential logic. *Journal of Symbolic Logic*, 16:1–13.

Mundici, D. (1994). A constructive proof of McNaughton's theorem in infinite-valued logic. *The Journal of Symbolic Logic*, 59(2):596–602.

Preto, S. and Finger, M. (2020). An efficient algorithm for representing piecewise linear functions into logic. *Electronic Notes in Theoretical Computer Science*, 351:167–186. Proceedings of LSFA 2020, the 15th International Workshop on Logical and Semantic Frameworks, with Applications (LSFA 2020).

Preto, S. and Finger, M. (2022). Efficient representation of piecewise linear functions into Łukasiewicz logic modulo satisfiability. *Mathematical Structures in Computer Science*, 32(9):1119–1144.

Preto, S. and Finger, M. (2023a). Effective reasoning over neural networks using Łukasiewicz logic. In Hitzler, P., Kamruzzaman Sarker, M., and Eberhart, A., editors, *Compendium of Neurosymbolic Artificial Intelligence*, volume 369 of *Frontiers in Artificial Intelligence and Applications*, chapter 28, pages 609–630. IOS Press.

Preto, S. and Finger, M. (2023b). Proving properties of binary classification neural networks via Łukasiewicz logic. *Logic Journal of the IGPL*, 31(5):805–821.

Preto, S. and Finger, M. (To appear in the proceedings of LSFA 2024). Regional, lattice and logical representations of neural networks.

Preto, S., Manyà, F., and Finger, M. (2023). Benchmarking Łukasiewicz logic solvers with properties of neural networks. In *2023 IEEE 53rd International Symposium on Multiple-Valued Logic (ISMVL)*, pages 158–163.