Extracting Formulae in Many-Valued Logic from Deep Neural Networks

Yani Zhang and Helmut Bölcskei
Chair for Mathematical Information Science, ETH Zürich
yanizhang@mins.ee.ethz.ch, hboelcskei@ethz.ch

Abstract

We propose a new perspective on deep ReLU networks, namely as circuit counterparts of Łukasiewicz infinite-valued logic—a many-valued (MV) generalization of Boolean logic. An algorithm for extracting formulae in MV logic from deep ReLU networks is presented[1]. As the algorithm applies to networks with general, in particular also real-valued, weights, it can be used to extract logical formulae from deep ReLU networks trained on data.

1 Introduction

State-of-the-art deep neural networks exhibit impressive reasoning capabilities, e.g. in mathematical tasks [1], program synthesis [2], and algorithmic reasoning [3]. This paper reports an attempt at systematically connecting neural networks with mathematical logic. Specifically, we shall be interested in reading out logical formulae from (trained) deep neural networks.

Let us first take a step back. Consider a neural network that realizes a map $f : [0,1]^n \to [0,1]$. When the input and output variables take on two possible values only, say 0 and 1, f reduces to a Boolean function and can hence be studied by means of Boolean algebra, see e.g. [4]. Boolean functions can be realized by Boolean circuits [5]. The idea of using Boolean algebra to analyze and design Boolean circuits dates back to [6, 7] and most prominently by Shannon in [8]. Specifically, this correspondence works as follows. Given a Boolean circuit, one can deduce a Boolean algebraic expression that realizes the circuit's input-output relation. Conversely, for a given Boolean algebraic expression, it is possible to specify a Boolean circuit whose input-output relation equals this expression.

The main aim of the present paper is to initiate the development of a generalization of this correspondence from Boolean functions $f : \{0,1\}^n \to \{0,1\}$ to general functions $f : [0,1]^n \to [0,1]$. This immediately leads to the following two questions:

1. What is the logical system replacing Boolean logic?
2. What is the counterpart of Boolean circuits?

As to the first question, we shall show that the theory of infinite-valued Łukasiewicz logic [9] provides a suitable framework for characterizing general (nonbinary) functions $f : [0,1]^n \to [0,1]$ from a logical perspective. With slight abuse of terminology, we shall refer to infinite-valued Łukasiewicz logic as many-valued (MV) logic throughout the paper. Based on a fundamental result [10], which characterizes the class of truth functions in MV logic—also called McNaughton functions—as continuous piecewise linear functions with integer coefficients, we show that neural networks employing the ReLU nonlinearity $\rho(x) = \max\{0, x\}$ and integer weights[2] naturally implement statements in MV logic. This answers the second question above by identifying ReLU networks as the counterpart of Boolean circuits.

In practice, trained neural networks will, however, not exhibit integer weights, unless this is explicitly enforced in the training process. Extensions of MV logic, namely Rational Łukasiewicz logic [11] and $\mathbb{R}\mathcal{L}$ [12], have truth functions that are again continuous piecewise linear, but with rational and

[1] A Python implementation is available at https://www.mins.ee.ethz.ch/research/downloads/NN2MV.html.
[2] By weights, we mean the entries of the weight matrices and bias vectors associated with the network.

real-valued coefficients, respectively. Such functions are likewise naturally realized by ReLU networks, but correspondingly with rational and real-valued weights.

Besides the conceptual contribution residing in the systematic development of the connection between ReLU networks and MV logic along with its extensions, we also devise an algorithm for extracting logical formulae from (trained) ReLU networks with integer, rational, or real-valued weights. For pedagogical reasons and to render the presentation more accessible, we first present the entire framework for MV logic and ReLU networks with integer weights, and then provide extensions to the rational and real case. In addition, we carry out a detailed comparison between our algorithm and the only two constructive procedures for converting McNaughton functions to their associated MV logical formulae available in the literature [13, 14].

The overall philosophy of viewing ReLU networks as the circuit counterpart of MV logic and its extensions is inspired by [15, 16, 17]. Specifically, Amato et al. [15, 16] pointed out that neural networks, with the clipped ReLU (CReLU) nonlinearity $\sigma(x) = \min\{1, \max\{0, x\}\}$ and rational weights, realize truth functions in Rational Łukasiewicz logic. Di Nola et al. [17] proved that CReLU networks with real weights realize truth functions in $\mathbb{R}\mathcal{L}$ logic. The universal correspondence between ReLU networks and MV logic as well as its extensions reported here along with the algorithm for extracting logical formulae from ReLU networks appear to be new.

Finally, we refer to [18, 19, 20] for different frameworks that integrate neural learning with symbolic logic. Deep neural networks have been successfully applied to symbolic analysis through training on numerical data [21] or by leveraging large scale pre-training [22].

We start with a brief review of the correspondence between Boolean logic and Boolean circuits as described in [8, 23].

1.1 Boolean Logic and Boolean Circuits

Boolean algebra on the binary set $\{0, 1\}$ consists of the application of the logical operations `OR`, `AND`, `NOT`, denoted by $\oplus, \odot$, and $\neg$, respectively, to propositional variables. The algebra is fully characterized through the following identities:

$$\begin{aligned}
&x \oplus 0 = x && x \odot 1 = x \\
&x \oplus \neg x = 1 && x \odot \neg x = 0 \\
&x \oplus y = y \oplus x && x \odot y = y \odot x \\
&(x \oplus y) \oplus z = x \oplus (y \oplus z) && \\
&(x \odot y) \odot z = x \odot (y \odot z) && \\
&x \oplus (y \odot z) = (x \oplus y) \odot (x \oplus z) && \\
&x \odot (y \oplus z) = (x \odot y) \oplus (x \odot z) &&
\end{aligned} \tag{1}$$

Note that one can define the operation $\odot$ in terms of $\oplus$ and $\neg$ according to $x \odot y := \neg(\neg x \oplus \neg y)$ and then rewrite all the identities in (1) involving $\odot$ based on $\oplus$ and $\neg$ only.

Shannon developed a systematic approach to the analysis and synthesis of switching circuits using Boolean algebra [8]. At the heart of Shannon's theory is the following interpretation of Boolean logic in terms of switching circuits. A propositional (Boolean) variable x is interpreted as a make contact on a switch. The negation of x represents a break contact. The constants 0 and 1 signify open and closed circuits, respectively. The Boolean operations $\oplus$ and $\odot$ correspond to parallel and, respectively, series connections of switches. With these correspondences, every switching circuit can be associated with a Boolean formula, namely by identifying the switches in the circuit with Boolean variables, and then connecting them by $\oplus$ and $\odot$ operations according to the connections appearing in the circuit. Conversely, following these correspondences, Boolean circuits are directly associated with Boolean formulae. Exploiting these equivalences, the manipulation of Boolean circuits can be carried out on their corresponding Boolean expressions and vice versa.

Boolean circuits are represented by directed acyclic graphs whose nodes are logic gates $\oplus, \odot$, and $\neg$. For example, the circuit in Figure 1(a) computes the Boolean function

$$(x_1 \odot x_2) \oplus ((x_2 \oplus x_3) \odot (x_2 \odot x_3)). \tag{2}$$

The expression in (2) can be manipulated using the identities listed in (1) to arrive at the functionally

Figure 1: Two equivalent Boolean circuits.

equivalent, but algebraically simpler, expression $x_2 \odot (x_1 \oplus x_3)$, which corresponds to the smaller (only 2 logic gates instead of 5) Boolean circuit depicted in Figure 1(b).

Shannon's seminal paper [23] showed that the circuit complexity, in terms of the number of switches, of n-ary Boolean functions is upper-bounded by $O(2^n/n)$ and that almost all Boolean functions have circuit complexity close to this bound.

The motivation for studying Boolean circuits is that the time required to compute a given Boolean function on a Turing machine is closely related to the function's circuit complexity [24]. For example, Pippenger and Fischer [25] proved that a language of time complexity $T(n)$ has circuit complexity $O(T(n) \log T(n))$. Conversely, polynomial circuit complexity implies nonuniform polynomial time complexity.

1.2 MV Logic

We now generalize from Boolean logic to many-valued logic [26], where propositional variables take truth values in the interval $[0, 1]$. The corresponding algebraic counterpart is known as Chang's MV algebra [27]. MV logic consists of two logical operations, $\oplus$ and $\neg$, in analogy to the Boolean `OR` and `NOT`; the operation $\odot$, in analogy to the Boolean `AND`, is defined in terms of $\oplus$ and $\neg$ according to $x \odot y := \neg(\neg x \oplus \neg y)$. This leads us to the definition of the so-called standard MV algebra.

Definition 1.1. *Consider the unit interval $[0, 1]$, and define $x \oplus y = \min\{1, x + y\}$ and $\neg x = 1 - x$, for $x, y \in [0, 1]$. It can be verified that the structure $\mathcal{I} = \langle [0, 1], \oplus, \neg, 0\rangle$ is an MV algebra [9]. In particular, $\mathcal{I}$ constitutes the algebraic counterpart of Łukasiewicz infinite-valued logic [27]. We further define the operation $x \odot y := \neg(\neg x \oplus \neg y) = \max\{0, x + y - 1\}$.*

It can be shown that the Boolean algebra $\mathcal{B} := \{\{0, 1\}, \texttt{OR}, \texttt{NOT}, 0\}$ is a special case of MV algebras. The MV algebra $\mathcal{I}$ in 1.1 is referred to as standard because an equation holds in every MV algebra iff it holds in $\mathcal{I}$ [27, 28]. Additional relevant material on MV algebras is provided in Appendix A.

2 ReLU Networks as Circuit Counterpart of MV Logic

MV terms are finite strings composed of propositional variables $x_1, x_2, \ldots$ connected by $\oplus$, $\odot$, and $\neg$ operations and brackets $()$, such as $(x_1 \oplus \neg x_2) \odot x_3$. Term functions are the corresponding truth functions obtained by interpreting the logical operations according to how they are specified in the concrete MV algebra used, e.g. $x \oplus y = \min\{1, x + y\}$ in the standard MV algebra $\mathcal{I}$. See Definitions A.2 and A.3. In the Boolean algebra $\mathcal{B}$, term functions are binary tables $\{\{0, 1\}^n \to \{0, 1\} : n \in \mathbb{N}\}$. In the standard MV algebra $\mathcal{I}$, term functions are characterized by continuous piecewise linear functions with integer coefficients as formalized by the McNaughton theorem [10].

Theorem 2.1. [10] *Consider the MV algebra $\mathcal{I}$ in Definition 1.1. Let $n \in \mathbb{N}$. For a function $f : [0, 1]^n \to [0, 1]$ to have a corresponding MV term τ such that the associated term function $\tau^{\mathcal{I}}$ satisfies $\tau^{\mathcal{I}} = f$ on $[0, 1]^n$, it is necessary and sufficient that f satisfy the following conditions:*

1. *f is continuous with respect to the natural topology on $[0, 1]^n$,*
2. *there exist linear polynomials $p_1, \ldots, p_\ell$ with integer coefficients, i.e.,*

$$p_j(x_1, \ldots, x_n) = m_{j1}x_1 + \cdots + m_{jn}x_n + b_j,$$

for $j = 1, \ldots, \ell$, with $m_{j1}, \ldots, m_{jn}, b_j \in \mathbb{Z}$, such that for every $x \in [0,1]^n$, there is a $j \in \{1, \ldots, \ell\}$ with $f(x) = p_j(x)$.

Functions satisfying these conditions are called McNaughton functions.

ReLU networks (see Definition B.1) are compositions of affine transformations and the ReLU nonlinearity $\rho(x) = \max\{0, x\}$ (applied element-wise) and as such realize continuous piecewise linear functions. Specifically, the class of ReLU networks with integer weights is equivalent to the class of formulae in MV logic. The corresponding formal statement is as follows.

Theorem 2.2. *For $n \in \mathbb{N}$, let $\tau(x_1, \ldots, x_n)$ be an MV term in n variables with $\tau^{\mathcal{I}} : [0,1]^n \to [0,1]$ the associated term function in $\mathcal{I}$. There exists a ReLU network Φ with integer weights, satisfying*

$$\Phi(x_1, \ldots, x_n) = \tau^{\mathcal{I}}(x_1, \ldots, x_n),$$

for all $(x_1, \ldots, x_n) \in [0,1]^n$. Conversely, for every ReLU network $\Phi : [0,1]^n \to [0,1]$ with integer weights, there exists an MV term $\tau(x_1, \ldots, x_n)$ whose associated term function in $\mathcal{I}$ satisfies

$$\tau^{\mathcal{I}}(x_1, \ldots, x_n) = \Phi(x_1, \ldots, x_n),$$

for all $(x_1, \ldots, x_n) \in [0,1]^n$.

The remainder of this section is devoted to the proof of Theorem 2.2, along with the development of an algorithm for extracting formulae in MV logic from ReLU networks with integer weights. First, we show how, for a given MV term τ, a ReLU network with integer weights realizing the associated term function $\tau^{\mathcal{I}}$ can be constructed. We start by noting that the operation $\neg x = 1 - x$, by virtue of being affine, is trivially realized by a ReLU network. Further, there exist ReLU networks $\Phi^{\oplus}$ and $\Phi^{\odot}$, with integer weights, realizing the $\oplus$ and $\odot$ operations in $\mathcal{I}$, i.e.,

$$\begin{aligned}\Phi^{\oplus}(x, y) &= \min\{1, x + y\}\\ \Phi^{\odot}(x, y) &= \max\{0, x + y - 1\},\end{aligned}$$

for all $x, y \in [0,1]$. Detailed constructions of $\Phi^{\oplus}$ and $\Phi^{\odot}$ are provided in Lemma B.3. According to Lemma B.2 [29], compositions of ReLU networks are again ReLU networks. The ReLU network realizing the term function associated with the MV term τ can hence be obtained by concatenating ReLU networks implementing the operations $\oplus, \odot$, and $\neg$ as they appear in the expression for τ. What is more, inspection of the proof of Lemma B.2 reveals that the integer-valued nature of the weights is preserved in the process of composition. Therefore, the resulting overall ReLU network has integer weights.

For example, applying the procedure just outlined (see Appendix E for details) to the simple example $\tau = (x \oplus x) \odot \neg y$ yields the associated ReLU network

$$\Phi^{\tau} = W_3 \circ \rho \circ W_2 \circ \rho \circ W_1, \tag{3}$$

where

$$\begin{aligned}W_1(x, y) &= \begin{pmatrix} -2 & 0 \\ 0 & 1 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad x, y \in \mathbb{R},\\ W_2(x) &= \begin{pmatrix} -1 & -1 & 1 \end{pmatrix} x + 1, \quad x \in \mathbb{R}^3,\\ W_3(x) &= x, \quad x \in \mathbb{R}.\end{aligned}$$

The proof of the converse statement in Theorem 2.2 will be effected in a constructive manner, in the process developing an algorithm for extracting the MV logical formula corresponding to a given ReLU network with integer weights. The algorithm consists of the following three steps.

Step 1: Transform the ReLU network (with integer weights) into an equivalent network with the CReLU nonlinearity $\sigma(x) = \min\{1, \max\{0, x\}\}$. This is done by exploiting the fact that the domain of the ReLU network is the unit cube $[0,1]^n$ and, consequently, with finite-valued weights, the input of each layer in the network is bounded. Concretely, we replace each ρ-neuron by one or multiple σ-neurons, while retaining the integer-valued nature of the weights. For example, $\rho(x) = \sigma(x)$, for $x \in [0,1]$, and $\rho(x) = \sigma(x) + \sigma(x - 1)$, for $x \in [0,2]$.

Step 2: Extract MV terms from individual σ-neurons, which are of the form

$$\sigma(m_1x_1 + \cdots + m_nx_n + b),$$

with $m_1, \ldots, m_n, b \in \mathbb{Z}$. The following lemma provides an inductive way for accomplishing this.

Lemma 2.3. [30, 13] *Consider the function* $f(x_1, \ldots, x_n) = m_1x_1 + \cdots + m_nx_n + b$, $(x_1, \ldots, x_n) \in [0,1]^n$, *with* $m_1, \ldots, m_n, b \in \mathbb{Z}$. *Without loss of generality, assume that* $\max_{i=1}^n |m_i| = m_1$. *Let* $f_\circ(x_1, \ldots, x_n) = (m_1 - 1)x_1 + m_2x_2 + \cdots + m_nx_n + b$. *Then,*

$$\sigma(f) = (\sigma(f_\circ) \oplus x_1) \odot \sigma(f_\circ + 1). \tag{4}$$

Before proceeding to the next step, we demonstrate the application of Lemma 2.3 by way of the simple example $\sigma(x_1 - x_2 + x_3 - 1)$. First, we eliminate the variable x_1 according to

$$\sigma(x_1 - x_2 + x_3 - 1) = (\sigma(-x_2 + x_3 - 1) \oplus x_1) \odot \sigma(-x_2 + x_3). \tag{5}$$

Next, we remove the x_3-terms inside $\sigma(\cdot)$,

$$\sigma(-x_2 + x_3 - 1) = (\sigma(-x_2 - 1) \oplus x_3) \odot \sigma(-x_2) \tag{6}$$
$$\sigma(-x_2 + x_3) = (\sigma(-x_2) \oplus x_3) \odot \sigma(-x_2 + 1). \tag{7}$$

We note that $\sigma(-x_2) = 0$, for $x_2 \in [0,1]$. Owing to $x \odot 0 = 0$, for $x \in [0,1]$, (6) reduces to $\sigma(-x_2 + x_3 - 1) = 0$. Likewise, in (7) $\sigma(-x_2) \oplus x_3 = x_3$. We can then further simplify (7) according to

$$x_3 \odot \sigma(-x_2 + 1) = x_3 \odot (1 - \sigma(x_2)) \tag{8}$$
$$= x_3 \odot \neg x_2, \tag{9}$$

where in (8) we used $\sigma(x) = 1 - \sigma(-x + 1)$, for $x \in \mathbb{R}$, and (9) is by $\sigma(x) = x$ and $\neg x = 1 - x$, both for $x \in [0,1]$. Substituting the simplified results of (6) and (7) back into (5), we obtain the MV term corresponding to $\sigma(x_1 - x_2 + x_3 - 1)$ as $x_1 \odot (x_3 \odot \neg x_2)$.

Step 3: Compose the MV terms corresponding to the individual σ-neurons according to the layered structure of the CReLU network to get the MV term associated with the ReLU network. To illustrate this step, suppose that the neurons $\sigma^{(1)}(\cdot)$ and $\sigma^{(2)}(\cdot)$ have associated MV terms $\tau^{(1)}$ and $\tau^{(2)}$, respectively, and a third neuron $\sigma^{(3)}(m_1x_1 + m_2x_2 + b)$ has associated MV term $\tau^{(3)}(x_1, x_2)$. The MV term corresponding to the CReLU network $\sigma^{(3)}(m_1\sigma^{(1)} + m_2\sigma^{(2)} + b)$ is obtained by replacing all occurrences of x_1 in $\tau^{(3)}$ by $\tau^{(1)}$ and all occurrences of x_2 by $\tau^{(2)}$. This finalizes the proof of Theorem 2.2.

The essence of the proof of Theorem 2.2 resides in a strong algebraic property shared by the standard MV algebra $\mathcal{I}$ and ReLU networks. Concretely, compositions of ReLU networks with integer weights again yield ReLU networks with integer weights, and compositions of formulae in MV logic result in formulae in MV logic. As we shall see in Section 4, the parallelism between logical formulae and ReLU networks identified here extends to the cases of ReLU networks with rational weights and Rational Łukasiewicz logic as well as ReLU networks with real weights and $\mathbb{RL}$.

We hasten to add that the extracted formula associated with a given McNaughton function is not unique. Different algebraic expressions can exist, but they must be functionally equivalent as they all represent the same truth function in $\mathcal{I}$.

3 Aspects of Our Algorithm

Recalling that ReLU networks (with integer weights) realize continuous piecewise linear functions (with integer coefficients), the algorithm devised in the previous section can equivalently be seen as extracting an MV formula from a given McNaughton function f. We are aware of two other constructive extraction procedures, namely the Schauder hat method [13] and the hyperplane method [14]. The purpose of this section is to briefly review these two procedures and to compare them to our algorithm.

We start with the Schauder hat method and note that Schauder hats are functions of pyramidal shape supported on unions of simplices. This method starts with the construction of a simplicial complex over $[0,1]^n$ obtained by splitting the unit cube according to different permutations of the

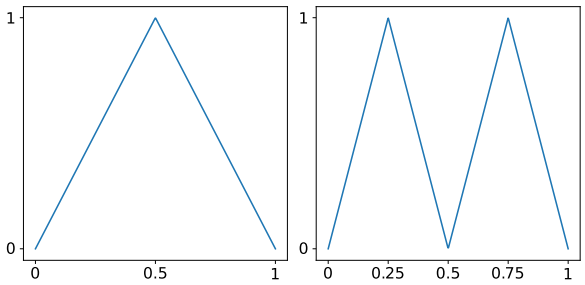

Figure 2: Left: the function g, right: the function g_2.

linear pieces of f. The resulting simplicial complex is further subdivided into a unimodular simplicial complex. Thanks to unimodularity, each Schauder hat can then be expressed in terms of "$\min$" and "$\max$" operations, which are realizable by the operations $\oplus, \odot$, and $\neg$ according to

$$\begin{aligned} \min\{x, y\} &= \neg(\neg x \odot y) \odot y := x \wedge y \\ \max\{x, y\} &= \neg(\neg x \oplus y) \oplus y := x \vee y. \end{aligned} \tag{10}$$

The overall MV term is finally constructed by combining the MV terms corresponding to the individual Schauder hats through the $\oplus$ operation. The reader is referred to [13] for a detailed account of the algorithm.

The hyperplane method [14] expresses f, with linear pieces $p_1, \ldots, p_\ell$, in terms of the truncated linear polynomials $\sigma(p_1), \ldots, \sigma(p_\ell)$, where $\sigma(x) = \min\{1, \max\{0, x\}\}$, for $x \in \mathbb{R}$, according to

$$f = \min_I \max_J \sigma(p_i), \tag{11}$$

where $I, J \subset \{1, \ldots, \ell\}$ are index sets. Next, MV terms corresponding to $\sigma(p_1), \ldots, \sigma(p_\ell)$ are determined by repeated application of Lemma 2.3. These expressions are finally combined into an MV term based on (11) and (10).

The first difference between our algorithm and the two existing ones resides in the fact that we work with a ReLU network Φ that realizes the McNaughton function f, instead of starting from f itself. While this aspect might seem innocuous, it has important ramifications as discussed next. First, it is simple to describe a McNaughton function on the unit interval $[0, 1]$, but rather complicated to specify McNaughton functions on the multi-dimensional unit cube $[0, 1]^n$, for $n \geq 2$. Specifically, Mundici [13] showed that $f : [0, 1]^n \to [0, 1]$ can be specified by listing all its linear pieces $p_1, \ldots, p_\ell$ along with the indices of the linear pieces that f falls on at the rational points $(c_1/d_1, \ldots, c_n/d_n) \in [0, 1]^n$ with $0 < d_1, \ldots, d_n \leq (n + 1)(2na)^n$ and $c_1, \ldots, c_n \in \mathbb{Z}$. Here, a is the maximum absolute value of all coefficients of $p_1, \ldots, p_\ell$. In comparison, it is much easier to specify a ReLU network, namely by simply providing the associated affine maps. In particular, our approach also provides a parametrization of all valid McNaughton functions, simply by varying the integer weights across all ReLU networks that map $[0, 1]^n$ to $[0, 1]$. Note that this, of course, also entails varying network architectures, i.e., depth and the number of nodes in the individual layers (save for the input and output layers).

The second difference lies in the algebraic expressions obtained by these algorithms, notably in their complexity, as measured by the length of the formulae. We illustrate this aspect by way of examples, limiting ourselves to one-dimensional functions, for ease of exposition. Consider the hat function $g : [0, 1] \to [0, 1]$ in Figure 2,

$$\begin{aligned} g(x) &= \rho(2x) - 2\rho(2x - 1) + \rho(2x - 2) \\ &= \begin{cases} 2x, & 0 \leq x \leq \frac{1}{2} \\ 2 - 2x, & \frac{1}{2} < x \leq 1. \end{cases} \end{aligned}$$

The Schauder hat method. The unimodular simplicial complex delivered by the algorithm has vertices $\{0, 1/2, 1\}$. There is only one Schauder hat, which is centered at $x = 1/2$ and given by

$$h_{\frac{1}{2}}(x) = \begin{cases} x, & 0 \leq x \leq \frac{1}{2} \\ 1 - x, & \frac{1}{2} < x \leq 1. \end{cases}$$

The MV term corresponding to $h_{\frac{1}{2}}(x) = \min\{x, 1-x\}$, for $x \in [0,1]$, is given by $x \wedge \neg x$. As $g(x) = 2h_{\frac{1}{2}}(x)$, for $x \in [0,1]$, the MV term associated with g is

$$(x \wedge \neg x) \oplus (x \wedge \neg x). \tag{12}$$

The hyperplane method. Denote the two linear pieces of g by $p_1 : x \mapsto 2x$ and $p_2 : x \mapsto 2-2x$. Iterative application of Lemma 2.3 produces the MV term corresponding to $\sigma(p_1)$ as $x \oplus x$, and that associated with $\sigma(p_2)$ as $\neg x \oplus \neg x$. Direct inspection of Figure 2 shows that $g(x) = \min\{p_1(x), p_2(x)\}$, for $x \in [0,1]$. Therefore, the overall MV term corresponding to g is given by

$$(x \oplus x) \wedge (\neg x \oplus \neg x). \tag{13}$$

Our algorithm. We first note that g can be realized by a ReLU network Φ_g according to $\Phi_g = W_2 \circ \rho \circ W_1 = g$ with

$$W_1(x) = \begin{pmatrix} 2 \\ 2 \\ 2 \end{pmatrix} x - \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix}, \quad W_2(x) = \begin{pmatrix} 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

Next, we run our algorithm described in Steps 1-3 in the proof of Theorem 2.2 on Φ_g. First, Φ_g is converted into the equivalent CReLU network $\Psi_g = W_2^* \circ \sigma \circ W_1^* = \Phi_g$ with

$$W_1^*(x) = \begin{pmatrix} 2 \\ 2 \end{pmatrix} x - \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad W_2^*(x) = \begin{pmatrix} 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix},$$

where the individual σ-neurons have associated MV terms as follows:

$$\sigma(2x) : x \oplus x \tag{14}$$
$$\sigma(2x-1) : x \odot x \tag{15}$$
$$\sigma(x_1 - x_2) : x_1 \odot \neg x_2. \tag{16}$$

Replacing x_1 in (16) by (14) and x_2 by (15), the MV term corresponding to the function g is given by

$$(x \oplus x) \odot \neg(x \odot x). \tag{17}$$

Interestingly, the three algorithms produce different MV terms for the same McNaughton function. As one would expect, these terms can be shown to all be equivalent by manipulating their algebraic expressions using identities of MV algebras (listed in Definition A.1). We finally note that the complexity of the MV terms does not differ much. However, this changes fundamentally in the second example we consider, namely the composition of g with itself $g_2 := g \circ g$.

The Schauder hat method. Permutations of the linear pieces of g_2 divide the interval $[0,1]$ into four simplices, namely $\{[0, \frac{1}{4}], [\frac{1}{4}, \frac{1}{2}], [\frac{1}{2}, \frac{3}{4}], [\frac{3}{4}, 1]\}$. Here the simplices happen to coincide with the linear regions of g_2, but this is not generally the case. To get unimodularity, the simplices are further subdivided into a simplicial complex with the vertex set $\{0, \frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{3}{4}, 1\}$. Four Schauder hats are obtained, as depicted in Figure 3, and each of these hats has an associated MV term. For example, the term corresponding to the hat in the top left of Figure 3 is given by $(\neg(x \oplus x \oplus x)) \wedge x$. The overall MV term associated with g_2 is obtained by linking the MV terms corresponding to the individual Schauder hats through the $\oplus$ operation. The resulting algebraic expression is quite long, so we refrain from displaying it.

The hyperplane method. Denote the four linear pieces of g_2 by $p_1 : x \mapsto 4x$, $p_2 : x \mapsto -4x+2$, $p_3 : x \mapsto 4x-2$, and $p_4 : x \mapsto -4x+4$. The hyperplane method starts by ordering the linear pieces over every linear region of g_2:

$$\begin{aligned}
x \in [0, 1/4]: &\quad p_3(x) \leq \boldsymbol{p}_1(x) \leq p_2(x) \leq p_4(x) \\
x \in [1/4, 1/2]: &\quad p_3(x) \leq \boldsymbol{p}_2(x) \leq p_1(x) \leq p_4(x) \\
x \in [1/2, 3/4]: &\quad p_2(x) \leq \boldsymbol{p}_3(x) \leq p_4(x) \leq p_1(x) \\
x \in [3/4, 1]: &\quad p_2(x) \leq \boldsymbol{p}_4(x) \leq p_3(x) \leq p_1(x),
\end{aligned}$$

where the bold symbols indicate the linear piece that equals g_2 on the corresponding interval, e.g., $g_2(x) = p_1(x)$, for $x \in [0, \frac{1}{4}]$. Following [14], g_2 is then expressed in terms of the truncated linear pieces connected by $\wedge$ and $\vee$ according to

$$g_2 = (\sigma(p_3) \vee \sigma(p_1)) \wedge (\sigma(p_3) \vee \sigma(p_2)) \wedge (\sigma(p_2) \vee \sigma(p_3)) \wedge (\sigma(p_2) \vee \sigma(p_4)). \tag{18}$$

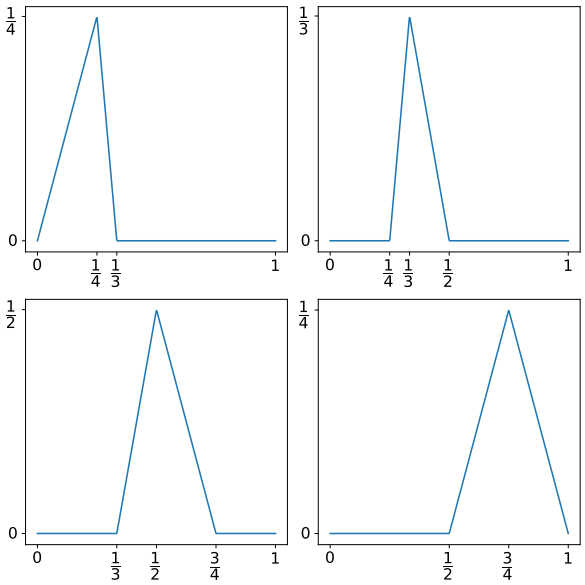

Figure 3: Schauder hats at vertices $\{\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{3}{4}\}$ associated with the function g_2.

Repeated application of Lemma 2.3 produces MV terms associated with each of the truncated linear pieces:

$$\begin{aligned}
&\sigma(p_1) : x \oplus x \oplus x \oplus x \\
&\sigma(p_2) : \neg(((((x \odot x) \oplus x) \odot (x \oplus x)) \oplus x) \odot (x \oplus x \oplus x)) \\
&\sigma(p_3) : ((x \odot x \odot x) \oplus x) \odot (((x \odot x) \oplus x) \odot (x \oplus x)) \\
&\sigma(p_4) : \neg(x \odot x \odot x \odot x).
\end{aligned} \tag{19}$$

Substituting (19) back into (18) yields the overall MV term associated with g_2, which, again, is too long to be shown here.

Our algorithm. As $g_2 = g \circ g$, the ReLU network realizing g_2 is simply $\Phi_g \circ \Phi_g$. The resulting MV term for g_2 is consequently given by the composition of (17) with itself, i.e., one replaces every occurrence of x in (17) by $(x \oplus x) \odot \neg(x \odot x)$. The final result is

$$(((x \oplus x) \odot \neg(x \odot x)) \oplus ((x \oplus x) \odot \neg(x \odot x))) \odot \neg(((x \oplus x) \odot \neg(x \odot x)) \odot ((x \oplus x) \odot \neg(x \odot x))), \tag{20}$$

which, while still somewhat unwieldy, is significantly shorter than the expressions obtained by the other two methods. Finally, we note that one can mix the results obtained by the different algorithms, e.g., for $g_2 = g \circ g$, we can also replace every occurrence of x in (17) by (12) or by (13), to obtain a valid MV term for g_2. We observe that our method delivers a shorter MV term as it exploits the compositional structure of $g_2 = g \circ g$, a property that is naturally present in deep ReLU network realizations of McNaughton functions. It is hence sensible to expect that our approach leads to shorter formulae whenever a "deep" ReLU network realization of the McNaughton function under consideration is available.

The next example serves to illustrate this aspect in a broader context. Specifically, we want to show how the nonuniqueness in ReLU network realizations, i.e., for a given function there are infinitely many ReLU networks that realize it, can be exploited by our algorithm to derive different algebraic expressions for the same truth function in MV logic. Considering a "shallow" realization of the function g_2 above according to

$$\Phi_g^2(x) = \rho(4x) - 2\rho(4x-1) + 2\rho(4x-2) - 2\rho(4x-3) + \rho(4x-4), \tag{21}$$

we first transform (21) into the equivalent CReLU network

$$\Psi_g^2 = W_2 \circ \sigma \circ W_1,$$

with

$$W_1(x) = \begin{pmatrix} 4 \\ 4 \\ 4 \\ 4 \end{pmatrix} x + \begin{pmatrix} 0 \\ -1 \\ -2 \\ -3 \end{pmatrix}, \quad x \in \mathbb{R},$$
$$W_2(x) = \begin{pmatrix} 1 & -1 & 1 & -1 \end{pmatrix} x, \quad x \in \mathbb{R}^4.$$

The MV term associated with the σ-neuron in the second layer of Ψ_g^2 is given by

$$\sigma(x_1 - x_2 + x_3 - x_4) : ((x_3 \odot \neg(x_2 \oplus x_4)) \oplus x_1) \odot ((\neg(x_2 \oplus x_4) \oplus x_3) \odot (\neg x_2 \oplus \neg x_4)), \tag{22}$$

and in the first layer we have

$$\sigma(4x) : x \oplus x \oplus x \oplus x \tag{23}$$
$$\sigma(4x - 1) : ((((x \odot x) \oplus x) \odot (x \oplus x)) \oplus x) \odot (x \oplus x \oplus x) \tag{24}$$
$$\sigma(4x - 2) : ((x \odot x \odot x) \oplus x) \odot (((x \odot x) \oplus x) \odot (x \oplus x)) \tag{25}$$
$$\sigma(4x - 3) : x \odot x \odot x \odot x. \tag{26}$$

Now, substituting (23) for x_1, (24) for x_2, (25) for x_3, and (26) for x_4 in (22), we obtain the MV term associated with Φ_g^2, which is too long to be displayed here, but formally has to be equivalent to (20). This illustrates that the deep network realization of $g_2 = g \circ g$ according to $\Phi_g \circ \Phi_g$ leads to an MV formula, namely (20), that is shorter than the one obtained here starting from the shallow realization (21).

The final example we consider is meant to illustrate how identifying redundant elements in ReLU network realizations can lead to shorter, yet formally equivalent, MV formulae. We shall approach this matter backwards, augmenting a given ReLU network realization by using the identity $x = \rho(x) - \rho(-x), x \in \mathbb{R}$. Specifically, consider the 2-dimensional function f realized by a ReLU network with two layers according to

$$f(x, y) = \rho(x + y) - \rho(y - x) - \rho(x + y - 1). \tag{27}$$

The associated MV term, obtained by running our algorithm, is

$$(x \oplus y) \odot \neg(y \odot \neg x).$$

Now, making use of $x = \rho(x) - \rho(-x), x \in \mathbb{R}$, as announced, we augment (27) to an equivalent ReLU network with three layers:

$$f(x, y) = \rho(\rho(x + y) - \rho(y - x) - \rho(x + y - 1)) - \rho(-\rho(x + y) + \rho(y - x) + \rho(x + y - 1)),$$

which, by running our algorithm, results in the longer, but formally equivalent, MV term

$$((x \oplus y) \odot \neg(\neg x \odot y)) \odot \neg(\neg(x \oplus y) \odot (\neg x \odot y)).$$

The last two examples show how one can effectively manipulate formulae in MV logic by manipulating their associated ReLU network realizations, extending the philosophy put forward by Shannon [8, 23] for the Boolean case to MV logic.

4 Extensions to the Rational and Real Cases

Extensions of MV logic, namely Rational Łukasiewicz logic and $\mathbb{R}\mathcal{L}$ logic, as mentioned in the introduction, have truth functions that are continuous piecewise linear, but with rational and, respectively, real coefficients.

4.1 The Rational Case

Rational Łukasiewicz logic extends MV logic by adding a division (by integers) operation. The algebraic counterpart is given by the so-called divisible many-valued (DMV) algebras [11].

Definition 4.1. *Consider the MV algebra $\mathcal{I}$ in Definition 1.1. Define the unary operations $\delta_i x = \frac{1}{i}x, x \in [0,1]$, for all $i \in \mathbb{N}$. The structure $\mathcal{I}_d = \langle [0,1], \oplus, \neg, \{\delta_i\}_{i\in\mathbb{N}}, 0\rangle$ is a DMV algebra [11].*

The class of term functions in $\mathcal{I}_\mathrm{d}$ is the class of continuous piecewise linear functions whose linear pieces have rational coefficients [11, 31]. In analogy to Theorem 2.1, such functions are called rational McNaughton functions. We next extend Theorem 2.2 to the rational case.

Theorem 4.2. *For $n \in \mathbb{N}$, let $\tau(x_1,\dots,x_n)$ be a DMV term in n variables and $\tau^{\mathcal{I}_d} : [0,1]^n \to [0,1]$ the associated term function in $\mathcal{I}_d$. There exists a ReLU network Φ with rational weights, satisfying*

$$\Phi(x_1,\dots,x_n) = \tau^{\mathcal{I}_d}(x_1,\dots,x_n),$$

for all $(x_1,\dots,x_n) \in [0,1]^n$. Conversely, for every ReLU network $\Phi : [0,1]^n \to [0,1]$ with rational weights, there exists a DMV term $\tau(x_1,\dots,x_n)$ whose associated term function in $\mathcal{I}_d$ satisfies

$$\tau^{\mathcal{I}_d}(x_1,\dots,x_n) = \Phi(x_1,\dots,x_n),$$

for all $(x_1,\dots,x_n) \in [0,1]^n$.

Proof. We already know how to realize the operations $\oplus, \odot$, and $\neg$ by ReLU networks, see Section 2. The division operation $\delta_i : x \to \frac{1}{i}x$, for $i \in \mathbb{N}$, by virtue of being affine, is trivially realized by a single-layer ReLU network. Following Lemma B.2, ReLU network realizations of formulae in Rational Łukasiewicz logic are obtained by concatenating ReLU networks implementing the operations $\oplus, \odot, \neg$, and $\{\delta_i\}_{i\in\mathbb{N}}$. Again, inspection of the proof of Lemma B.2 reveals that the resulting ReLU network has rational weights. We recover the algebraic property observed already for MV logic, namely compositions of ReLU networks with rational weights result in ReLU networks with rational weights, and compositions of DMV formulae again yield DMV formulae.

Next, we extend our algorithm, described in Section 2, to extract DMV terms from ReLU networks with rational weights. Steps 1 and 3 remain unaltered. We only need to modify Step 2 because the σ-neurons are now of the form

$$h = \sigma(m_1x_1 + \cdots + m_nx_n + b), \tag{28}$$

with $m_1,\dots,m_n,b \in \mathbb{Q}$, rendering Lemma 2.3 no longer applicable. Instead, we employ an idea by [31] to transform (28) into multiple copies of h which all have arguments with integer coefficients. Concretely, let $s \in \mathbb{N}$ be the least common multiple of the denominators of $m_1,\dots,m_n,b$. Recognizing that

$$s\sigma(x) = \sigma(sx) + \sigma(sx-1) + \cdots + \sigma(sx-(s-1)),$$

for $x \in \mathbb{R}$, and setting $h_i = \sigma(s(m_1x_1 + \cdots + m_nx_n + b) - i)$, it follows that $h = \sum_{i=0}^{s-1} \frac{1}{s}h_i$. As $h = \sum_{i=0}^{s-1} \frac{1}{s}h_i \le 1$, the DMV term associated with h is finally obtained according to $\oplus_{i=0}^{s-1}\delta_s\tau_i$. □

The Schauder hat and the hyperplane methods were extended to Rational Łukasiewicz logic in [11].

4.2 The Real Case

We finally turn to the case of ReLU networks with real coefficients, which is of particular practical interest as it allows to extract formulae from trained ReLU networks. The Riesz many-valued algebra (RMV) [12] extends the MV algebra in Definition 1.1 by adding the multiplication operation $\{\Delta_r : x \to rx,$ for $x \in [0,1]\}_{r\in[0,1]}$. The term functions of the corresponding logical system $\mathbb{R}\mathcal{L}$ are continuous piecewise linear functions with real coefficients [12], which can be realized by ReLU networks with real weights. Moreover, one gets ReLU networks realizing truth functions in $\mathbb{R}\mathcal{L}$ in the same manner as in the integer and rational cases, namely by composing ReLU networks realizing the logical operations appearing in the RMV formuale under consideration. Again, we have the algebraic property of the compositions of ReLU networks resulting in ReLU networks, while retaining the real-valued nature of the network weights, and compositions of RMV formulae yielding RMV formulae.

We now generalize our algorithm to extract RMV formulae from ReLU networks with real weights. Concretely, Steps 1 and 3 in Section 2 again remain unaltered. In Step 2, with each σ-neuron of the form

$$\sigma(m_1x_1 + \cdots + m_nx_n + b),$$

where $m_1, \ldots, m_n, b \in \mathbb{R}$, instead of Lemma 2.3, one applies the following result.

Lemma 4.3. [12] *Consider the function $f(x_1, \ldots, x_n) = m_1x_1 + \cdots + m_nx_n + b$, $(x_1, \ldots, x_n) \in [0,1]^n$, with $m_1, \ldots, m_n, b \in \mathbb{R}$. For all $m \in (0,1]$ and $i \in \{1, \ldots, n\}$, with $f_\circ(x_1, \ldots, x_n) = m_1x_1 + \cdots + m_{i-1}x_{i-1} + (m_i - m)x_i + m_{i+1}x_{i+1} + \cdots + m_nx_n + b$, it holds that*

$$\sigma(f) = (\sigma(f_\circ) \oplus (mx_i)) \odot \sigma(f_\circ + 1). \tag{29}$$

We demonstrate the application of Lemma 4.3 through a simple example. Consider the σ-neuron $\sigma\left(\frac{1}{\sqrt{2}}x_1 - 2x_2\right)$. First apply Lemma 4.3 with $m = \frac{1}{\sqrt{2}}$ and $i = 1$ to get

$$\sigma\left(\frac{1}{\sqrt{2}}x_1 - 2x_2\right) = \left(\sigma(-2x_2) \oplus \left(\frac{1}{\sqrt{2}}x_1\right)\right) \odot \sigma(-2x_2 + 1), \tag{30}$$

thereby eliminating x_1. As $\sigma(-2x_2) = 0$ and $0 \oplus x = x$, (30) reduces to $(\frac{1}{\sqrt{2}}x_1) \odot \sigma(-2x_2 + 1)$. Next, by applying Lemma 4.3 twice and using $\sigma(x) = 1 - \sigma(1 - x), x \in \mathbb{R}$, we obtain the RMV term associated with $\sigma(-2x_2 + 1)$ as $\neg(x_2 \oplus x_2)$. The RMV term corresponding to $\frac{1}{\sqrt{2}}x_1$ is given by $\Delta_{\frac{1}{\sqrt{2}}}x_1$. Therefore, the overall RMV term associated with $\sigma\left(\frac{1}{\sqrt{2}}x_1 - 2x_2\right)$ is $\Delta_{\frac{1}{\sqrt{2}}}x_1 \odot \neg(x_2 \oplus x_2)$.

The hyperplane method was extented to the real case in [12]. An extension of the Schauder hat method to the real case does not seem to be available in the literature, but can easily be devised. We will report it elsewhere.

We finally note that the conclusions on the differences between our algorithm and the other two algorithms as summarized in Section 2 carry over to the rational and real cases.

5 Acknowledgment

The authors are deeply grateful to Prof. Olivia Caramello for drawing their attention to the McNaughton theorem and, more generally, to MV logic.

References

[1] G. Lample and F. Charton, Deep learning for symbolic mathematics, arXiv preprint arXiv:1912.01412 (2019).

[2] S. Bubeck, V. Chandrasekaran, R. Eldan, et al., Sparks of artificial general intelligence: Early experiments with GPT-4, arXiv preprint arXiv:2303.12712 (2023).

[3] B. Liu, J. T. Ash, S. Goel, A. Krishnamurthy, and C. Zhang, Transformers learn shortcuts to automata, arXiv preprint arXiv:2210.10749 (2022).

[4] S. Skyum and L. G. Valiant, A complexity theory based on Boolean algebra, Journal of the ACM (JACM) 32 (1985) 484–502.

[5] R. B. Boppana and M. Sipser, The complexity of finite functions, in: Algorithms and Complexity, Elsevier, 1990, pp. 757–804.

[6] A. Nakashima, The theory of relay circuit composition, The Journal of the Institute of Telegraph and Telephone Engineers of Japan 38 (1935) 461–489.

[7] V. I. Shestakov, Some Mathematical Methods for the Construction and Simplification of Two-Terminal Electrical Networks of Class A, Phd thesis, The Lomonosov State University, Moscow, Russia, 1938.

[8] C. E. Shannon, A symbolic analysis of relay and switching circuits, Electrical Engineering 57 (1938) 713–723.

[9] R. Cignoli, I. d'Ottaviano, and D. Mundici, Algebraic Foundations of Many-Valued Reasoning, Springer, 2000.

[10] R. McNaughton, A theorem about infinite-valued sentential logic, The Journal of Symbolic Logic 16 (1951) 1–13.

[11] B. Gerla, Rational Łukasiewicz logic and DMV-algebras, Neural Network World 6 (2001).

[12] A. Di Nola and I. Leuştean, Łukasiewicz logic and Riesz spaces, Soft Computing 18 (2014) 2349–2363.

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

[14] S. Aguzzoli, Geometrical and Proof Theoretical Issues in Łukasiewicz Propositional Logics, Phd thesis, University of Siena, Italy, 1998.

[15] P. Amato, A. Di Nola, and B. Gerla, Neural networks and rational Łukasiewicz logic, in: Annual Meeting of the North American Fuzzy Information Processing Society Proceedings, 2002, pp. 506–510.

[16] P. Amato, A. Di Nola, and B. Gerla, Neural networks and rational McNaughton Functions, Journal of Multiple-Valued Logic and Soft Computing 11 (2005) 95–110.

[17] A. Di Nola, B. Gerla, and I. Leuştean, Adding real coefficients to Łukasiewicz logic: An application to neural networks, Fuzzy Logic and Applications: 10th International Workshop (2013) 77–85.

[18] G. G. Towell and J. W. Shavlik, Knowledge-based artificial neural networks, Artificial intelligence 70 (1994) 119–165.

[19] A. S. Avila Garcez and G. Zaverucha, The connectionist inductive learning and logic programming system, Applied Intelligence 11 (1999) 59–77.

[20] R. Riegel, A. Gray, F. Luus, et al., Logical neural networks, arXiv preprint arXiv:2006.13155 (2020).

[21] L. Biggio, T. Bendinelli, A. Lucchi, and G. Parascandolo, A seq2seq approach to symbolic regression, Learning Meets Combinatorial Algorithms at NeurIPS2020 (2020).

[22] L. Biggio, T. Bendinelli, A. Neitz, A. Lucchi, and G. Parascandolo, Neural symbolic regression that scales, in: International Conference on Machine Learning, 2021, pp. 936–945.

[23] C. E. Shannon, The synthesis of two-terminal switching circuits, The Bell System Technical Journal 28 (1949) 59–98.

[24] J. E. Savage, Computational work and time on finite machines, Journal of the ACM 19 (1972) 660–674.

[25] N. Pippenger and M. J. Fischer, Relations among complexity measures, Journal of the ACM 26 (1979) 361–381.

[26] A. Tarski, Logic, semantics, metamathematics: Papers from 1923 to 1938, Hackett Publishing, 1983.

[27] C. C. Chang, Algebraic analysis of many-valued logics, Transactions of the American Mathematical Society 88 (1958) 467–490.

[28] C. C. Chang, A new proof of the completeness of the Łukasiewicz axioms, Transactions of the American Mathematical Society 93 (1959) 74–80.

[29] D. Elbrächter, D. Perekrestenko, P. Grohs, and H. Bölcskei, Deep neural network approximation theory, IEEE Transactions on Information Theory 67 (2021) 2581–2623.

[30] A. Rose and J. B. Rosser, Fragments of many-valued statement calculi, Transactions of the American Mathematical Society 87 (1958) 1–53.

[31] M. Baaz and H. Veith, Interpolation in fuzzy logic, Archive for Mathematical Logic 38 (1999) 461–489.

A MV algebras

Definition A.1. [9] *A many-valued algebra is a structure $\mathcal{A} = \langle A, \oplus, \neg, 0\rangle$ consisting of a nonempty set A, a constant $0 \in A$, a binary operation $\oplus$, and a unary operation $\neg$ satisfying the following axioms:*

$$x \oplus (y \oplus z) = (x \oplus y) \oplus z \tag{31.1}$$
$$x \oplus y = y \oplus x$$
$$x \oplus 0 = x \tag{31.2}$$
$$\neg\neg x = x$$
$$x \oplus \neg 0 = \neg 0$$
$$\neg(\neg x \oplus y) \oplus y = \neg(\neg y \oplus x) \oplus x.$$

Specifically, (31.1)-(31.2) state that the structure $\langle A, \oplus, 0\rangle$ is an abelian monoid. An MV algebra $\langle A, \oplus, \neg, 0\rangle$ is said to be nontrivial iff A contains more than one element. In each MV algebra we can define a constant 1 and a binary operation $\odot$ as follows:

$$1 := \neg 0 \tag{32}$$
$$x \odot y := \neg(\neg x \oplus \neg y). \tag{33}$$

The ensuing identities are then direct consequences of Definition A.1:

$$x \odot (y \odot z) = (x \odot y) \odot z$$
$$x \odot y = y \odot x$$
$$x \odot 1 = x$$
$$x \odot 0 = 0.$$

We will frequently use the notions of MV terms and term functions formalized as follows.

Definition A.2. [9] *Let $n \in \mathbb{N}$ and $S_n = \{(,), 0, \neg, \oplus, x_1, \ldots, x_n\}$. An MV term in the variables $x_1, \ldots, x_n$ is a finite string over S_n arising from a finite number of applications of the operations $\neg$ and $\oplus$ as follows. The elements 0 and x_i, for $i = 1, \ldots, n$, considered as one-element strings, are MV terms.*

1. *If the string τ is an MV term, then $\neg\tau$ is also an MV term.*
2. *If the strings τ and γ are MV terms, then $(\tau \oplus \gamma)$ is also an MV term.*

We write $\tau(x_1, \ldots, x_n)$ to emphasize that τ is an MV term in the variables $x_1, \ldots, x_n$.

For instance, the following finite strings over $S_2 = \{(,), 0, \neg, \oplus, x_1, x_2\}$ are MV terms in the variables x_1 and x_2:

$$0, x_1, x_2, \neg 0, \neg x_2, (x_1 \oplus \neg x_2).$$

We shall always omit the outermost pair of brackets for conciseness, i.e., we write $x_1 \oplus \neg x_2$ instead of $(x_1 \oplus \neg x_2)$. Besides, for brevity we use the symbols $\odot$ and 1 as abbreviations according to (32) and (33) when writing MV terms.

MV terms are actually logical formulae without operational meaning. To endow them with meaning, an MV algebra must be specified. The associated truth functions, a.k.a. term functions which we define presently, are then obtained by interpreting the operations $\oplus$ and $\neg$ according to how they are specified in the MV algebra.

Definition A.3. [9] *Let $\tau(x_1, \ldots, x_n)$ be an MV term in the variables $x_1, \ldots, x_n$. Let $\mathcal{A} = \langle A, \oplus, \neg, 0\rangle$ be an MV algebra. The term function $\tau^{\mathcal{A}} : A^n \to A$ associated with τ in $\mathcal{A}$ is defined as follows. For every input $(a_1, \ldots, a_n) \in A^n$, first substitute a_i for all occurrences of the variable x_i in τ:*

1. *$x_i^{\mathcal{A}} : (a_1, \ldots, a_n) \mapsto a_i$, for $i = 1, \ldots, n$.*

Then, proceed by induction on the number of operation symbols, i.e., $\oplus$ and $\neg$, occurring in τ by applying the following rules:

2. *$(\neg\tau)^{\mathcal{A}} = \neg\tau^{\mathcal{A}}$, for MV term τ,*
3. *$(\tau \oplus \gamma)^{\mathcal{A}} = \tau^{\mathcal{A}} \oplus \gamma^{\mathcal{A}}$, for MV terms τ and γ.*

B ReLU networks

Definition B.1. [29] *Let $L \in \mathbb{N}$ and $N_0, N_1, \ldots, N_L \in \mathbb{N}$. A ReLU network is a map $\Phi : \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ given by*

$$\Phi = \begin{cases} W_1, & L = 1 \\ W_2 \circ \rho \circ W_1, & L = 2 \\ W_L \circ \rho \circ W_{L-1} \circ \rho \circ \cdots \circ \rho \circ W_1, & L \geq 3 \end{cases},$$

where, for $\ell \in \{1, 2, \ldots, L\}$, $W_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}, W_\ell(x) := A_\ell x + b_\ell$ are affine transformations with weight matrices $A_\ell = \mathbb{R}^{N_\ell \times N_{\ell-1}}$ and bias vectors $b_\ell \in \mathbb{R}^{N_\ell}$, and the ReLU activation function $\rho : \mathbb{R} \to \mathbb{R}, \rho(x) := \max\{0, x\}$ acts component-wise. Moreover, we denote by $\mathcal{N}_{d,d'}$ the set of ReLU networks with input dimension $N_0 = d$ and output dimension $N_L = d'$. Moreover, we define the number of layers of the network Φ, denoted by $\mathcal{L}(\Phi)$, to be equal to L.

The next result formalizes properties of ReLU network compositions.

Lemma B.2. [29] *Let $d_1, d_2, d_3 \in \mathbb{N}$, $\Phi_1 \in \mathcal{N}_{d_1,d_2}$, and $\Phi_2 \in \mathcal{N}_{d_2,d_3}$. There exists a network $\Psi \in \mathcal{N}_{d_1,d_3}$ with $\mathcal{L}(\Psi) = \mathcal{L}(\Phi_1) + \mathcal{L}(\Phi_2)$, and satisfying*

$$\Psi(x) = (\Phi_2 \circ \Phi_1)(x), \quad \text{for all } x \in \mathbb{R}^{d_1}.$$

Proof. The proof is based on the identity $x = \rho(x) - \rho(-x)$. First, note that by Definition B.1, we can write

$$\Phi_1 = W_{L_1}^1 \circ \rho \circ W_{L_1-1}^1 \circ \cdots \circ \rho \circ W_1^1$$

and

$$\Phi_2 = W_{L_2}^2 \circ \rho \circ W_{L_2-1}^2 \circ \cdots \circ \rho \circ W_1^2.$$

Next, let $N_{L_1-1}^1$ denote the width of layer $L_1 - 1$ in Φ_1 and N_1^2 the width of layer 1 in Φ_2. We define the affine transformations $\widetilde{W}_{L_1}^1 : \mathbb{R}^{N_{L_1-1}^1} \to \mathbb{R}^{2d_2}$ and $\widetilde{W}_1^2 : \mathbb{R}^{2d_2} \to \mathbb{R}^{N_1^2}$ according to

$$\widetilde{W}_{L_1}^1(x) := \begin{pmatrix} \mathbb{I}_{d_2} \\ -\mathbb{I}_{d_2} \end{pmatrix} W_{L_1}^1(x)$$
$$\widetilde{W}_1^2(x) := W_1^2 \left(\begin{pmatrix} \mathbb{I}_{d_2} & -\mathbb{I}_{d_2} \end{pmatrix} x \right).$$

The proof is completed by noting that the network

$$\Psi := W_{L_2}^2 \circ \cdots \circ W_2^2 \circ \rho \circ \widetilde{W}_1^2 \circ \rho \circ \widetilde{W}_{L_1}^1 \circ \rho \circ W_{L_1-1}^1 \circ \cdots \circ W_1^1$$

satisfies the claimed properties. □

Lemma B.3. *There exist ReLU networks $\Phi^\oplus \in \mathcal{N}_{2,1}$ and $\Phi^\odot \in \mathcal{N}_{2,1}$ satisfying*

$$\Phi^\oplus(x, y) = \min\{1, x + y\}$$
$$\Phi^\odot(x, y) = \max\{0, x + y - 1\},$$

for all $x, y \in [0, 1]$.

Proof. First, to realize the operation $x \oplus y = \min\{1, x + y\}$, we note that addition can be implemented by a single-layer ReLU network according to

$$x + y = \begin{pmatrix} 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$

It remains to realize the "min" operation with a ReLU network. To this end, we observe that

$$\min\{1, x\} = 1 - \rho(1 - x) = (W_2 \circ \rho \circ W_1)(x),$$

for $x \in [0, 1]$, where

$$W_1(x) = -x + 1, \qquad W_2(x) = -x + 1.$$

Now, applying Definition B.2 to concatenate the networks $\Phi_1(x,y) = \begin{pmatrix}1 & 1\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}$ and $\Phi_2(x) = (W_2 \circ \rho \circ W_1)(x)$ yields the desired ReLU network realization of $x \oplus y$ according to

$$x \oplus y = (W_2^{\oplus} \circ \rho \circ W_1^{\oplus})(x,y),$$

for $x, y \in [0,1]$, where

$$W_1^{\oplus}(x,y) = \begin{pmatrix}-1 & -1\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} + 1, \quad W_2^{\oplus}(x) = -x + 1.$$

For the operation $x \odot y = \max\{0, x+y-1\}$, we simply note that

$$\max\{0, x+y-1\} = \rho\left(\begin{pmatrix}1 & 1\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} - 1\right) = (W_2^{\odot} \circ \rho \circ W_1^{\odot})(x,y),$$

for $x, y \in [0,1]$, where

$$W_1^{\odot}(x,y) = \begin{pmatrix}1 & 1\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} - 1, \qquad W_2^{\odot}(x) = x.$$

□

C Proof of Lemma 2.3

Proof. We follow the line of arguments in [13] and consider four different cases.
Case 1: $f_\circ(x) \geq 1$, for all $x \in [0,1]^n$. In this case, the LHS of (4) is

$$\sigma(f) = 1.$$

The RHS becomes

$$(\sigma(f_\circ) \oplus x_1) \odot \sigma(f_\circ + 1) = (1 \oplus x_1) \odot 1 = 1.$$

Case 2: $f_\circ(x) \leq -1$, for all $x \in [0,1]^n$. In this case, the LHS of (4) is

$$\sigma(f) = 0$$

and the RHS satisfies

$$(\sigma(f_\circ) \oplus x_1) \odot \sigma(f_\circ + 1) = (0 \oplus x_1) \odot 0 = 0.$$

Case 3: $-1 < f_\circ(x) \leq 0$, for all $x \in [0,1]^n$. In this case, $f \in (-1,1]$ as $x_1 \in [0,1]$. The RHS of (4) becomes

$$\begin{aligned}
&(\sigma(f_\circ) \oplus x_1) \odot \sigma(f_\circ + 1) \\
&= (0 \oplus x_1) \odot (f_\circ + 1) \\
&= x_1 \odot (f_\circ + 1) \\
&= \max\{0, x_1 + f_\circ + 1 - 1\} \\
&= \max\{0, f\} \\
&= \sigma(f).
\end{aligned}$$

Case 4: $0 < f_\circ(x) < 1$, for all $x \in [0,1]^n$. In this case, $f \in (0,2)$. The RHS of (4) becomes

$$\begin{aligned}
&(\sigma(f_\circ) \oplus x_1) \odot \sigma(f_\circ + 1) \\
&= (f_\circ \oplus x_1) \odot 1 \\
&= f_\circ \oplus x_1 \\
&= \min\{1, f_\circ + x_1\} \\
&= \min\{1, f\} \\
&= \sigma(f).
\end{aligned}$$

□

D Proof of Lemma 4.3

We can follow the line of arguments in the proof of Lemma 2.3 and consider four different cases. *Case 1:* $f_\circ(x) \geq 1$, for all $x \in [0,1]^n$. In this case, the LHS of (29) is

$$\sigma(f) = 1.$$

The RHS becomes

$$(\sigma(f_\circ) \oplus (mx_i)) \odot \sigma(f_\circ + 1) = (1 \oplus (mx_i)) \odot 1 = 1.$$

Case 2: $f_\circ(x) \leq -1$, for all $x \in [0,1]^n$. In this case, the LHS of (29) is

$$\sigma(f) = 0$$

and the RHS is given by

$$(\sigma(f_\circ) \oplus (mx_i)) \odot \sigma(f_\circ + 1) = (0 \oplus (mx_i)) \odot 0 = 0.$$

Case 3: $-1 < f_\circ(x) \leq 0$, for all $x \in [0,1]^n$. In this case, $f \in (-1,1]$ as $mx_i \in [0,1]$. The RHS of (4) becomes

$$\begin{aligned}
&(\sigma(f_\circ) \oplus (mx_i)) \odot \sigma(f_\circ + 1) \\
&= (0 \oplus (mx_i)) \odot (f_\circ + 1) \\
&= (mx_i) \odot (f_\circ + 1) \\
&= \max\{0, mx_i + f_\circ + 1 - 1\} \\
&= \max\{0, f\} \\
&= \sigma(f).
\end{aligned}$$

Case 4: $0 < f_\circ(x) < 1$, for all $x \in [0,1]^n$. In this case, $f \in (0,2)$. The RHS of (29) becomes

$$\begin{aligned}
&(\sigma(f_\circ) \oplus (mx_i)) \odot \sigma(f_\circ + 1) \\
&= (f_\circ \oplus (mx_i)) \odot 1 \\
&= f_\circ \oplus (mx_i) \\
&= \min\{1, f_\circ + mx_i\} \\
&= \min\{1, f\} \\
&= \sigma(f).
\end{aligned}$$

E Construction of the network Φ^τ in Section 2

By Lemma B.3, we have the ReLU network associated with $x \oplus x$ according to

$$W_2^\oplus \circ \rho \circ \left(\begin{pmatrix} -1 & -1 \end{pmatrix} \begin{pmatrix} x \\ x \end{pmatrix} + 1 \right),$$

which can be rewritten as

$$W_2^\oplus \circ \rho \circ (-2x + 1). \tag{35}$$

As $y = \rho(y) - \rho(-y)$, for $y \in \mathbb{R}$, and $\neg y = 1 - y$, for $y \in [0,1]$, we obtain the 2-layer ReLU network associated with $\neg y$ according to

$$-\rho(y) + \rho(-y) + 1. \tag{36}$$

The ReLU network realizing the term function associated with $\tau = (x \oplus x) \odot \neg y$ is obtained by composing the network $\Phi^\odot$ in Lemma B.3 with (35) and (36) according to

$$W_2^\odot \circ \rho \circ W_1^\odot \circ \begin{pmatrix} W_2^\oplus \circ \rho \circ (-2x + 1) \\ -\rho(y) + \rho(-y) + 1 \end{pmatrix},$$

which yields the network Φ^τ in (3).