CELLULAR AUTOMATA, MANY-VALUED LOGIC, AND DEEP NEURAL NETWORKS

YANI ZHANG AND HELMUT BÖLCSKEI

Abstract. We develop a theory characterizing the fundamental capability of deep neural networks to learn, from evolution traces, the logical rules governing the behavior of cellular automata (CA). This is accomplished by first establishing a novel connection between CA and Łukasiewicz propositional logic. While binary CA have been known for decades to essentially perform operations in Boolean logic, no such relationship exists for general CA. We demonstrate that many-valued (MV) logic, specifically Łukasiewicz propositional logic, constitutes a suitable language for characterizing general CA as logical machines. This is done by interpolating CA transition functions to continuous piecewise linear functions, which, by virtue of the McNaughton theorem, yield formulae in MV logic characterizing the CA. Recognizing that deep rectified linear unit (ReLU) networks realize continuous piecewise linear functions, it follows that these formulae are naturally extracted from CA evolution traces by deep ReLU networks. A corresponding algorithm together with a software implementation is provided. Finally, we show that the dynamical behavior of CA can be realized by recurrent neural networks.

§1. Introduction. Neural networks were originally introduced as a computational model whose functionality imitates that of the human brain [31]. Machine learning based on neural networks has achieved state-of-the-art results in numerous applications, such as pattern recognition [30], game intelligence [40], or protein structure prediction [26]. Informally speaking, a neural network consists of layers of nodes that are connected by weighted edges. In practice, the network topology and weights are learned through training on data either in a supervised [28] or an unsupervised [34] manner.

An ability ingrained in the human brain is to draw inferences from data it is presented with and apply what has been learned to new problems. In machine learning parlance, this aspect is usually referred to as generalization or extrapolation. For example, given the geometric sequence of integers $1, 2, 4, 8, \dots,$ one would conclude that the rule generating the sequence is: The next number is obtained by multiplying the present number by 2. Based on this extracted logical rule, the entire sequence can now

H. Bölcskei gratefully acknowledges support by the Lagrange Mathematics and Computing Research Center, Paris, France.

be calculated. Such logical reasoning tasks, which appear natural to humans, may be challenging to learn for neural networks. Moreover, even if the network had learned, e.g., the mechanism generating the geometric sequence above, it would have the corresponding logical rule encoded in its topology and weights and it is unclear how to extract the rule and present it in a manner accessible to humans.

In this paper, we report an attempt at building a theory characterizing the fundamental capability of neural networks to learn logical rules from data. This immediately leads to the question, "What is the logical structure behind data?" Here, we shall make a first step towards developing the corresponding research program. Concretely, we consider data generated by cellular automata[1] (CA). Abstractly speaking, a CA is a discrete dynamical system evolving on a regular lattice whose points take values in a finite set. Starting from an initial configuration, all lattice points change their states at synchronous discrete time steps according to a transition function that takes the present values of the lattice point under consideration and its neighbors as inputs. The evolution of the lattice point states over time furnishes the data sequence we shall be interested in. Now, assume that the underlying CA is binary, i.e., the state set is given by $\{0, 1\}$. In this case the transition function governing the CA evolution is a Boolean function [46, pp. 29-31], which can be viewed as determining the logical structure behind the data sequence generated by the CA.

The question we shall ask is whether one can extract the logical rule governing a general (i.e., not only binary) CA by training a neural network on data generated by the CA. While the connection between binary CA and Boolean logic has been known for decades [49], [46], to the best of our knowledge no such relationship has been reported in the literature for CA with arbitrary state sets. One of the main conceptual contributions of this paper is to demonstrate that many-valued[2] (MV) logic is a suitable language for interpreting general CA as logical machines. We then show that all possible transition functions can be realized by deep ReLU networks[3], which, in turn, are found to naturally express statements in MV logic. The dynamical system component of CA is demonstrated to be realizable by recurrent neural networks (RNNs). Finally, we propose and analyze a procedure for extracting the logical formulae behind CA transition functions from neural networks trained on corresponding CA evolution data.

[1] We will abbreviate both the singular "cellular automaton" and the plural form "cellular automata" as "CA".

[2] With slight abuse of terminology, we shall use the term MV logic to refer to Łukasiewicz propositional logic.

[3] ReLU stands for the Rectified Linear Unit nonlinearity, defined as $x \mapsto \max\{0, x\}$.

Notation: $\mathbb{1}_{\{\cdot\}}$ denotes the truth function which takes the value 1 if the statement inside $\{\cdot\}$ is true and equals 0 otherwise. 0_N stands for the N-dimensional column vector with all entries equal to 0. $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$ and $|A|$ stands for the cardinality of the set A. $||\cdot||_1$ is the ℓ_1-norm.

1.1. Cellular automata. CA were invented in the 1940s by von Neumann [45] and Ulam [43] in an effort to build models that are capable of universal computation and self-reproduction. Von Neumann's conceptualization emphasized the aspect of self-reproduction, while Ulam suggested the use of finite state machines on two-dimensional lattices. A widely known CA is the two-dimensional Game of Life devised by Conway [18]. Despite the simplicity of its rules, the Game of Life exhibits remarkable behavioral complexity and has therefore attracted widespread and longstanding interest. We begin by briefly reviewing the Game of Life.

Consider an infinite two-dimensional grid of square cells centered on the points of an underlying lattice (symbolized by dashed lines in Figure 1). Initially, each cell (or equivalently lattice point) is in one of two possible states, namely "live" or "dead". Each cell has eight neighbors, taking into account horizontal, vertical, and diagonal directions (see Figure 1). For a given cell, we refer to the set made up of the cell itself and its neighbors as the neighborhood of the cell. The cells change their states synchronously at discrete time steps, all following the same rule given by:

1. Every live cell with two or three live neighbors stays live; other live cells turn into dead cells.
2. Every dead cell with three live neighbors becomes live; other dead cells stay dead.

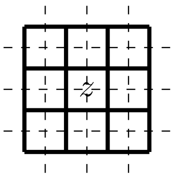

FIGURE 1. Cell z and its neighborhood.

This process repeats at discrete time steps, forming an evolution of the system. We refer to Figure 2 for an example illustration. As already mentioned, despite the simplicity of the transition rule, the evolutions induced by different initial configurations exhibit rich and complex long-term behavior. It is shown in [8], [47], [35], for example, that given an initial configuration with a finite number of live cells, it is an undecidable problem whether all these live cells would eventually die out. We refer to [4] for an in-depth discussion of the behavioral patterns of the Game of Life and proceed to formally define CA.

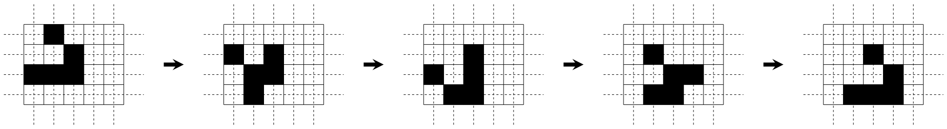

FIGURE 2. Steps of the evolution of the Game of Life. Black cells are live; white cells are dead.

DEFINITION 1.1 (Cellular automaton). Let $d, n, k \in \mathbb{N}, k \geq 2$. A cellular automaton is an ordered quadruple $(\mathbb{Z}^d, K, \mathcal{E}, f)$, where

1. the d-dimensional lattice $\mathbb{Z}^d$ is referred to as the cellular space,
2. $K = \{0, \frac{1}{k-1}, \ldots, \frac{k-2}{k-1}, 1\}$ is the state set with k states,
3. $\mathcal{E} = \{0_d, z_1, \ldots, z_{n-1}\} \subset \mathbb{Z}^d$ is the neighborhood set,
4. $f : K^n \to K$ is the transition function.

The lattice dimension d is called the cellular space dimension. Cells are centered on the lattice points of $\mathbb{Z}^d$. A configuration, denoted by c, over the cellular space assigns a state in the state set to each cell (or equivalently lattice point). For $z \in \mathbb{Z}^d$, denote by $c[z]$ its state under the configuration c. Let $\mathcal{C}$ be the set of all possible configurations over the cellular space. The CA map $F : \mathcal{C} \to \mathcal{C}$ effecting the configuration evolution from one time step to the next is determined by the transition function f according to

$$F(c)[z] = f(c[z], c[z + z_1], \ldots, c[z + z_{n-1}]), \qquad \text{for all } z \in \mathbb{Z}^d. \tag{1}$$

Remark. The CA map F is often referred to as the "global mapping function", whereas the transition function f is called the "local mapping function". Note that our choice of the state set K in Definition 1.1 is without loss of generality, as for every general state set K^*, there is a bijection $h : K^* \to K$, with $|K^*| = |K|$. For example, for the state set $K^* = \{1, 2, 3\}$, $h : K^* \to \{0, 1/2, 1\}, h(x) = (x - 1)/2$ is a bijection.

We continue with examples illustrating Definition 1.1.

EXAMPLE 1.1. Consider a CA with cellular space dimension $d = 1$, neighborhood set $\mathcal{E} = \{0, -1\} \subset \mathbb{Z}$, state set $K = \{0, 1\}$, and the transition function $f : K^2 \to K$ as provided in Table 1. The CA map F is given by

$$F(c)[z] = f(c[z], c[z - 1]), \qquad \text{for all } z \in \mathbb{Z}.$$

Figure 3 depicts one step of the CA evolution.

$x_0\, x_{-1}$	00	10	01	11
$f(x_0, x_{-1})$	0	0	0	1

TABLE 1. Transition function f.

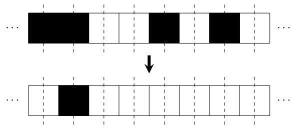

FIGURE 3. The state 1 is indicated by black cells, the state 0 by white cells.

EXAMPLE 1.2 (Game of Life). Consider a CA with cellular space dimension $d = 2$, state set $K = \{0, 1\}$, and neighborhood set $\mathcal{E} = \{0_2, z_1, \ldots, z_8\} \subset \mathbb{Z}^2$, where

$$z_1 = (-1, 1),\ z_2 = (0, 1),\ z_3 = (1, 1),\ z_4 = (-1, 0),$$
$$z_5 = (1, 0),\ z_6 = (-1, -1),\ z_7 = (0, -1),\ z_8 = (1, -1).$$

The transition function $f : K^9 \to K$ is given by

$$f(x_0, \ldots, x_8) = \begin{cases} \mathbb{1}_{x_1+\cdots+x_8=3}, & \text{if } x_0 = 0, \\ \mathbb{1}_{x_1+\cdots+x_8=2} + \mathbb{1}_{x_1+\cdots+x_8=3}, & \text{if } x_0 = 1. \end{cases}$$

The resulting CA is the Game of Life with dead cells corresponding to state 0 and live cells to state 1.

One of the main goals of this paper is to show that deep ReLU networks are capable of learning CA transition functions from CA evolution sequences. This result also establishes that every CA transition function can be realized by a deep ReLU network. Concretely, we build this universal CA transition function representation theorem in two steps. First, we establish a correspondence between CA transition functions and MV algebras [12]. Then, we show that statements in MV logic are naturally expressed by deep ReLU networks. En route it is shown that MV logic is a suitable language for describing CA with general state sets as logical machines.

1.2. One-dimensional binary CA. In order to develop intuition on the connection between CA, MV logic, and ReLU networks, we consider a simple example setup. Specifically, we investigate CA with cellular space $\mathbb{Z}$, state set $K = \{0, 1\}$, and neighborhood set $\mathcal{E} = \{-1, 0, 1\}$. This class of CA is referred to as elementary CA [49]. As the transition function depends on the states of the cell itself and its left and right neighbor, there are 2^3 possible states for a given cell's neighborhood and hence a total of $2^{2^3} = 256$ transition functions. Wolfram proposed a scheme [49] to index the associated CA using an 8-digit binary string. For example, Table 2 specifies the transition function of the elementary CA of index 30 ($30_{10} = 00011110_2$), and Table 3 contains that corresponding to elementary CA 110 ($110_{10} = 01101110_2$). It is remarkable that elementary CA, despite their simplicity, are capable of universal computation. Specifically, Cook [13] proved that elementary CA 110 is Turing complete.

$x_{-1}\,x_0\,x_1$	111	110	101	100	011	010	001	000
$f_{30}(x_{-1},x_0,x_1)$	0	0	0	1	1	1	1	0

Table 2. Transition function of elementary CA 30.

$x_{-1}\,x_0\,x_1$	111	110	101	100	011	010	001	000
$f_{110}(x_{-1},x_0,x_1)$	0	1	1	0	1	1	1	0

Table 3. Transition function of elementary CA 110.

We now bring Boolean logic into the picture, concretely by interpreting the state set elements 0 and 1 as truth values in Boolean algebra, and assuming that x_{-1}, x_0, x_1 are Boolean propositional variables. The transition function f_{110}, for example, can in effect be regarded as a Boolean function, namely $f_{110} = \texttt{OR}(\texttt{XOR}(x_0, x_1), \texttt{AND}(\texttt{NOT}(x_{-1}), \texttt{OR}(x_0, x_1)))$. In the same manner, we get $f_{30}(x_{-1}, x_0, x_1) = \texttt{XOR}(x_{-1}, \texttt{OR}(x_0, x_1))$. As shown in [46, pp. 29-31], [38, Section 12.2], the transition function of every elementary CA can be expressed as a formula in Boolean logic, particularly in a canonical form called disjunctive normal form (DNF), which we define next.

Definition 1.2. In Boolean logic, a finite string of symbols is called a *conjunction* if it is given by AND connections of a finite number of propositional variables (possibly negated), and a *disjunction* if the connections are effected through the OR operation. A logical formula is said to be in *disjunctive normal form* if it is given by the OR connection of a finite number of conjunctions.

Let the OR operation be denoted by $\oplus$, AND by $\odot$, and NOT by $\neg$. Following the procedure in [38, Section 12.2], we can express the transition function in Table 2 in DNF according to

$$f_{30} = (x_{-1} \odot \neg x_0 \odot \neg x_1) \oplus (\neg x_{-1} \odot x_1) \oplus (\neg x_{-1} \odot x_0). \tag{2}$$

Having established the connection between elementary CA and Boolean logic, by expressing the transition function in terms of a Boolean formula, we are ready to have ReLU networks enter the story. Specifically, this shall be effected by realizing the Boolean logical operations $\oplus$, $\odot$, and $\neg$ through ReLU networks (see Definition 3.1 below). With the ReLU function $\rho : \mathbb{R} \to \mathbb{R}$, $\rho(x) := \max\{0, x\}$, the binary and n-ary conjunction can be written as

$$x_1 \odot x_2 = \rho(x_1 + x_2 - 1), \tag{3}$$

$$x_1 \odot \cdots \odot x_n = \rho(\Sigma_{i=1}^n x_i - (n-1)). \tag{4}$$

Likewise, the binary and n-ary disjunction can be realized according to

$$x_1 \oplus x_2 = \rho(x_1 + x_2) - \rho(x_1 + x_2 - 1), \tag{5}$$

$$x_1 \oplus \cdots \oplus x_n = \rho(\Sigma_{i=1}^n x_i) - \rho(\Sigma_{i=1}^n x_i - 1). \tag{6}$$

Finally, the Boolean operation $\neg$ can be expressed as

$$\neg x = 1 - x.$$

We now combine these results to realize the logical formula f_{30} in (2) in terms of the ReLU network $\Phi^{f_{30}} : \mathbb{R}^3 \to \mathbb{R}$ with three layers according to

$$\Phi^{f_{30}} := W_3^{f_{30}} \circ \rho \circ W_2^{f_{30}} \circ \rho \circ W_1^{f_{30}} \tag{7}$$

where

$$W_1^{f_{30}}(x_{-1}, x_0, x_1) = \begin{pmatrix} -1 & -1 & 1 \\ 1 & 0 & -1 \\ 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_0 \\ x_{-1} \end{pmatrix},$$

$$W_2^{f_{30}}(x) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} x + \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \qquad \text{for } x \in \mathbb{R}^3,$$

$$W_3^{f_{30}}(x) = \begin{pmatrix} 1 & -1 \end{pmatrix} x, \qquad \text{for } x \in \mathbb{R}^2.$$

It can now be verified directly that

$$\Phi^{f_{30}}(x_{-1}, x_0, x_1) = f_{30}(x_{-1}, x_0, x_1), \text{ for } (x_{-1}, x_0, x_1) \in \{0,1\}^3.$$

We hasten to add that ReLU network realizations of Boolean formulae are not unique. For example, upon noting that $\Phi^{\mathrm{Id}}(x) = \rho(x) - \rho(-x) = x, x \in \mathbb{R}$, is a valid ReLU network, the map $\Phi^{f_{30}}$ in (7) can be augmented according to

$$\Phi^{\mathrm{Id}} \circ \cdots \circ \Phi^{\mathrm{Id}} \circ \Phi^{f_{30}}$$

without changing the input-output function it realizes. This results in an infinite collection of different ReLU networks, all realizing the Boolean formula f_{30}. In addition, the Boolean logical formula corresponding to a given binary truth table is not unique either [38, Section 12.4]. These issues of nonuniqueness will be addressed in more detail in Section 4.

We can summarize what was done in this section as follows. In an effort to implement the transition function of a one-dimensional binary CA by a ReLU neural network, we first expressed the transition function, originally specified in the form of a truth table, by a formula in Boolean logic. This formula was then realized by a ReLU network. The bridge via Boolean algebra effected in this manner serves to illustrate that CA are actually "logical machines", an aspect elaborated on in the literature before [46], [49], [45], but notably only in the binary case.

Inspired by this connection, we shall show that the transition functions of general CA with arbitrary state sets, in particular of arbitrary cardinality, can be expressed in terms of formulae in MV logic [12], which, in turn, are found to be naturally realized by ReLU networks. Embedding these (feedforward) networks into a recurrent neural network structure [17], we get a dynamical system realization of general CA, entirely in terms of neural networks.

1.3. Previous work. Despite the fact that neural networks and CA stand for parallel efforts in building computational models of the human brain [9], [27], a profound theoretical understanding of the connections between the two structures seems to be lacking. Wulff and Hertz [52] consider shallow neural networks with the threshold activation function to learn the transition functions of one- and two-dimensional binary CA. While no details on the training algorithm are provided, the experiments in [52] lead to the conclusion that the approach employed is capable of learning only a small subset of the CA considered. More recently, Gilpin [20] designed a convolutional neural network simulating two-dimensional CA with arbitrary state sets and neighborhood size.

The connection between general CA and MV logic we report here seems completely new. For binary CA and Boolean logic, besides the classical results in [46], [49], more recent work studying CA evolution through the lens of formal logic is reported in [14], [25]. Specifically, Sukanta and Chakraborty [14] show that binary CA on finite one-dimensional cellular spaces are models of certain logical languages in the domain of binary strings. Inokuch et al. [25] use the interpretation of binary CA transition functions as Boolean logical formulae to demonstrate that the multiplication of logical formulae, defined by monoid action, corresponds to the composition of global CA maps associated with the logical formulae under consideration.

As for the connection between MV logic and neural networks, Amato et al. [6] establish a correspondence between formulae in Łukasiewicz logic and neural networks with rational weights and the clipped ReLU (CReLU) nonlinearity $\sigma(x) = \min\{1, \max\{0, x\}\}$. In [15] this result is further extended to a correspondence between Riesz MV algebras and CReLU networks with real weights; this is accomplished by adding scalar multiplication over the real numbers to Łukasiewicz propositional logic.

1.4. Outline of the paper. Section 2 establishes that CA are in essence machines performing MV logic. In Section 3, we show how formulae in MV logic can be realized by (deep) ReLU networks. The problem of extracting CA transition functions, along with underlying formulae in MV logic, from ReLU networks that have been trained on corresponding CA evolution data is addressed in Section 4.

§2. Many-valued logic and cellular automata.

We now turn to generalizing the correspondence between one-dimensional binary CA and Boolean logic. Specifically, we show that CA with arbitrary cellular space dimension and arbitrary state set (cardinality) can be seen as machines carrying out operations in Łukasiewicz propositional logic [42], a many-valued extension of Boolean logic. The corresponding algebraic counterpart is known as Chang's many-valued algebras [10]. The connection we uncover is built on a fundamental result in MV logic, which states that every piecewise linear function, whose linear pieces have integer coefficients, corresponds to a formula in MV logic and vice versa. Known in the literature as McNaughton theorem [32], this result constitutes the equivalent of translating between binary truth tables and Boolean logical formulae.

2.1. Many-valued algebras.

We start with a brief review, following [12], of the basic elements in the theory of MV algebras.

Definition 2.1. 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:

$$x \oplus (y \oplus z) = (x \oplus y) \oplus z \tag{8.1}$$

$$x \oplus y = y \oplus x \tag{8.2}$$

$$x \oplus 0 = x \tag{8.3}$$

$$\neg\neg x = x \tag{8.4}$$

$$x \oplus \neg 0 = \neg 0 \tag{8.5}$$

$$\neg(\neg x \oplus y) \oplus y = \neg(\neg y \oplus x) \oplus x. \tag{8.6}$$

Specifically, (8.1)-(8.3) 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 $|A| > 1$. On each MV algebra we can define a constant 1 and a binary operation $\odot$ as follows:

$$1 := \neg 0 \tag{9}$$

$$x \odot y := \neg(\neg x \oplus \neg y). \tag{10}$$

The ensuing identities are then direct consequences of Definition 2.1:

$$x \odot (y \odot z) = (x \odot y) \odot z \tag{11.1}$$

$$x \odot y = y \odot x \tag{11.2}$$

$$x \odot 1 = x \tag{11.3}$$

$$x \odot 0 = 0. \tag{11.4}$$

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

Definition 2.2 (MV term). 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 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.

In the remainder of the paper, 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 strings over $S_2 = \{(,), 0, \neg, \oplus, x_1, x_2\}$ are MV terms in the variables x_1 and x_2:

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

MV terms are syntactic expressions without semantic meaning. To endow them with semantics, an underlying MV algebra must be specified. The resulting functions will be referred to as term functions.

Definition 2.3 (Term function). Let $\tau(x_1, \ldots, x_n)$ be an MV term and $\mathcal{A} = \langle A, \oplus, \neg, 0\rangle$ an MV algebra. The term function $\tau^{\mathcal{A}} : A^n \to A$ associated with τ under $\mathcal{A}$ is obtained by substituting, for $i = 1, \ldots, n$, $a_i \in A$ for all occurrences of x_i in τ and interpreting the symbols $\oplus$ and $\neg$ according to how they are specified in $\mathcal{A}$.

We next demonstrate that Boolean algebra is an MV algebra.

Definition 2.4 (Boolean algebra). Consider the set $B = \{0, 1\}$. Define the binary operation $\oplus$ and the unary operation $\neg$ on B according to

$$\begin{aligned} 0 \oplus 0 &= 0 \\ 0 \oplus 1 &= 1 \\ 1 \oplus 0 &= 1 \\ 1 \oplus 1 &= 1 \end{aligned} \qquad \text{and} \qquad \begin{aligned} \neg 0 &= 1 \\ \neg 1 &= 0. \end{aligned}$$

It is readily verified that the resulting structure $\mathcal{B} = \langle B, \oplus, \neg, 0\rangle$ satisfies the axioms in Definition 2.1 and is hence an MV algebra. In fact, by further defining the binary operation $\odot$ on B through

$$x \odot y := \neg(\neg x \oplus \neg y),$$

which yields

$$\begin{aligned} 0 \odot 0 &= 0 \\ 0 \odot 1 &= 0 \\ 1 \odot 0 &= 0 \\ 1 \odot 1 &= 1, \end{aligned}$$

we immediately see that the operations $\odot, \oplus$, and $\neg$ correspond to the `AND`, `OR`, and `NOT` operations, respectively, in Boolean logic with 0 designating `False` and 1 standing for `True`.

Recall the informal discussion in Section 1.2 explaining how the transition functions of binary one-dimensional CA can be expressed in terms of Boolean logic. We now formalize this observation (for arbitrary cellular space dimension) by casting it into MV algebras.

LEMMA 2.1. *Consider a CA with cellular space dimension $d \in \mathbb{N}$, neighborhood size $n \in \mathbb{N}$, state set $K = \{0,1\}$, and transition function $f : K^n \to K$ specified in the form of a (Boolean) truth table. There exists an MV term $\tau(x_1, \ldots, x_n)$ with associated term under the Boolean algebra $\mathcal{B} = \langle \{0,1\}, \oplus, \neg, 0\rangle$ satisfying*

$$\tau^{\mathcal{B}}(x_1, \ldots, x_n) = f(x_1, \ldots, x_n), \qquad \textit{for all } (x_1, \ldots, x_n) \in \{0,1\}^n.$$

PROOF. We first transform the truth table specifying the transition function f into a Boolean formula [38, Section 12.2]. Identifying, in the resulting Boolean algebraic expression, the Boolean operations `AND`, `OR`, and `NOT` with the operations $\odot$, $\oplus$, and $\neg$ in a general MV algebra yields a valid MV term τ. Converting τ into its associated term function $\tau^{\mathcal{B}}$ according to Definition 2.3 establishes the result. $\dashv$

In order to extend Lemma 2.1 to CA with state sets of arbitrary cardinality, we need a logic that can cope with more than two truth values. We shall see that the theory of MV logic [12] provides a suitable framework and start by developing the underlying algebraic structure, namely that of MV algebras.

2.2. MV algebras and CA. We begin with a simple example of an MV algebra.

EXAMPLE 2.1. For $k \in \mathbb{N}$, $k \geq 2$, consider the CA state set

$$K = \left\{0, \frac{1}{k-1}, \ldots, \frac{k-2}{k-1}, 1\right\} \tag{12}$$

of cardinality k. Define the binary operation $\oplus$ and the unary operation $\neg$ on K according to

$$\begin{aligned} x \oplus y &= \min\{1, x+y\} \\ \neg x &= 1 - x. \end{aligned}$$

It is readily seen that the structure $\mathcal{A}_k := \langle K, \oplus, \neg, 0\rangle$ satisfies the axioms in Definition 2.1 and hence constitutes an MV algebra. For $k = 2$, $\mathcal{A}_k$ reduces to the Boolean algebra in Definition 2.4.

For the MV algebra $\mathcal{A}_k$ in Example 2.1, given an MV term τ in n variables, the associated term function $\tau^{\mathcal{A}_k} : K^n \to K$ can be interpreted as a CA transition function. We illustrate this observation through a simple example, but first note that $\tau^{\mathcal{A}_k}$ is guaranteed to map K^n to K owing to the specific structure of the state set K and the operations $\oplus$ and $\neg$.

Example 2.2. Consider the MV algebra $\mathcal{A}_3$ according to Example 2.1 with the associated set $K = \{0, 1/2, 1\}$. The term function $\tau^{\mathcal{A}_3} : K^3 \to K$ corresponding to the MV term $\tau = x_{-1} \oplus x_0 \oplus x_1$ equals the transition function of the 3-color totalistic CA [49] with cellular space $\mathbb{Z}$, neighborhood set $\mathcal{E} = \{-1, 0, 1\}$, state set K, and transition function f as specified in Table 4 (note that f depends only on the sum of the neighborhood state values).

$x_{-1} + x_0 + x_1$	3	5/2	2	3/2	1	1/2	0
$f(x_{-1}, x_0, x_1)$	1	1	1	1	1	1/2	0

Table 4. Transition function f of the totalistic CA.

Inspired by this example, we now ask whether every CA transition function has an underlying MV term with associated term function under a suitable MV algebra equal to the transition function. Formally speaking, we seek a generalization of Lemma 2.1 to the case of CA with arbitrary state set cardinality k. If answered in the affirmative, this would show that CA are essentially machines that perform operations in a suitable logic. It turns out that this is, indeed, the case, but we need to consider an MV algebra that accommodates state sets K of arbitrary cardinality k.

Definition 2.5. Consider the unit interval $[0, 1]$ on $\mathbb{R}$, 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. In particular, $\mathcal{I}$ constitutes the algebraic counterpart of Łukasiewicz propositional logic [10]. We further define the operation $x \odot y := \neg(\neg x \oplus \neg y) = \max\{0, x + y - 1\}$.

The MV algebra $\mathcal{I}$ in Definition 2.5 is the so-called standard MV algebra. As shown in [10, Section 5], [11], an equation holds in every MV algebra iff it holds in the standard MV algebra $\mathcal{I}$, endowing $\mathcal{I}$ with universality. Next, recall that for the binary state set $\{0, 1\}$, in the proof of Lemma 2.1 we started by converting the binary truth table specifying the transition function f into a formula in Boolean logic. In MV logic truth tables are given by mappings from $[0, 1]^n$ to $[0, 1]$. While in the Boolean case, every truth table can be cast into a formula in Boolean logic [38, Section 12.2], this does not hold for the MV algebra $\mathcal{I}$. The McNaughton theorem, stated next, explicitly characterizes the class of functions from $[0, 1]^n$ to $[0, 1]$ that have underlying MV terms τ. Here, "underlying" means that the term function associated with τ under $\mathcal{I}$ equals the function under consideration.

THEOREM 2.2. *[McNaughton theorem [32]] Consider the MV algebra $\mathcal{I} = \langle [0,1], \oplus, \neg, 0 \rangle$ in Definition 2.5. Let $n \in \mathbb{N}$. For a function $f_c : [0,1]^n \to [0,1]$ to have an associated MV term τ such that $\tau^{\mathcal{I}} = f_c$ on $[0,1]^n$, it is necessary and sufficient that*

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

$$p_j(x_1, \ldots, x_n) = m_{j1}x_1 + \cdots + m_{jn}x_n + b_j, \quad j = 1, \ldots, \ell,$$

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

Functions satisfying these conditions are called McNaughton functions.

We now demonstrate how the McNaughton theorem can be employed to decide whether a given CA transition function has an underlying MV term. Since term functions under $\mathcal{I}$ are continuous piecewise linear functions according to Theorem 2.2, we start by linearly interpolating CA transition functions $f : K^n \to K$ to continuous piecewise linear functions $f_{\mathrm{c}} : [0,1]^n \to [0,1]$. The following simple example illustrates the approach we pursue.

EXAMPLE 2.3. Consider a one-dimensional CA with cellular space $\mathbb{Z}$, neighborhood set $\mathcal{E} = \{0\}$, state set $K = \{0, 1/3, 2/3, 1\}$, and transition function f according to

x	0	1/3	2/3	1
$f(x)$	0	1	1	0

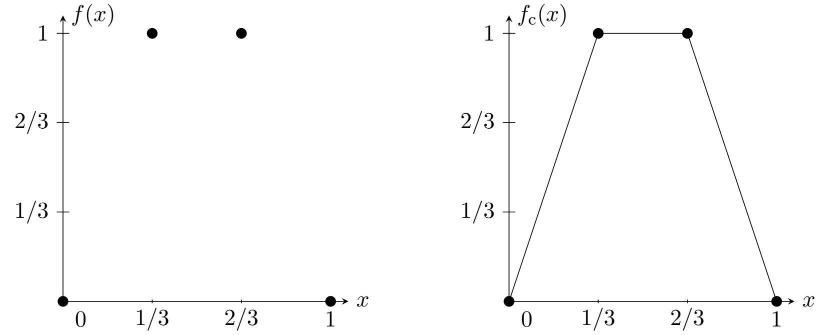

FIGURE 4. The transition function f and the associated function f_{c} in Example 2.3.

The continuous piecewise linear function $f_{\mathrm{c}} : [0,1] \to [0,1]$ obtained by interpolating f is given by

$$f_{\mathrm{c}}(x) = \begin{cases} 3x, & 0 \leq x < 1/3 \\ 1, & 1/3 \leq x < 2/3 \\ -3x+3, & 2/3 \leq x \leq 1 \end{cases} . \tag{13}$$

As f_{c} in (13) has exclusively integer coefficients and is piecewise linear and continuous by construction, it satisfies Conditions 1 and 2 in Theorem 2.2. Hence, there is an underlying MV term, shown in Appendix A to be given by

$$\tau = (x \oplus x \oplus x) \wedge \neg 0 \wedge (x \odot x \odot x),$$

where, for brevity, we write $x \wedge y = (x \oplus \neg y) \odot y$.

However, for the same cellular space and state set, a slightly different transition function g, given by

x	0	1/3	2/3	1
$g(x)$	0	1/3	1	0

yields the associated piecewise linear function

$$g_{\mathrm{c}}(x) = \begin{cases} x, & 0 \leq x < 1/3 \\ 2x - 1/3, & 1/3 \leq x < 2/3 \\ -3x+3, & 2/3 \leq x \leq 1 \end{cases} ,$$

which fails to satisfy Condition 2 in Theorem 2.2, as one of its linear pieces exhibits a noninteger coefficient. As the McNaughton theorem is iff, we can conclude that g_{c} does not have an underlying MV term τ such that $\tau^{\mathcal{I}} = g_{\mathrm{c}}$.

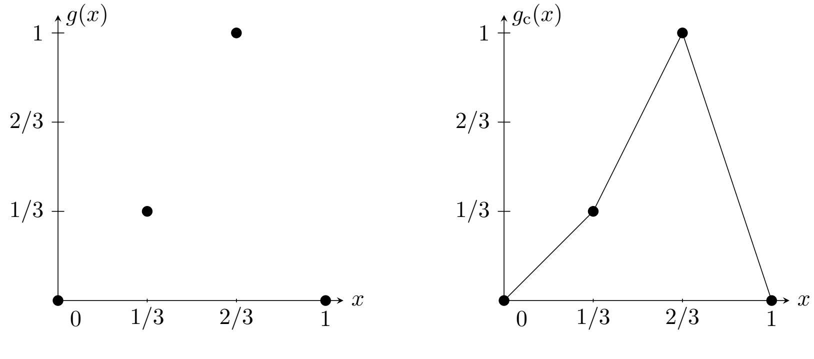

FIGURE 5. The transition function g and the continuous function g_{c} in Example 2.3.

Example 2.3 manifests missing structure for the MV algebra $\mathcal{I}$ to allow universal conversion from CA transition functions to MV terms. Concretely, inspection of the state set K shows that we would need a McNaughton theorem that allows rational coefficients in Condition 2 of Theorem 2.2. It turns out that so-called divisible many-valued (DMV) algebras resolve this issue.

Definition 2.6. Consider the MV algebra $\mathcal{I} = \langle [0,1], \oplus, \neg, 0 \rangle$ in Definition 2.5. Define the family of unary operations $\{\delta_i : [0,1] \to [0,1]\}_{i \in \mathbb{N}}$ according to

$$\delta_i x = \frac{1}{i} x, \quad x \in [0,1], \quad \text{for all } i \in \mathbb{N}.$$

It is easily verified that the structure $\mathcal{I}_{\mathrm{d}} = \langle [0,1], \oplus, \neg, 0, \{\delta_i\}_{i \in \mathbb{N}} \rangle$ is a DMV algebra [19]. In particular, it constitutes the algebraic counterpart of Rational Łukasiewicz logic [19].

The following result, often referred to as rational McNaughton theorem, explicitly characterizes the class of term functions under the DMV algebra $\mathcal{I}_{\mathrm{d}}$.

Theorem 2.3. *[Rational McNaughton theorem [7]] Consider the DMV algebra $\mathcal{I}_d = \langle [0,1], \oplus, \neg, 0, \{\delta_i\}_{i \in \mathbb{N}} \rangle$ in Definition 2.6. Let $n \in \mathbb{N}$. For a function $f_c : [0,1]^n \to [0,1]$ to have an associated DMV term τ such that $\tau^{\mathcal{I}_d} = f_c$ on $[0,1]^n$, it is necessary and sufficient that*

1. *f_c is continuous with respect to the natural topology on $[0,1]^n$,*
2. *there exist linear functions $p_1, \dots, p_\ell$ with rational coefficients, i.e.,*

$$p_j(x_1, \dots, x_n) = m_{j1} x_1 + \cdots + m_{jn} x_n + b_j, \quad j = 1, \dots, \ell,$$

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

The central result we have been working towards in this section now follows readily.

Proposition 2.4. *Consider a CA with cellular space dimension d, neighborhood size n, state set $K = \{0, 1/(k-1), \dots, (k-2)/(k-1), 1\}$ of cardinality $k \in \mathbb{N}, k \geq 2$, and transition function $f : K^n \to K$. There exists a DMV term $\tau(x_1, \dots, x_n)$ with associated term function under the DMV algebra $\mathcal{I}_d$ satisfying*

$$\tau^{\mathcal{I}_d}(x_1, \dots, x_n) = f(x_1, \dots, x_n), \quad \text{for all } (x_1, \dots, x_n) \in K^n.$$

Proof. We start by linearly interpolating the CA transition funtion $f : K^n \to K$ to a continuous function $f_c : [0,1]^n \to [0,1]$. As all the points

$$(x_1, \dots, x_n, f(x_1, \dots, x_n)) \in K^{n+1}$$

have exclusively rational coordinates, the linear pieces of the interpolated function f_{c} have rational coefficients as well. Application of Theorem 2.3 to f_{c} then yields a DMV term τ satisfying $\tau^{\mathcal{I}_{\mathrm{d}}} = f_{\mathrm{c}}$ on $[0,1]^n$. This implies $\tau^{\mathcal{I}_{\mathrm{d}}} = f_{\mathrm{c}} = f$ on K^n, as desired. ⊣

§3. Recurrent neural networks realize CA dynamics. In this section, we show that recurrent neural networks can realize the overall dynamical behavior of CA. This will be accomplished in two steps. First, we demonstrate that ReLU networks naturally realize operations in DMV algebras, which, in turn, leads to a universal realization theorem for CA transition functions. The ReLU network realizing the CA transition function is then embedded into an RNN that emulates the dynamics of the CA.

3.1. DMV algebras and ReLU neural networks. We start by formally defining ReLU neural networks [16].

Definition 3.1 (ReLU neural network). Let $L \in \mathbb{N}$ and N_0, $N_1, \dots,$ $N_L \in \mathbb{N}$. A ReLU neural 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 $\ell \in \{1, 2, \dots, L\}$, $W_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}, W_\ell(x) := A_\ell x + b_\ell$ 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. We denote by $\mathcal{N}_{d,d'}$ the set of ReLU neural networks of input dimension $N_0 = d$ and output dimension $N_L = d'$. The number of layers of the network Φ, denoted by $\mathcal{L}(\Phi)$, is defined to equal L.

It follows immediately from Definition 3.1 that ReLU networks realize continuous piecewise linear functions. We shall frequently make use of basic ReLU network constructions, namely compositions [16, Lemma II.3], augmentations [16, Lemma II.4], and parallelizations [16, Lemma A.7], collected here for completeness. The proofs of the corresponding results Lemma 3.1-3.3 provided in [16] present explicit network constructions that we will occasionally refer to.

Lemma 3.1 (Composition of ReLU networks [16]). *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* $\mathcal{L}(\Psi) = \mathcal{L}(\Phi_1) + \mathcal{L}(\Phi_2)$, *satisfying*

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

LEMMA 3.2 (Augmentation of ReLU networks [16]). *Let $d_1, d_2, L \in \mathbb{N}$, and $\Phi \in \mathcal{N}_{d_1,d_2}$ with $\mathcal{L}(\Phi) < L$. There exists a network $\Psi \in \mathcal{N}_{d_1,d_2}$ with $\mathcal{L}(\Psi) = L$, satisfying $\Psi(x) = \Phi(x)$, for all $x \in \mathbb{R}^{d_1}$.*

LEMMA 3.3 (Parallelization of ReLU networks [16]). *Let $n, d, L \in \mathbb{N}$ and, for $i \in \{1, \dots, n\}$, let $d'_i \in \mathbb{N}$ and $\Phi_i \in \mathcal{N}_{d,d'_i}$ with $\mathcal{L}(\Phi_i) = L$. There exists a network $\Psi \in \mathcal{N}_{d,\sum_{i=1}^n d'_i}$ with $\mathcal{L}(\Psi) = L$, satisfying*

$$\Psi(x) = (\Phi_1(x), \dots, \Phi_n(x)) \in \mathbb{R}^{\sum_{i=1}^n d'_i}, \qquad \textit{for all } x \in \mathbb{R}^d.$$

We proceed with the ReLU network constructions realizing operations in the DMV algebra $\mathcal{I}_\mathrm{d} = \langle [0,1], \oplus, \neg, 0, \{\delta_i\}_{i\in\mathbb{N}} \rangle$. To this end, we start by noting that the operation $\neg x = 1 - x$ is trivially implemented by a ReLU network with one layer according to

$$\neg x = W_1(x) = 1 - x, \qquad \text{for } x \in [0,1].$$

By the same token, as the operations $\delta_i x = \frac{1}{i}x$, for $i \in \mathbb{N}$, are affine mappings, they can be realized by single-layer ReLU networks. The following lemma details the ReLU network constructions realizing the operations $x \oplus y = \min\{1, x+y\}$ and $x \odot y = \max\{0, x+y-1\}$ in $\mathcal{I}_\mathrm{d}$.

LEMMA 3.4. *There exist ReLU networks $\Phi^\oplus \in \mathcal{N}_{2,1}$ and $\Phi^\odot \in \mathcal{N}_{2,1}$ satisfying*

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

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

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

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

For the “min” operation, we observe that

$$\min\{1, x\} = 1 - \rho(1-x) = (W_2 \circ \rho \circ W_1)(x), \quad x \in [0,1],$$

where

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

Now, applying Lemma 3.1 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

$$x \oplus y = (W_2^\oplus \circ \rho \circ W_1^\oplus)(x,y), \quad x, y \in [0,1],$$

where

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

To realize the operation $x \odot y = \max\{0, x+y-1\}$, we directly 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

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

⊣

Equipped with the ReLU network realizations of the logical operations underlying $\mathcal{I}_{\mathrm{d}}$, we are now ready to state the following universal representation result.

Proposition 3.5. *Consider the DMV algebra $\mathcal{I}_d$ in Definition 2.6. Let $n \in \mathbb{N}$. For each DMV term $\tau(x_1,\dots,x_n)$ and its associated term function $\tau^{\mathcal{I}_d} : [0,1]^n \to [0,1]$, there exists a ReLU network $\Phi \in \mathcal{N}_{n,1}$ such that*

$$\Phi(x_1,\dots,x_n) = \tau^{\mathcal{I}_d}(x_1,\dots,x_n), \text{ for all } (x_1,\dots,x_n) \in [0,1]^n.$$

Proof. The proof follows by realizing the logical operations appearing in the term function $\tau^{\mathcal{I}_{\mathrm{d}}}$ through corresponding concatenations, according to Lemma 3.1, of ReLU networks implementing the operations $\oplus$ and $\odot$ as by Lemma 3.4 and noting that $\neg x = 1-x, \delta_i x = \frac{1}{i}x$ are trivally ReLU networks with one layer. ⊣

We note that, in general, the ReLU network Φ in Proposition 3.5 will be a properly deep network as it is obtained by concatenating the networks realizing the basic logical operations $\neg$, $\oplus, \odot$, and $\{\delta_i\}_{i\in\mathbb{N}}$.

Now the ground has been prepared for the central result in this section, namely a universal ReLU network realization theorem for CA transition functions. Specifically, this will be effected by combining the connection between CA and DMV algebras established in Section 2 with the ReLU network realizations of the logical operations in DMV algebras presented above.

Theorem 3.6. *Consider a CA with cellular space dimension $d \in \mathbb{N}$, neighborhood size $n \in \mathbb{N}$, state set $K = \{0, \frac{1}{k-1}, \dots, \frac{k-2}{k-1}, 1\}$ of cardinality $k \in \mathbb{N}, k \geq 2$, and transition function $f : K^n \to K$. There exists a ReLU network $\Phi \in \mathcal{N}_{n,1}$ satisfying*

$$\Phi(x_1,\dots,x_n) = f(x_1,\dots,x_n), \qquad \text{for all } (x_1,\dots,x_n) \in K^n.$$

PROOF. By Proposition 2.4, there exists a DMV term $\tau(x_1, \dots, x_n)$ whose corresponding term function under the DMV algebra $\mathcal{I}_{\mathrm{d}}$ satisfies

$$\tau^{\mathcal{I}_{\mathrm{d}}}(x_1, \dots, x_n) = f(x_1, \dots, x_n), \quad \text{for all } (x_1, \dots, x_n) \in K^n. \tag{14}$$

Application of Proposition 3.5 then yields a ReLU network $\Phi \in \mathcal{N}_{n,1}$ such that

$$\Phi(x_1, \dots, x_n) = \tau^{\mathcal{I}_{\mathrm{d}}}(x_1, \dots, x_n), \quad \text{for all } (x_1, \dots, x_n) \in [0,1]^n. \tag{15}$$

The proof is finalized by combining (14) and (15) to get

$$\Phi(x_1, \dots, x_n) = \tau^{\mathcal{I}_{\mathrm{d}}}(x_1, \dots, x_n) = f(x_1, \dots, x_n),$$

for all $(x_1, \dots, x_n) \in K^n$. ⊣

3.2. Realizing the dynamical behavior of CA. We now turn our attention to the dynamical systems aspects of CA. Concretely, we show how CA evolution can be realized through RNNs. The basic idea underlying the construction we present is to suitably impose a recurrent structure on top of the ReLU network realizing the transition function of the CA under consideration. Concretely, we build on the RNN construction techniques developed in [24] and, for simplicity of exposition, consider the case of one-dimensional CA. An extension to the multi-dimensional case is quite readily obtainable using tools from multi-dimensional signal processing [44] and multi-dimensional RNNs [21], [29]. For the sake of simplicity of exposition, however, we do not provide these extensions here.

An RNN is a discrete dynamical system mapping an input sequence to an output sequence, both possibly of infinite length, through—at each discrete time step—application of a feedforward neural network that updates a hidden-state vector and computes the next output signal sample [17]. The formal definition of an RNN is as follows.

DEFINITION 3.2 (Recurrent neural network). For hidden-state vector dimension $m \in \mathbb{N}$, let $\Phi \in \mathcal{N}_{m+1,m+1}$ be a ReLU neural network. The recurrent neural network associated with Φ is the operator $\mathcal{R}_\Phi$ mapping input sequences $(x[z])_{z\in\mathbb{N}_0}$ in $\mathbb{R}$ to output sequences $(y[z])_{z\in\mathbb{N}_0}$ in $\mathbb{R}$ according to

$$\begin{pmatrix} y[z] \\ h[z] \end{pmatrix} = \Phi\left(\begin{pmatrix} x[z] \\ h[z-1] \end{pmatrix}\right), \quad z \in \mathbb{N}_0, \tag{16}$$

where $h[z] \in \mathbb{R}^m$ is the hidden-state vector with initial value $h[-1] = 0_m$.

The key to emulating the evolution of a CA with an RNN is the construction of an appropriate hidden-state vector in combination with a suitable ReLU network that encodes the CA transition function. Before stating the corresponding result, by way of preparation, we introduce a decomposition of Φ in (16) according to

$$\Phi = \begin{pmatrix} \Phi^f \\ \Phi^h \end{pmatrix}, \tag{17}$$

where $\Phi^f \in \mathcal{N}_{m+1,1}$ is responsible for the computation of the output sample according to

$$y[z] = \Phi^f \left(\begin{pmatrix} x[z] \\ h[z-1] \end{pmatrix} \right), \quad z \in \mathbb{N}_0, \tag{18}$$

and $\Phi^h \in \mathcal{N}_{m+1,m}$ effects the evolution of the hidden-state vector such that

$$h[z] = \Phi^h \left(\begin{pmatrix} x[z] \\ h[z-1] \end{pmatrix} \right), \quad z \in \mathbb{N}_0. \tag{19}$$

The following theorem states the announced universal realization theorem for one-dimensional CA by RNNs.

THEOREM 3.7. *Consider a CA with cellular space dimension $d = 1$, neighborhood set $\mathcal{E} = \{0, -1, \dots, -n+1\}$ of size $n \in \mathbb{N}, n \geq 2$, state set $K = \{0, \frac{1}{k-1}, \dots, \frac{k-2}{k-1}, 1\}$ of cardinality $k \in \mathbb{N}, k \geq 2$, and transition function $f : K^n \to K$ with associated CA map F. There exists an RNN that maps every configuration $(c[z])_{z\in\mathbb{Z}}$ over the cellular space $\mathbb{Z}$ to the next configuration $(F(c)[z])_{z\in\mathbb{Z}}$ according to*

$$F(c)[z] = f(c[z], c[z-1], \dots, c[z-n+1]), \quad \forall z \in \mathbb{Z}. \tag{20}$$

Remark. Note that here we do not work with the general neighborhood set $\mathcal{E} = \{0, z_1, \dots, z_{n-1}\} \subset \mathbb{Z}$, but rather consider the concrete neighborhood set $\mathcal{E} = \{0, -1, \dots, -n+1\}$. This does not result in a loss of generality as the following argument shows. Take a general neighborhood set $\{0, z_1, \dots, z_{n-1}\} \subset \mathbb{Z}$ and expand it into a set of the form

$$\{\ell_1, \ell_1 - 1, \dots, \ell_2 + 1, \ell_2\}, \tag{21}$$

where $\ell_1 := \max\{0, z_1, \dots, z_{n-1}\}$ and $\ell_2 := \min\{0, z_1, \dots, z_{n-1}\}$ by, when needed, adding void neighbors that do not affect the CA transition function. For example, consider the CA with neighborhood set $\mathcal{E}_* = \{1, 0, -2\}$, transition function f_*, and corresponding global map

$$F_*(c_*)[z] = f_*(c_*[z+1], c_*[z], c_*[z-2]), \quad \text{for } (c_*[z])_{z\in\mathbb{Z}}.$$

Then, expand $\mathcal{E}_*$ to $\mathcal{E} = \{1, 0, -1, -2\}$ and make f_* formally depend on the void neighbor $z = -1$ according to

$$f(c_*[z+1], c_*[z], c_*[z-1], c_*[z-2]) = \\ f_*(c_*[z+1], c_*[z], c_*[z-2]), \text{ for } (c_*[z])_{z\in\mathbb{Z}}.$$

These modifications lead to the global map

$$F_*(c_*)[z] = f(c_*[z+1], c_*[z], c_*[z-1], c_*[z-2]), \qquad \text{for } (c_*[z])_{z\in\mathbb{Z}}. \tag{22}$$

The general form (20) is finally obtained upon making the substitution

$$c[z] = c_*[z+1], \text{ for all } z \in \mathbb{Z},$$

to yield

$$F(c)[z] = f(c[z], c[z-1], c[z-2], c[z-3]), \qquad \text{for } (c[z])_{z\in\mathbb{Z}}.$$

Proof. For ease of exposition, we provide the proof for one-sided infinite sequences $(c[z])_{z\in\mathbb{N}_0}$ only. The general case follows mutatis mutandis, but is based on exactly the same ideas. The proof is constructive and will be effected by explicitly specifying the ReLU networks Φ^f and Φ^h in (17) underlying the desired RNN. By (20), the output sample $F(c)[z]$, which we identify with $y[z]$, is a function of the states of the cells $\{z, z-1, \ldots, z-n+1\}$. Therefore, equating the current input sample $x[z]$ in (18) with $c[z]$, we choose the hidden-state vector $h[z-1]$ such that it stores the states of the neighbors $\{z-1, \ldots, z-n+1\}$ of the current input cell, i.e.,

$$h[z-1] = \begin{pmatrix} c[z-1] \\ c[z-2] \\ \vdots \\ c[z-n+1] \end{pmatrix}. \tag{23}$$

This leads to

$$h[z] = \begin{pmatrix} c[z] \\ c[z-1] \\ \vdots \\ c[z-n+2] \end{pmatrix}$$

and informs the choice of Φ^h, which must satisfy

$$\Phi^h\left(\begin{pmatrix} c[z] \\ h[z-1] \end{pmatrix}\right) = \Phi^h\left(\begin{pmatrix} c[z] \\ c[z-1] \\ \vdots \\ c[z-n+2] \\ c[z-n+1] \end{pmatrix}\right) = \begin{pmatrix} c[z] \\ c[z-1] \\ \vdots \\ c[z-n+2] \end{pmatrix} = h[z].$$

The evolution of the hidden-state vector hence proceeds by dropping the oldest value $c[z-n+1]$ and inserting the new value $c[z]$ at the top. Following the methodology developed in [24], Φ^h can be realized by a two-layer ReLU network according to

$$\Phi^h : \mathbb{R}^n \to \mathbb{R}^{n-1}, \quad \Phi^h(x) = (W_2^h \circ \rho \circ W_1^h)(x),$$

with affine maps $W_1^h : \mathbb{R}^n \to \mathbb{R}^{2n}$, $W_2^h : \mathbb{R}^{2n} \to \mathbb{R}^{n-1}$ given by

$$\begin{aligned} W_1^h(x) &= \begin{pmatrix} \mathbb{I}_n \\ -\mathbb{I}_n \end{pmatrix} x, \quad \text{for } x \in \mathbb{R}^n, \\ W_2^h(x) &= \begin{pmatrix} \mathbb{I}_{n-1} & 0_{n-1} & -\mathbb{I}_{n-1} & 0_{n-1} \end{pmatrix} x, \quad \text{for } x \in \mathbb{R}^{2n}. \end{aligned} \tag{24}$$

We are left with the design of the network Φ^f which has to satisfy

$$F(c)[z] = \Phi^f \left(\begin{pmatrix} c[z] \\ c[z-1] \\ \vdots \\ c[z-n+1] \end{pmatrix} \right).$$

It follows by inspection of (20) that this amounts to realizing the CA transition function f through the ReLU network Φ^f. Based on this insight, we can apply Theorem 3.6 to conclude the existence of Φ^f. In fact, the proof of Theorem 3.6 spells out how Φ^f can be obtained explicitly by composing the logical operations in the DMV term associated with f. The proof is now completed by noting that the overall network Φ is obtained by applying Lemma 3.2 and Lemma 3.3 to combine Φ^f and Φ^h according to (17). ⊣

§4. Identification of CA logic from trained neural networks.

With suitably chosen transition functions and initial configurations, CA can simulate a plethora of dynamical behavior characteristics [50]. The inverse problem of deducing CA transition functions from observations of their evolution is, however, extremely difficult [4], [1]. Formally, this is known as the CA identification problem [1, Section 1.4]: Given a finite sequence of consecutive configurations $\{c^1, c^2, \dots, c^T\}$ collected during the evolution, construct a CA $(\mathbb{Z}^d, K, \mathcal{E}, f)$ whose associated CA map F satisfies

$$F(c^t) = c^{t+1}, \quad \text{for } t = 1, \dots, T-1.$$

Note that the initial configuration need not be recoverable from observed evolution traces [49, Section 3], [27, Section 4].

Whilst the cellular space $\mathbb{Z}^d$ and the state set K can be read off directly from evolution traces, the neighborhood set $\mathcal{E}$ can either be selected manually [36], [2] or determined through the application of specific criteria such as, e.g., mutual information [53]. The most challenging aspect of the CA identification problem resides in determining the transition function f. A detailed discussion of known CA identification methods can be found in [3].

We next propose a novel approach to CA identification, namely reading out the DMV formula underlying the CA under consideration from an RNN trained on its evolution traces. The focus will be on elucidating the fundamental principles of this idea. Accordingly, we will not be concerned with the performance of specific RNN training algorithms, but will rather assume that the feedforward network part Φ^f inside the RNN has been trained to achieve what is called in machine learning parlance "interpolation", i.e.,

$$\Phi^f(x_1, \dots, x_n) = f(x_1, \dots, x_n), \tag{25}$$

for all $(x_1, \dots, x_n) \in K^n$. This, of course, requires that the RNN being trained have "seen" all possible combinations of neighborhood states and thereby the transition function f on its entire domain K^n, a condition met when the training configuration sequences are sufficiently long and their initial configurations exhibit sufficient richness [1, Thesis 3.1, Thesis 3.2]. We will make this a standing assumption. In addition, the cellular space, the state set, and the neighborhood set are taken to be known a priori.

4.1. Interpolation. The procedure for extracting DMV terms underlying CA evolution data presented in Section 4.3 below works off Φ^f as a (continuous) function mapping $[0,1]^n$ to $[0,1]$. It turns out, however, that condition (25) does not uniquely determine Φ^f on $[0,1]^n$ as there are—in general infinitely many—different ways of interpolating $f(x_1, \dots, x_n)$ to a continuous piecewise linear function; recall that ReLU networks always realize continuous piecewise linear functions. Since Theorem 2.3 states that the truth functions associated with DMV terms under $\mathcal{I}_{\mathrm{d}}$ are continuous piecewise linear functions with rational coefficients, we can constrain the weights of Φ^f to be rational. In fact, as the next two lemmata show, when interpolating CA transition functions, we can tighten this constraint even further, namely to integer weights and rational biases. We first establish that CA transition functions can be interpolated to continuous piecewise linear functions with integer weights and rational biases[4]. Then, it is shown that such functions can be realized by ReLU networks with integer weights and rational biases.

LEMMA 4.1. *Consider a CA with cellular space dimension $d \in \mathbb{N}$, neighborhood size $n \in \mathbb{N}$, state set $K = \{0, \frac{1}{k-1}, \dots, \frac{k-2}{k-1}, 1\}$ of cardinality $k \in \mathbb{N}, k \geq 2$, and transition function $f : K^n \to K$. There exists a function $f_c : [0,1]^n \to [0,1]$ with the following properties:*

1. *$f_c(x_1, \dots, x_n) = f(x_1, \dots, x_n)$, for $(x_1, \dots, x_n) \in K^n$,*
2. *f_c is continuous with respect to the natural topology on $[0,1]^n$,*

[4] To be consistent with terminology used in the context of neural networks, for a linear function of the form $p(x_1, \dots, x_n) = m_1 x_1 + \cdots + m_n x_n + b$, the coefficients m_i will often be referred to as weights and b as bias.

3. there exist $\ell \in \mathbb{N}$ and linear functions

$$p_j(x_1, \dots, x_n) = m_{j1}x_1 + \cdots + m_{jn}x_n + \frac{b_j}{k-1}, \quad j = 1, \dots, \ell, \tag{26}$$

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

Proof. We explicitly construct a function f_{c} satisfying Properties 1-3. For $n = 1$, one simply performs linear interpolation between each pair of points $(\frac{i}{k-1}, f(\frac{i}{k-1}))$ and $(\frac{i+1}{k-1}, f(\frac{i+1}{k-1}))$, $i = 0, \dots, k-2$, to obtain an f_{c} with the desired properties. While it is immediate that the resulting function f_{c} satisfies Properties 1 and 2 and has the structure as demanded by Property 3, the condition $b_j, m_{j1}, \dots, m_{jn} \in \mathbb{Z}$ will be verified summarily, for all n, at the end of the proof.

Before proceeding to the cases $n \geq 2$, we note that throughout the proof an n-simplex, $n \in \mathbb{N}$, with vertices $\{v_0, v_1, \dots, v_n\} \subset \mathbb{R}^n$ will be denoted by $\Delta^n = [v_0, v_1, \dots, v_n]$. We turn to the case $n = 2$ and refer to Figure 6 for an illustration of the domain K^2 of f for $k = 3$. First, we divide $[0,1]^2$ into the squares $[\frac{i_1}{k-1}, \frac{i_1+1}{k-1}] \times [\frac{i_2}{k-1}, \frac{i_2+1}{k-1}]$, $i_1, i_2 \in \{0, \dots, k-2\}$, followed by a subdivision of each of the resulting squares into 2-simplices. There are precisely two possibilities for each of these subdivisions. To illustrate this, we consider the square $[0, \frac{1}{k-1}] \times [0, \frac{1}{k-1}]$ and note that it can be split into two 2-simplices according to either

$$\Delta_1^2 = \left[(0,0), \left(0, \frac{1}{k-1}\right), \left(\frac{1}{k-1}, \frac{1}{k-1}\right)\right]$$
$$\Delta_2^2 = \left[(0,0), \left(\frac{1}{k-1}, 0\right), \left(\frac{1}{k-1}, \frac{1}{k-1}\right)\right]$$

or

$$\Delta_3^2 = \left[(0,0), \left(0, \frac{1}{k-1}\right), \left(\frac{1}{k-1}, 0\right)\right]$$
$$\Delta_4^2 = \left[\left(\frac{1}{k-1}, \frac{1}{k-1}\right), \left(0, \frac{1}{k-1}\right), \left(\frac{1}{k-1}, 0\right)\right],$$

as depicted in Figure 7.

Fixing the subdivisions of all squares, on each 2-simplex $\Delta^2 = [v_0, v_1, v_2]$, we define a linear function p_{Δ^2} that interpolates $f(x)$, i.e., $p_{\Delta^2}(x) = f(x)$, for $x \in \{v_0, v_1, v_2\}$. As $\{v_0, v_1, v_2\}$ is affinely independent, p_{Δ^2} is uniquely determined [41, Chapter 13]. We then stitch the resulting linear pieces together as follows. For each point $x \in [0,1]^2$, if it falls into the interior of some 2-simplex Δ^2, we set $f_{\mathrm{c}}(x) = p_{\Delta^2}(x)$; if x resides in the intersection of two or more 2-simplices, then $f_{\mathrm{c}}(x)$ is taken to have the (shared) value of the interpolating functions associated with the intersecting simplices evaluated at x. The resulting function f_{c} satisfies Properties 1 and 2 and exhibits the structure as demanded by Property 3.

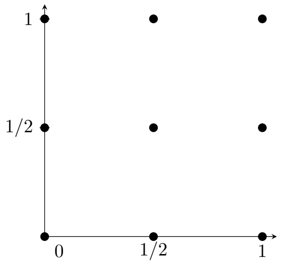

FIGURE 6. The set K^2 for $k = 3$.

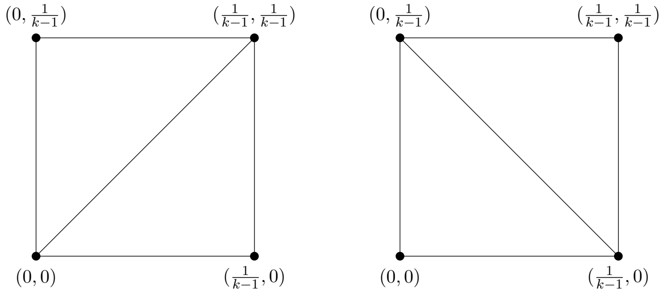

FIGURE 7. Two different subdivisions.

For $n \geq 3$, we first divide the unit cube $[0,1]^n$ into the n-dimensional cubes $[\frac{i_1}{k-1}, \frac{i_1+1}{k-1}] \times \cdots \times [\frac{i_n}{k-1}, \frac{i_n+1}{k-1}], i_1, \ldots, i_n \in \{0, \ldots, k-2\}$. Each of the resulting smaller cubes is then subdivided following the procedure described in [22, Proof of 2.10], which divides, e.g., $\Delta^{n-1} \times [0, \frac{1}{k-1}]$ into n-simplices as follows. Let $\Delta^{n-1} \times \{0\} = [v_0, \ldots, v_{n-1}]$ and $\Delta^{n-1} \times \{\frac{1}{k-1}\} = [w_0, \ldots, w_{n-1}]$ and note that the coordinates of $v_j \in \mathbb{R}^n$ and $w_j \in \mathbb{R}^n$ coincide in the first $n-1$ indices. Then, $\Delta^{n-1} \times [0, \frac{1}{k-1}]$ is given by the union of the n-simplices $[v_0, \ldots, v_j, w_j, \ldots, w_{n-1}]$, $j = 0, \ldots, n-1$, each intersecting the next one in an $(n-1)$-simplex face. The result of this procedure yields a subdivision of $[0,1]^n$ into n-simplices with vertices K^n. As in the case $n = 2$, an interpolating linear function is uniquely determined on each of these n-simplices [41, Chapter 13]; we stitch the resulting linear pieces together to obtain a function f_{c} satisfying Property 1 and exhibiting the structure as demanded by Property 3. As f_{c} constitutes a simplex interpolation of the discrete function f on the cube $[0,1]^n$, it is continuous [39], [48], thereby satisfying Property 2.

We hasten to add that, for $n \geq 2$, the set of n-simplices resulting from the subdivision procedure employed here is not unique, see Figure 7 for an illustration in the case $n = 2$. But, each fixed subdivision into n-simplices uniquely determines a continuous piecewise linear function f_{c} that interpolates f.

It remains to prove that the coefficients of the constituent linear polynomials

$$p_j(x_1,\dots,x_n) = m_{j1}x_1 + \dots + m_{jn}x_n + \frac{b_j}{k-1}, \quad j = 1,\dots,\ell, \tag{27}$$

of f_c, indeed, satisfy $b_j, m_{j1},\dots,m_{jn} \in \mathbb{Z}$, for $j = 1,\dots,\ell$. As already mentioned, the coefficients of each p_j are uniquely determined and obtained by solving a system of linear equations induced by the vertex coordinates of the n-simplex p_j resides on. We now show that the vertices $v_0, v_1,\dots,v_n \subset K^n$ of every n-simplex in $[0,1]^n$ resulting from the procedure described above satisfy, after reordering if necessary,

$$||v_i - v_{i+1}||_1 = \frac{1}{k-1}, \quad \text{for } i = 0,\dots,n-1. \tag{28}$$

In other words, each of the (reordered) vertices differs from the next one in only one position, specifically by $\pm\frac{1}{k-1}$. In the case $n = 1$ the condition (28) holds without reordering. For $n = 2$, no reordering is needed in Δ_1^2 and Δ_2^2, while in each of Δ_3^2 and Δ_4^2 we only need to swap the second vertex with the first one. For general $n \geq 3$, we make use of the fact that, for each n-simplex Δ^n of $[0,1]^n$ constructed according to the procedure described above, there exists an $(n-1)$-simplex $\Delta^{n-1} = [v_0, v_1,\dots,v_{n-1}]$ such that

$$\Delta^n = \left[\begin{pmatrix} v_0 \\ \frac{j}{k-1} \end{pmatrix}, \dots, \begin{pmatrix} v_i \\ \frac{j}{k-1} \end{pmatrix}, \begin{pmatrix} v_i \\ \frac{j+1}{k-1} \end{pmatrix}, \dots, \begin{pmatrix} v_{n-1} \\ \frac{j+1}{k-1} \end{pmatrix}\right],$$

for some $j \in \{0,\dots,k-2\}$ and with $i \in \{0,\dots,n-1\}$. If Δ^{n-1} satisfies (28), it follows directly that Δ^n satisfies (28). Now, noting that for $n = 3$, Δ^2 satisfies (28) as already established, and proceeding by induction across n establishes the desired statement.

Finally, assume that the linear piece

$$m_1x_1 + m_2x_2 + \dots + m_nx_n + m_0$$

of f_c is obtained, w.l.o.g. thanks to (28), by interpolating $n+1$ points at

$$\left(\frac{a_1}{k-1}, \frac{a_2}{k-1}, \dots, \frac{a_n}{k-1}, \frac{b_0}{k-1}\right)$$
$$\left(\frac{a_1+1}{k-1}, \frac{a_2}{k-1}, \dots, \frac{a_n}{k-1}, \frac{b_1}{k-1}\right)$$
$$\left(\frac{a_1+1}{k-1}, \frac{a_2+1}{k-1}, \dots, \frac{a_n}{k-1}, \frac{b_2}{k-1}\right)$$
$$\vdots$$
$$\left(\frac{a_1+1}{k-1}, \frac{a_2+1}{k-1}, \dots, \frac{a_n+1}{k-1}, \frac{b_n}{k-1}\right),$$

with $a_1, \ldots, a_n, b_0, \ldots, b_n \in \{0, 1, \ldots, k-1\}$. The coefficients $m_0, m_1, \ldots, m_n$ hence have to satisfy the following system of linear equations

$$
\begin{aligned}
\frac{a_1}{k-1}m_1 + \frac{a_2}{k-1}m_2 + \cdots + \frac{a_n}{k-1}m_n + m_0 &= \frac{b_0}{k-1} \\
\frac{a_1+1}{k-1}m_1 + \frac{a_2}{k-1}m_2 + \cdots + \frac{a_n}{k-1}m_n + m_0 &= \frac{b_1}{k-1} \\
\frac{a_1+1}{k-1}m_1 + \frac{a_2+1}{k-1}m_2 + \cdots + \frac{a_n}{k-1}m_n + m_0 &= \frac{b_2}{k-1} \\
&\vdots \\
\frac{a_1+1}{k-1}m_1 + \frac{a_2+1}{k-1}m_2 + \cdots + \frac{a_n+1}{k-1}m_n + m_0 &= \frac{b_n}{k-1}.
\end{aligned} \tag{29}
$$

Subtracting the first row in (29) from the second and multiplying the result by $k-1$, we obtain

$$m_1 = b_1 - b_0 \in \mathbb{Z}.$$

Continuing likewise by subtracting the first row from the third yields

$$m_2 = b_2 - b_0 - m_1 \in \mathbb{Z}$$

and so forth, establishing that $m_i \in \mathbb{Z}$, for all $i = 1, 2, \ldots, n$. Substituting back into any of the equations in (29) establishes that m_0 is of the desired form as well and thereby finalizes the proof. ⊣

We next show that functions f_c satisfying Properties 1-3 in Lemma 4.1 can, indeed, be realized by ReLU networks with integer weights and biases in $\mathbb{Q}_k := \{\frac{b}{k-1} : b \in \mathbb{Z}\}$.

Lemma 4.2. *Let $n \in \mathbb{N}$ and let $f_c : [0,1]^n \to [0,1]$ be a continuous piecewise linear function with linear pieces $p_1, \ldots, p_\ell, \ell \in \mathbb{N}$, of the form*

$$p_j(x_1, \ldots, x_n) = m_{j1}x_1 + \cdots + m_{jn}x_n + \frac{b_j}{k-1}, \quad j = 1, \ldots, \ell, \tag{30}$$

where $k \geq 2$ and $m_{j1}, \ldots, m_{jn}, b_j \in \mathbb{Z}$, for $j = 1, \ldots, \ell$. There exists a ReLU network $\Phi \in \mathcal{N}_{n,1}$ with integer weights and biases in $\mathbb{Q}_k$, satisfying $\Phi(x) = f_c(x)$, for all $x \in [0,1]^n$.

Proof. By Lemma D.1, the function f_c can be written in terms of $p_1, \ldots, p_\ell$ through "min" and "max" operations in the form

$$f_\mathrm{c} = \max_I \min_J p_j, \tag{31}$$

where $I, J \subset \{1, \ldots, \ell\}$ are index sets. The linear pieces p_j, for $j = 1, \ldots, \ell$, in (30) can straightforwardly be realized by single-layer ReLU networks with integer weights m_{ji} and biases $b_j/(k-1)$. Next, we note that the "min" operation can be implemented by a ReLU network with integer weights and biases equal to 0 according to

$$\min\{x_1, x_2\} = \rho(x_1) - \rho(-x_1) - \rho(x_1 - x_2), \text{ for } x_1, x_2 \in \mathbb{R}.$$

As

$$\min\{x_1, x_2, x_3\} = \min\{x_1, \min\{x_2, x_3\}\}, \text{ for } x_1, x_2, x_3 \in \mathbb{R},$$

it follows that the minimum over any number of variables can be realized by nesting minimum operations. Hence, inspection of the proof of [16, Lemma II.3] reveals that there exists a ReLU network with integer weights and biases equal to 0 computing the minimum over any number of variables. Likewise, we have

$$\max\{x_1, x_2\} = \rho(x_1) - \rho(-x_1) + \rho(x_2 - x_1), \text{ for } x_1, x_2 \in \mathbb{R},$$

and

$$\max\{x_1, x_2, x_3\} = \max\{x_1, \max\{x_2, x_3\}\}, \text{ for } x_1, x_2, x_3 \in \mathbb{R},$$

which analogously establishes that the operation taking the maximum over any number of variables can be realized by a ReLU network with integer weights and biases equal to 0. Based on (31), f_{c} can be written as the composition of ReLU networks with integer weights and biases either equal to 0 (for the min and max operations) or equal to $b_j/(k-1)$ (for the p_j). Inspection of the proof of [16, Lemma II.3] then shows that the resulting overall ReLU network will have integer weights and biases in $\mathbb{Q}_k$, as desired. $\dashv$

The proof of Lemma 4.1, combined with Lemma 4.2, reveals that even if we constrain the network Φ^f interpolating the transition function f to have integer weights and biases in $\mathbb{Q}_k$, the network will not be unique. This is a consequence of the n-simplices underlying the linear pieces p_j constructed in the proof of Lemma 4.1 not being unique, see e.g. Figure 7 for an illustration of this phenomenon in the case $n = 2$. Correspondingly the DMV formulae extracted from the networks realizing the functions f_{c} based on different choices of n-simplices, following the procedure described in Section 4.3 below, will be functionally different. Beyond this nonuniqueness, we can even get interpolating ReLU networks with integer weights and biases in $\mathbb{Q}_k$ that do not effect simplex interpolation. This aspect is illustrated in the next example.

Example 4.1. Consider a CA with cellular space dimension $d = 1$, neighborhood size $n = 2$, state set $\{0, 1/2, 1\}$ with $k = 3$, and transition function $f : \{0, 1/2, 1\}^2 \to \{0, 1/2, 1\}$ specified as follows:

$x_{-1}\, x_0$	$(0,0)$	$(0,1/2)$	$(0,1)$	$(1/2,0)$	$(1/2,1/2)$
$f(x_{-1},\, x_0)$	0	0	0	0	1

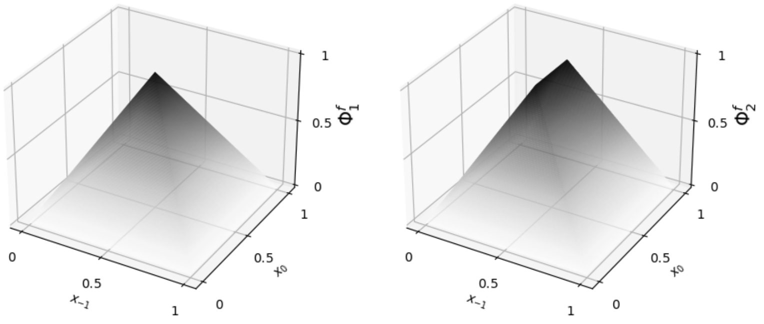

FIGURE 8. Different ReLU networks interpolating f.

$x_{-1}\,x_0$	$(1/2, 1)$	$(1, 0)$	$(1, 1/2)$	$(1, 1)$
$f(x_{-1},\, x_0)$	0	0	0	0

Figure 8 shows two different ReLU networks $\Phi_1^f, \Phi_2^f : [0,1]^2 \to [0,1]$, both with integer weights and biases in $\mathbb{Q}_k$, interpolating the transition function. We remark that Φ_1^f effects simplex interpolation, while Φ_2^f does not. Now, applying the DMV formula extraction algorithm introduced in Section 4.3 below yields the DMV term associated with Φ_1^f as

$$\tau_1 = (x_{-1} \oplus x_{-1}) \wedge (\neg x_{-1} \oplus \neg x_{-1}) \wedge (x_0 \oplus x_0) \wedge (\neg x_0 \oplus \neg x_0),$$

and that associated with Φ_2^f according to

$$\tau_2 = (x_{-1} \oplus x_{-1}) \wedge (\neg x_{-1} \oplus \neg x_{-1}) \wedge (x_0 \oplus x_0 \oplus x_0) \wedge (\neg x_0 \oplus \neg x_0 \oplus \neg x_0).$$

These two DMV terms are algebraically and functionally different, but the associated term functions under $\mathcal{I}_\mathrm{d}$ coincide on K^2, i.e.,

$$\tau_1^{\mathcal{I}_\mathrm{d}}(x_{-1}, x_0) = \tau_2^{\mathcal{I}_\mathrm{d}}(x_{-1}, x_0) = f(x_{-1}, x_0),$$

for $(x_{-1}, x_0) \in \{0, 1/2, 1\}^2$.

We shall consider all term functions that coincide with a given CA transition function on K^n to constitute an equivalence class in the sense of faithfully describing the logical behavior of the underlying CA.

There are two further sources of potential nonuniqueness discussed next.

4.2. Uniqueness properties of DMV formula extraction. We first comment on a structural property pertaining to the RNN emulating the CA evolution. Recall the decomposition of Φ in (17). Our construction built on the idea of having the hidden-state vector store the neighbors of the current input cell, leading to the specific form of Φ^h, and making the subnetwork Φ^f be responsible for the computation of the output samples. But this separation need not be the only way the RNN can realize the CA evolution. The resulting ambiguity is, however, easily eliminated by enforcing the split of Φ according to (17) on the RNN to be trained. In practice, this means that the weights of Φ^h are fixed according to (24) and only the weights of Φ^f are trained. In the light of Lemma 4.1 and Lemma 4.2, the training algorithm has to enforce integer weights and biases in $\mathbb{Q}_k$ on Φ^f. This can be done using techniques such as those described in e.g. [51], [23]. At the end of the training process, the DMV formula underlying the CA can be read out from Φ^f using the algorithm presented in Section 4.3 below. Recall that given what was said after Lemma 4.2, this formula will not be unique in general.

The second aspect we wish to dwell on here revolves around the fact that algebraically different DMV formulae can be functionally equivalent, just like what one has in classical algebra. To demonstrate the matter, we analyze a simple example in Boolean algebra, namely the elementary CA of rule 30 with transition function specified in DNF in (2). A functionally equivalent, but algebraically different, expression for this transition rule is given by

$$f_{30} = (x_{-1} \odot \neg x_0 \odot \neg x_1) \oplus (\neg x_{-1} \odot x_0 \odot x_1) \oplus \\ (\neg x_{-1} \odot x_0 \odot \neg x_1) \oplus (\neg x_{-1} \odot \neg x_0 \odot x_1). \tag{32}$$

This shows that even if we restrict ourselves to DNF, the algebraic expression will not be unique in general. Moreover, we can also express rule 30 in conjunctive normal form (CNF), i.e., as a concatenation of a finite number of clauses linked by the Boolean $\odot$ operation, where each clause consists of a finite number of variables or negated variables connected by the Boolean $\oplus$ operation, e.g.,

$$f_{30} = (x_{-1} \oplus x_0 \oplus x_1) \odot (\neg x_{-1} \oplus \neg x_1) \odot (\neg x_{-1} \oplus \neg x_0), \tag{33}$$

or

$$f_{30} = (x_{-1} \oplus x_0 \oplus x_1) \odot (\neg x_{-1} \oplus \neg x_1) \odot (\neg x_{-1} \oplus \neg x_0 \oplus x_1). \tag{34}$$

If we do not limit ourselves to normal forms, starting from an expression f_{30} (with x_1 as one of its variables) for the transition function of CA 30, equivalent expressions for f_{30} are given by

$$f_{30} \oplus (x_1 \odot \neg x_1), \tag{35}$$

and

$$f_{30} \odot (x_1 \oplus \neg x_1). \tag{36}$$

Now, noting that the ReLU network realization of f_{30} as expressed in (32), is given by

$$\hat{\Phi}^{f_{30}} := \hat{W}_3^{f_{30}} \circ \rho \circ \hat{W}_2^{f_{30}} \circ \rho \circ \hat{W}_1^{f_{30}},$$

with

$$\hat{W_1}^{f_{30}}(x_{-1}, x_0, x_1) = \begin{pmatrix} -1 & -1 & 1 \\ 1 & 1 & -1 \\ -1 & 1 & -1 \\ 1 & -1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_0 \\ x_{-1} \end{pmatrix} + \begin{pmatrix} 0 \\ -1 \\ 0 \\ 0 \end{pmatrix},$$

$$\hat{W_2}^{f_{30}}(x) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} x + \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \quad x \in \mathbb{R}^4$$

$$\hat{W_3}^{f_{30}}(x) = \begin{pmatrix} 1 & -1 \end{pmatrix} x, \quad x \in \mathbb{R}^2,$$

we can conclude, by comparing to the network (7) built based on (2), that different algebraic expressions for f_{30} lead to different ReLU network realizations. Yet, these two networks must exhibit identical input-output relations. Conversely, the observation just made shows that ReLU networks can be modified without changing their input-output relations.

4.3. Extracting DMV formulae from trained networks. We now discuss how DMV formulae can be read out from the network Φ^f. Recall that the idea is that Φ^f was trained on CA evolution data and the extracted DMV formula describes the logical behavior underlying this CA. In the Boolean case, with $K = \{0, 1\}$, the truth table representing the CA transition function can be obtained by passing all possible neighborhood state combinations through the trained network Φ^f and recording the corresponding outputs. Following, e.g., the procedure in [38, Section 12.2], this truth table can then be turned into a Boolean formula. For state sets K of cardinality larger than two, we are not aware of systematic procedures that construct DMV terms from truth tables. The approach we develop here applies to state sets of arbitrary cardinality and does not work off truth tables, but rather starts from a (trained) ReLU network Φ^f that, by virtue of satisfying the interpolation property (25), realizes a linear interpolation of the truth table.

We start by noting that Φ^f encodes its underlying DMV formula both through the network topology and the network weights and biases. Honoring how the network architecture, i.e., layer-by-layer compositions of affine mappings and the ReLU nonlinearity, affects the extracted DMV term, we aim to proceed in a decompositional manner and on a node-by-node basis. It turns out, however, that the individual neurons, in general, will not correspond to DMV terms. To see this, recall that DMV term functions map $[0, 1]^n$ to $[0, 1]$ and note that the range of the function ρ is not in $[0, 1]$ in general, e.g., $\rho(3x) : [0, 1] \to [0, 3]$. We will deal with this matter by transforming the ρ-neurons in Φ^f into σ-neurons with a suitably chosen $\sigma : \mathbb{R} \to [0, 1]$. This will be done in a manner that preserves the network's input-output relation and is reversible in a sense to be made precise below. We start by defining σ-networks.

Definition 4.1 (σ-network). Let $L \in \mathbb{N}$ and $N_0, N_1, \dots, N_L \in \mathbb{N}$. A σ-network Ψ is a map $\Psi : \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ given by

$$\Psi = \begin{cases} W_1, & L = 1 \\ W_2 \circ \sigma \circ W_1, & L = 2 \\ W_L \circ \sigma \circ W_{L-1} \circ \sigma \circ \cdots \circ \sigma \circ W_1, & L \geq 3 \end{cases}, \tag{37}$$

where, for $\ell \in \{1, 2, \dots, L\}$, $W_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}, W_\ell(x) := A_\ell x + b_\ell$ 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 activation function $\sigma : \mathbb{R} \to [0,1], \sigma(x) := \min\{1, \max\{0, x\}\}$ acts component-wise. Moreover, we define the depth of the network Ψ as $\mathcal{L}(\Psi) := L$.

We proceed to formally establish the result on ρ-network to σ-network transformation announced above.

Lemma 4.3. *For $n \in \mathbb{N}$, let $\Phi : [0,1]^n \to [0,1]$ be a ReLU network with integer weights and biases in $\mathbb{Q}_k$, with $k \geq 2$. There exists a σ-network $\Psi : [0,1]^n \to [0,1]$, with $\mathcal{L}(\Psi) = \mathcal{L}(\Phi)$, integer weights, and biases in $\mathbb{Q}_k$, satisfying*

$$\Psi(x) = \Phi(x), \quad \textit{for all } x \in [0,1]^n.$$

Proof. By Definition 3.1, we can write[5]

$$\Phi = W_L \circ \rho \circ W_{L-1} \circ \rho \circ \cdots \circ \rho \circ W_1,$$

with $N_L = 1, N_0 = n, W_\ell(x) = A_\ell x + b_\ell, A_\ell \in \mathbb{Z}^{N_\ell \times N_{\ell-1}}$, and $b_\ell \in \mathbb{Q}_k^{N_\ell}$, all for $\ell \in \{1, \dots, L\}$. We start with the conversion of neurons in the first hidden layer of Φ. These neurons are all of the form

$$\rho(a_1 x + b_1), \quad \text{with } a_1 \in \mathbb{Z}^{1 \times n}, b_1 \in \mathbb{Q}_k.$$

As the domain of Φ is the unit cube $[0,1]^n$, $a_1 x + b_1$ resides in a bounded interval in $\mathbb{R}$. Denote this interval by $[-u, u] \subset \mathbb{R}, u \geq 0$. If $u \leq 1$, no conversion is needed as $\rho(x') = \sigma(x')$, for $x' \in [-u, u]$. For $u > 1$, we note that

$$\rho(x) = \sigma(x) + \sigma(x-1) + \cdots + \sigma(x-m), \quad x \in [-u, u], \tag{38}$$

with $m \in \mathbb{N}, m \geq u - 1$. In summary, we can replace each ρ-neuron in the first hidden layer by one or several σ-neurons according to

$$\rho(a_1 x + b_1) = \sigma(a_1 x + b_1)$$

whenever $a_1 x + b_1 \leq 1$, for all $x \in [0,1]^n$, or

$$\rho(a_1 x + b_1) = \sigma(a_1 x + b_1) + \cdots + \sigma(a_1 x + b_1 - m)$$

[5] To keep the exposition simple, we consider the form of Φ for $L \geq 3$, the cases $L = 1, 2$ are trivially contained in the discussion.

when $a_1 x + b_1 \leq m + 1$, for all $x \in [0,1]^n$, for some $m > 1, m \in \mathbb{N}$. The resulting network $\Phi^{(1)} : [0,1]^n \to [0,1]$ is given by

$$\Phi^{(1)} = W_L \circ \rho \circ \cdots \circ W_3 \circ \rho \circ W_2^{(1)} \circ \sigma \circ W_1^{(1)},$$

with $W_1^{(1)}(x) = A_1^{(1)} x + b_1^{(1)}, A_1^{(1)} \in \mathbb{Z}^{N_1^{(1)} \times n}, b_1^{(1)} \in \mathbb{Q}_k^{N_1^{(1)}}$, and $W_2^{(1)}(x) = A_2^{(1)} x + b_2, A_2^{(1)} \in \mathbb{Z}^{N_2 \times N_1^{(1)}}$, where $N_1^{(1)}$ is the number of σ-neurons in the first hidden layer of $\Phi^{(1)}$, and satisfies

$$\Phi^{(1)}(x) = \Phi(x), \text{ for } x \in [0,1]^n.$$

Noting that the input of the second-hidden layer of $\Phi^{(1)}$ is contained in the unit cube $[0,1]^{N_1^{(1)}}$, we can proceed as in the first step and replace each of the ρ-neurons in the second hidden layer by σ-neurons. The resulting network is given by

$$\Phi^{(2)} = W_L \circ \cdots \circ W_4 \circ \rho \circ W_3^{(2)} \circ \sigma \circ W_2^{(2)} \circ \sigma \circ W_1^{(1)},$$

with $W_2^{(2)}(x) = A_2^{(2)} x + b_2^{(2)}, A_2^{(2)} \in \mathbb{Z}^{N_2^{(2)} \times N_1^{(1)}}, b_2^{(2)} \in \mathbb{Q}_k^{N_2^{(2)}}$, and $W_3^{(2)}(x) = A_3^{(2)} x + b_3, A_3^{(2)} \in \mathbb{Z}^{N_3 \times N_2^{(2)}}$, where $N_2^{(2)}$ is the number of σ-neurons in the second hidden layer of $\Phi^{(2)}$, and satisfies

$$\Phi^{(2)}(x) = \Phi(x), \text{ for } x \in [0,1]^n.$$

Continuing in this manner, we eventually get the network $\Phi^{(L-1)} : [0,1]^n \to [0,1]$ given by

$$\Phi^{(L-1)} = W_L^{(L-1)} \circ \sigma \circ W_{L-1}^{(L-1)} \circ \sigma \circ \cdots \circ \sigma \circ W_2^{(2)} \circ \sigma \circ W_1^{(1)},$$

with $W_\ell^{(\ell)}(x) = A_\ell^{(\ell)} x + b_\ell^{(\ell)}, A_\ell^{(\ell)} \in \mathbb{Z}^{N_\ell^{(\ell)} \times N_{\ell-1}^{(\ell-1)}}, b_\ell^{(\ell)} \in \mathbb{Q}_k^{N_\ell^{(\ell)}}$, where $N_\ell^{(\ell)}$ is the number of (σ-)neurons in the ℓ-th hidden layer of $\Phi^{(L-1)}$, for $\ell \in \{1, \ldots, L-1\}$, and $W_L^{(L-1)}(x) = A_L^{(L-1)} x + b_L, A_L^{(L-1)} \in \mathbb{Z}^{N_L \times N_{L-1}^{(L-1)}}$. The proof is concluded by noting that the resulting σ-network $\Psi := \Phi^{(L-1)}$ satisfies $\Psi(x) = \Phi(x)$, for $x \in [0,1]^n$, and has integer weights and biases in $\mathbb{Q}_k$. $\dashv$

The reverse transformation from a σ-network to a ρ-network can be effected through Lemma B.1. Starting from a ReLU nework Φ, transforming it into a σ-network Ψ according to Lemma 4.3, and then going back to a ReLU network Φ' by using Lemma B.1, will recover a network Φ' that is functionally equivalent to Φ, i.e., $\Phi'(x) = \Phi(x)$, for all $x \in [0,1]^n$, but can be structurally different from Φ. For example, choosing different $m \in \mathbb{N}$ with $m \geq u - 1$ in (38) would lead to such functionally equivalent, but structurally different recovered networks Φ'.

We are now ready to show how DMV terms can be extracted from σ-networks. Thanks to Lemma 4.3, we can restrict ourselves to σ-networks with integer weights and biases in $\mathbb{Q}_k$. We begin by considering a single σ-neuron, which is of the form

$$\sigma(m_1x_1 + \cdots + m_nx_n + b), \qquad \text{with } m_1, \ldots, m_n \in \mathbb{Z}, b \in \mathbb{Q}_k. \tag{39}$$

The following lemma provides an inductive way for the extraction of DMV terms from individual σ-neurons.

Lemma 4.4. *Consider the function* $f(x_1, \ldots, x_n) = m_1x_1 + \cdots + m_nx_n + b$, *defined on* $[0, 1]^n$, *with* $m_1, \ldots, m_n \in \mathbb{Z}, b \in \mathbb{Q}_k$. *Without loss of generality, assume that* $\max_{i=1,\ldots,n} |m_i| = m_1$. *With* $f_\circ(x_1, \ldots, x_n) = (m_1 - 1)x_1 + m_2x_2 + \cdots + m_nx_n + b$, *we have*

$$\sigma(f) = (\sigma(f_\circ) \oplus x_1) \odot \sigma(f_\circ + 1). \tag{40}$$

Proof. See Appendix C [37], [33]. ⊣

We next demonstrate, by way of a simple example, how Lemma 4.4 can be used to extract DMV terms from individual σ-neurons.

Example 4.2. We extract the DMV term underlying $\sigma(x - 2y + 1/2)$ and start by noting that

$$\begin{aligned} & \sigma(x - 2y + 1/2) && (41)\\ &= 1 - \sigma(-x + 2y - 1/2 + 1) && (42)\\ &= \neg\sigma(-x + 2y + 1/2) && (43)\\ &= \neg((\sigma(-x + y + 1/2) \oplus y) \odot \sigma(-x + y + 3/2)), && (44) \end{aligned}$$

where in (42) we used $\sigma(x) = 1 - \sigma(-x + 1)$, for $x \in \mathbb{R}$, and (43) is by $\neg x = 1 - x$. In (44) we applied Lemma 4.4 with $x_1 = y$. We again apply Lemma 4.4 with $x_1 = y$ to remove the y-terms inside $\sigma(\cdot)$ as follows:

$$\begin{aligned} \sigma(-x + y + 1/2) &= (\sigma(-x + 1/2) \oplus y) \odot \sigma(-x + 3/2)\\ \sigma(-x + y + 3/2) &= (\sigma(-x + 3/2) \oplus y) \odot \sigma(-x + 5/2). \end{aligned} \tag{45}$$

Likewise, we can remove the x-terms inside $\sigma(\cdot)$ according to

(46)
$$\begin{aligned} \sigma(-x + 1/2) &= \neg\sigma(x + 1/2) = \neg((\sigma(1/2) \oplus x) \odot \sigma(3/2)) = \neg(1/2 \oplus x)\\ \sigma(-x + 3/2) &= \neg\sigma(x - 1/2) = \neg((\sigma(-1/2) \oplus x) \odot \sigma(1/2)) = \neg(x \odot 1/2)\\ \sigma(-x + 5/2) &= \neg\sigma(x - 3/2) = \neg((\sigma(-3/2) \oplus x) \odot \sigma(-1/2)) = 1. \end{aligned}$$

For the constant term 1/2, by Definition 2.6, we can write

$$1/2 = \delta_2 1.$$

Substituting (46) back into (45) and then substituting the results thereof into (44), we obtain the DMV term corresponding to $\sigma(x - 2y + 1/2)$ according to

$$\neg((((\neg(\delta_2 1 \oplus x) \oplus y) \odot \neg(x \odot \delta_2 1)) \oplus y) \odot (\neg(x \odot \delta_2 1) \oplus y)). \tag{47}$$

We conclude the example by noting that the algebraic expression of the DMV term extracted using this procedure depends on the order in which the variables are eliminated through Lemma 4.4; functionally these expressions will, however, all be equivalent.

We are now ready to describe how the DMV term underlying a given ReLU network with integer weights and biases in $\mathbb{Q}_k$ mapping $[0,1]^n$ to $[0,1]$ can be extracted. The corresponding algorithm starts by applying Lemma 4.3 to convert the network into a functionally equivalent σ-network

$$\Psi^f = W_L \circ \sigma \circ \cdots \circ W_2 \circ \sigma \circ W_1,$$

where $W_1, \ldots, W_L$ are affine transformations with integer weights and biases in $\mathbb{Q}_k$. As the range of Ψ^f is contained in $[0,1]$, we can apply the σ-activation function to the output layer without changing the mapping, i.e.,

$$\Psi^f = \sigma \circ W_L \circ \cdots \circ \sigma \circ W_2 \circ \sigma \circ W_1.$$

Next, Lemma 4.4 is applied to each layer $\sigma \circ W_\ell$, $\ell = 1, \ldots, L$, as in Example 4.2 above to extract the DMV terms corresponding to the individual output neurons in layer ℓ. The resulting DMV formulae are then algebraically composed honoring the compositional structure of Ψ^f to yield the overall DMV term corresponding to the ReLU network we started from. We note that in order to have the compositional structure of ReLU networks correspond to compositions in DMV algebra as just explained, it is crucial to first transform the ReLU network into a (functionally equivalent) σ-network according to Lemma 4.3. This is, as mentioned above, to ensure that the outputs of neurons inside the network are guaranteed to be in $[0,1]$ so as to remain in the set $([0,1])$ underlying the DMV algebra. We have automated our extraction algorithm in Python software[6]; the algorithm takes a ReLU network as input and outputs a corresponding DMV formula. Applying this algorithm, for example, to the ReLU network $f_c = \rho(x_{-1} + x_0 + x_1) - \rho(x_{-1} + x_0 + x_1 - 1)$ linearly interpolating the transition function in Table 4, indeed, recovers the MV term $\tau = x_{-1} \oplus x_0 \oplus x_1$ we started from in Example 2.2.

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

[6] Implementation available at `https://www.mins.ee.ethz.ch/research/downloads/NN2MV.html`

Appendix A. MV term corresponding to f_c in (13)**.** First, expand the domains of the linear pieces of f_c to $[0,1]$ and denote them by f_1, f_2, f_3, with

$$\begin{aligned} f_1(x) &= 3x, \\ f_2(x) &= 1, \\ f_3(x) &= -3x+3, \end{aligned}$$

for $x \in [0,1]$. Since $f_c : [0,1] \to [0,1]$, we can compose each linear piece with $\sigma(\cdot)$, which results in the truncated linear functions $\sigma(f_1), \sigma(f_2), \sigma(f_3)$ depicted in Figure 9. Next, we apply Lemma 4.4 to extract the MV terms corresponding to $\sigma(f_1), \sigma(f_2), \sigma(f_3)$. Trivally, $\sigma(f_2) = \neg 0$ on $[0,1]$. The function $\sigma(f_1)$ can be expressed as

$$\sigma(f_1)(x) = x \oplus x \oplus x, \text{ for } x \in [0,1].$$

Likewise, we have

$$\sigma(f_3)(x) = x \odot x \odot x, \text{ for } x \in [0,1].$$

Inspection of Figure 9 and f_c in Figure 4 allows us to conclude that

$$f_c(x) = \min\{\sigma(f_1)(x), \sigma(f_2)(x), \sigma(f_3)(x)\}, \text{ for } x \in [0,1].$$

Finally, it is readily verified that the "min" operation can be realized according to

$$\min\{x,y\} = x \wedge y,$$

and

$$\min\{x,y,z\} = \min\{\min\{x,y\}, z\}.$$

Putting the pieces together, we obtain the MV term corresponding to f_c as

$$(x \oplus x \oplus x) \wedge \neg 0 \wedge (x \odot x \odot x).$$

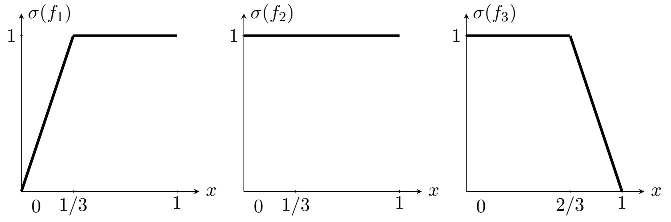

FIGURE 9. The truncated functions $\sigma(f_1), \sigma(f_2), \sigma(f_3)$.

Appendix B. Transforming σ-networks into equivalent ReLU networks.

LEMMA B.1. *For $d, d' \in \mathbb{N}$, let $\Psi : \mathbb{R}^d \to \mathbb{R}^{d'}$ be a σ-network with integer weights and biases in $\mathbb{Q}_k$. There exists a ReLU network $\Phi : \mathbb{R}^d \to \mathbb{R}^{d'}$, with $\mathcal{L}(\Phi) = \mathcal{L}(\Psi)$, integer weights, and biases in $\mathbb{Q}_k$, satisfying*

$$\Phi(x) = \Psi(x), \qquad \textit{for all } x \in \mathbb{R}^d.$$

PROOF. By Definition 4.1, we can write[7]

$$\Psi = W_L \circ \sigma \circ W_{L-1} \circ \sigma \circ \cdots \circ \sigma \circ W_1,$$

with $W_\ell(x) = A_\ell x + b_\ell, A_\ell \in \mathbb{Z}^{N_\ell \times N_{\ell-1}}, b_\ell \in \mathbb{Q}_k^{N_\ell}$, for $\ell \in \{1, \dots, L\}$, $N_0 = d$, and $N_L = d'$. The basic idea of the proof is to use the relationship

$$\sigma(x) = \rho(x) - \rho(x-1), \text{ for all } x \in \mathbb{R}, \tag{48}$$

to replace every σ-neuron in Ψ with a pair of ρ-neurons. We start with $\sigma \circ W_1$ to obtain the equivalent network

$$\Psi^{(1)} = W_L \circ \sigma \circ \cdots \circ \sigma \circ W_2 \circ H_1^{(1)} \circ \rho \circ W_1^{(1)},$$

where

$$W_1^{(1)}(x) := \begin{pmatrix} W_1(x) \\ W_1(x) - 1_{N_1} \end{pmatrix}, \quad x \in \mathbb{R}^{N_0},$$
$$H_1^{(1)}(x) := \begin{pmatrix} \mathbb{I}_{N_1} & -\mathbb{I}_{N_1} \end{pmatrix} x, \quad x \in \mathbb{R}^{2N_1},$$

with 1_{N_1} denoting the N_1-dimensional column vector with all entries equal to 1. It follows directly that $\Psi^{(1)}$ has integer weights and biases in $\mathbb{Q}_k$. Continuing in this manner, we get

$$\Psi^{(L-1)} = W_L \circ H_{L-1}^{(L-1)} \circ \rho \circ W_{L-1}^{(L-1)} \circ \cdots \circ W_2^{(2)} \circ H_1^{(1)} \circ \rho \circ W_1^{(1)},$$

with

$$W_\ell^{(\ell)}(x) = \begin{pmatrix} W_\ell(x) \\ W_\ell(x) - 1_{N_\ell} \end{pmatrix}, \quad x \in \mathbb{R}^{N_{\ell-1}},$$
$$H_\ell^{(\ell)}(x) = \begin{pmatrix} \mathbb{I}_{N_\ell} & -\mathbb{I}_{N_\ell} \end{pmatrix} x, \quad x \in \mathbb{R}^{2N_\ell},$$

for $\ell = 1, \dots, L-1$, satisfying

$$\Psi^{(L-1)}(x) = \Psi(x), \text{ for all } x \in \mathbb{R}^d.$$

The proof is concluded upon identifying $\Psi^{(L-1)}$ with Φ and noting that $\Psi^{(L-1)}$ has integer weights and biases in $\mathbb{Q}_k$ and $\mathcal{L}(\Psi^{(L-1)}) = \mathcal{L}(\Psi)$. $\dashv$

[7] To keep the exposition simple, we consider the form of Ψ for $L \geq 3$, the cases $L = 1, 2$ are trivially contained in the discussion.

Appendix C. Proof of Lemma 4.4 [37], [33].

PROOF. We follow the line of arguments in [33] 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 (40) 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 is

$$\sigma(f) = 0.$$

The RHS is given by

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

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 (40) 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 becomes

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

⊣

Appendix D.

LEMMA D.1. *Let $n \in \mathbb{N}$ and let $f_c : [0,1]^n \to [0,1]$ be a continuous piecewise linear function with linear pieces*

$$p_j(x_1, \ldots, x_n) = m_{j1}x_1 + \cdots + m_{jn}x_n + b_j, \quad j = 1, \ldots, \ell, \tag{49}$$

where $m_{j1}, \dots, m_{jn} \in \mathbb{R}$, $b_j \in \mathbb{R}$. *Let* Π *be the set of all permutations of the index set* $\{1, \dots, \ell\}$. *There exist a nonempty set of permutations* $\Sigma \subset \Pi$ *and, for each* $\pi \in \Sigma$, *an integer* i_π *with* $1 \leq i_\pi \leq \ell$, *such that*

$$f_c(x) = \max_{\pi \in \Sigma} \min_{i \in \{1,\dots,i_\pi\}} p_{\pi(i)}(x), \text{ for all } x \in [0,1]^n. \tag{50}$$

Proof. The proof follows by combining arguments in the proofs of [5, Lemma 1.4.3] and [12, Proposition 9.1.4], where the result is established for piecewise linear functions with integer coefficients. For each permutation $\pi = \pi(1)\pi(2)\cdots\pi(\ell) \in \Pi$, let

$$P_\pi := \{x \in [0,1]^n : p_{\pi(1)}(x) \geq p_{\pi(2)}(x) \geq \cdots \geq p_{\pi(\ell)}(x)\}$$

be the polyhedron associated with π. We refer to [5, Lemma 1.4.2] for a proof of P_π, indeed, constituting a polyhedron. Define the dimension of a polyhedron as the maximum number of affinely independent points it contains minus 1. Let Σ be the collection of permutations whose associated polyhedra are n-dimensional, i.e.,

$$\Sigma := \{\pi \in \Pi : P_\pi \text{ is } n\text{-dimensional}\}.$$

Note that $\cup_{\pi \in \Sigma} P_\pi = [0,1]^n$ [5, Lemma 1.4.3]. For every $\pi \in \Sigma$, let $i_\pi \in \{1, \dots, \ell\}$ be such that $f_c(x) = p_{\pi(i_\pi)}(x)$, for $x \in P_\pi$, and let $g_\pi := \min_{i \in \{1,\dots,i_\pi\}} p_{\pi(i)}$. For every $\pi \in \Sigma$, $g_\pi(x) = f_c(x) = p_{\pi(i_\pi)}(x)$, for $x \in P_\pi$. Therefore,

$$f_c(x) \leq \max_{\pi \in \Sigma} g_\pi(x), \qquad \text{for } x \in [0,1]^n.$$

It hence suffices to prove that $f_c(x) \geq g_\pi(x)$, for $x \in [0,1]^n$, for all $\pi \in \Sigma$, to obtain

$$f_c(x) = \max_{\pi \in \Sigma} g_\pi(x), \qquad \text{for } x \in [0,1]^n,$$

which establishes the desired result (50). By continuity of f_c and the fact that the min and max of continuous functions are continuous, it suffices to establish $f_c \geq g_\pi$ on the set

$$\{x \in [0,1]^n : \exists \pi \in \Sigma \mid x \text{ is in the interior of } P_\pi\}.$$

Now fix an arbitrary $\pi \in \Sigma$. As already noted above, $g_\pi(x) = f_c(x)$, for $x \in P_\pi$. Take an arbitrary point $y \notin P_\pi$ that is in the interior of some P_η. There exists a $k \in \{1, \dots, \ell\}$ so that $f_c(x) = p_{\pi(k)}(x)$, for $x \in P_\eta$. We treat the cases $k \leq i_\pi$ and $k > i_\pi$ separately. First, if $k \leq i_\pi$, then

$$f_c(y) = p_{\pi(k)}(y) \geq \min_{i \in \{1,\dots,i_\pi\}} p_{\pi(i)}(y) = g_\pi(y).$$

Second, assume that $k > i_\pi$. Let x be a point in the interior of P_π. Then, $f_c(x) = p_{\pi(i_\pi)}(x) > p_{\pi(k)}(x)$. Denote by μ the linear function connecting the points $(x, f_c(x))$ and $(y, f_c(y))$.

Within $\{w = \lambda x + (1-\lambda)y : \mu(w) = f_{\mathrm{c}}(w), 0 < \lambda \leq 1\}$, which is the set of intersection points of f_{c} and μ over the interval $[x, y)$, let w_0 be the point closest to y (but distinct from y).

There exists a $k' \in \{1, \ldots, \ell\}$ such that $f_{\mathrm{c}}(w_0) = p_{\pi(k')}(w_0)$ and f_{c} coincides with $p_{\pi(k')}$ on a small interval $[w_0, v] \subset [w_0, y]$, with $v \neq w_0$. By continuity of f_{c}, the restriction of the graph of $p_{\pi(k')}$ to $[w_0, v]$ lies strictly below μ, i.e.,

$$p_{\pi(k')}(x) < \mu(x), \quad \text{for } x \in [w_0, v],\ x \neq w_0.$$

Moving from w_0 to y along $p_{\pi(k')}$, we get $p_{\pi(k')}(y) < f_{\mathrm{c}}(y)$. On the other hand, moving from w_0 to x, we obtain $p_{\pi(k')}(x) \geq f_{\mathrm{c}}(x) = p_{\pi(i_\pi)}(x)$, thus $k' \leq i_\pi$. Hence $g_\pi(z) \leq p_{\pi(k')}(z)$, for $z \in [0,1]^n$, and in particular $g_\pi(y) \leq p_{\pi(k')}(y) < f_{\mathrm{c}}(y)$.

As π and y above were arbitrary, we can conclude that $f_{\mathrm{c}}(x) \geq g_\pi(x)$, for all $x \in [0,1]^n$, for all $\pi \in \Sigma$. The proof is complete. ⊣

REFERENCES

[1] A. Adamatzky, ***Identification of Cellular Automata***, 1 ed., CRC Press, 1994.

[2] ———, *Automatic programming of cellular automata: Identification approach*, ***Kybernetes***, vol. 26 (1997), no. 2, pp. 126–135.

[3] ———, *Identification of cellular automata*, ***Encyclopedia of Complexity and Systems Science***, (2009), pp. 4739–4751.

[4] A. Adamatzky (editor), ***Game of Life Cellular Automata***, 1 ed., Springer, 2010.

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

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

[7] ———, *Neural networks and rational McNaughton functions*, ***Journal of Multiple-Valued Logic and Soft Computing***, vol. 11 (2005), no. 1-2, pp. 95–110.

[8] E. R. Berlekamp, J. H. Conway, and R. K. Guy, ***Winning Ways for Your Mathematical Plays***, Academic Press, New York, 1982.

[9] A. W. Burks, *Von Neumann's self-reproducing automata*, ***Technical Report 08226-11-T***, University of Michigan, 1969.

[10] C. C. Chang, *Algebraic analysis of many-valued logics*, ***Transactions of the American Mathematical Society***, vol. 88 (1958), no. 2, pp. 467–490.

[11] ———, *A new proof of the completeness of the Łukasiewicz axioms*, ***Transactions of the American Mathematical Society***, vol. 93 (1959), no. 1, pp. 74–80.

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

[13] M. Cook, *Universality in elementary cellular automata*, ***Complex Systems***, vol. 15 (2004), no. 1, pp. 1–40.

[14] S. Das and M. K. Chakraborty, *Formal logic of cellular automata*, ***Complex Systems***, vol. 30 (2021), no. 2, pp. 187–203.

[15] 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), pp. 77–85.

[16] D. Elbrächter, D. Perekrestenko, P. Grohs, and H. Bölcskei, *Deep neural network approximation theory*, ***IEEE Transactions on Information Theory***, vol. 67 (2021), no. 5, pp. 2581–2623.

[17] J. L. Elman, *Finding structure in time*, ***Cognitive Science***, vol. 14 (1990), no. 2, pp. 179–211.

[18] M. Gardner, *The fantastic combinations of John Conway's new solitaire game "life"*, ***Scientific American***, vol. 223 (1970), pp. 120–123.

[19] B. Gerla, *Rational Łukasiewicz logic and DMV-algebras*, ***Neural Network Worlds***, vol. 25 (2001), pp. 1–13.

[20] W. Gilpin, *Cellular automata as convolutional neural networks*, ***Physical Review E***, vol. 100 (2019), p. 032402.

[21] A. Graves and J. Schmidhuber, *Offline handwriting recognition with multidimensional recurrent neural networks*, ***Proceedings of the 21st International Conference on Neural Information Processing Systems***, 2008.

[22] A. Hatcher, ***Algebraic Topology***, Cambridge University Press, 2002.

[23] I. Hubara et al., *Quantized neural networks: Training neural networks with low precision weights and activations*, ***Journal of Machine Learning Research***, vol. 18 (2017), no. 1, p. 6869–6898.

[24] C. Hutter, R. Gül, and H. Bölcskei, *Metric entropy limits on recurrent neural network learning of linear dynamical systems*, ***Applied and Computational Harmonic Analysis***, vol. 59 (2022), pp. 198–223.

[25] T. Ishida, S. Inokuchi, and Y. Kawahara, *Cellular automata and formulae on monoids*, ***Proceedings of the 11th International Conference on Cellular Automata for Research and Industry***, 2014, pp. 55–64.

[26] J. Jumper et al., *Highly accurate protein structure prediction with AlphaFold*, ***Nature***, vol. 596 (2021), no. 7873, pp. 583–589.

[27] J. Kari, *Theory of cellular automata: A survey*, ***Theoretical Computer Science***, vol. 334 (2005), no. 1-3, pp. 3–33.

[28] S. B. Kotsiantis, *Supervised machine learning: A review of classification techniques*, ***Informatica***, vol. 31 (2007), pp. 249–268.

[29] G. Leifert et al., *Cells in multidimensional recurrent neural networks*, ***Journal of Machine Learning Research***, vol. 17 (2016), no. 1, p. 3313–3349.

[30] M. Liang and X. Hu, *Recurrent convolutional neural network for object recognition*, ***2015 IEEE Conference on Computer Vision and Pattern Recognition***, 2015, pp. 3367–3375.

[31] W. S. McCulloch and W. Pitts, *A logical calculus of the ideas immanent in nervous activity*, ***Bulletin of Mathematical Biophysics***, vol. 5 (1943), pp. 115–133.

[32] R. McNaughton, *A theorem about infinite-valued sentential logic*, ***The Journal of Symbolic Logic***, vol. 16 (1951), no. 1, pp. 1–13.

[33] D. Mundici, *A constructive proof of McNaughton's theorem in infinite-valued logic*, ***The Journal of Symbolic Logic***, vol. 59 (1994), no. 2, pp. 596–602.

[34] A. Radford, L. Metz, and S. Chintala, *Unsupervised representation learning with deep convolutional generative adversarial networks*, ***Proceedings of the International Conference on Learning Representations***, 2016.

[35] P. Rendell, *Turing universality of the Game of Life*, ***Collision-Based Computing*** (A. Adamatzky, editor), Springer, 2002, pp. 513–539.

[36] F. C. Richards, T. P. Meyer, and N. H. Packard, *Extracting cellular automaton rules directly from experimental data*, ***Physica D: Nonlinear Phenomena***,

vol. 45 (1990), no. 1, pp. 189–202.

[37] A. Rose and J. B. Rosser, *Fragments of many-valued statement calculi*, ***Transactions of the American Mathematical Society***, vol. 87 (1958), no. 1, pp. 1–53.

[38] K. H. Rosen, ***Discrete Mathematics and Its Applications***, 7 ed., New York: McGraw-Hill, 2012.

[39] R. Rovatti, M. Borgatti, and R. Guerrieri, *A geometric approach to maximum-speed n-dimensional continuous linear interpolation in rectangular grids*, ***IEEE Transactions on Computers***, vol. 47 (1998), no. 8, pp. 894–899.

[40] D. Silver et al., *Mastering the game of Go with deep neural networks and tree search*, ***Nature***, vol. 529 (2016), pp. 484–489.

[41] L. Smith, ***Linear Algebra***, Springer, 1998.

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

[43] S. Ulam, *Random processes and transformations*, ***Proceedings of the International Congress on Mathematics***, vol. 2, 1950, pp. 264–275.

[44] P. P. Vaidyanathan, ***Multirate systems and filter banks***, Pearson Education India, 2006.

[45] J. Von Neumann, *The general and logical theory of automata*, ***Systems Research for Behavioral Science***, Routledge, 1968, pp. 97–107.

[46] B. H. Voorhees, ***Computational analysis of one-dimensional cellular automata***, vol. 15, World Scientific, 1996.

[47] R. T. Wainwright, *Life is universal!*, ***Proceedings of the 7th Conference on Winter Simulation-Volume 2***, 1974, pp. 449–459.

[48] A. Weiser and S. E. Zarantonello, *A note on piecewise linear and multilinear table interpolation in many dimensions*, ***Mathematics of Computation***, vol. 50 (1988), no. 181, pp. 189–196.

[49] S. Wolfram, *Statistical mechanics of cellular automata*, ***Reviews of Modern Physics***, vol. 55 (1983), pp. 601–644.

[50] ———, ***A New Kind of Science***, Wolfram Media, 2002.

[51] S. Wu, G. Li, F. Chen, and L. Shi, *Training and inference with integers in deep neural networks*, ***Proceedings of the 6th International Conference on Learning Representations***, 2018.

[52] N. Wulff and J. A. Hertz, *Learning cellular automaton dynamics with neural networks*, ***Proceedings of the 5th International Conference on Neural Information Processing Systems***, vol. 5, 1992, pp. 631–638.

[53] Y. Zhao and S. Billings, *Neighborhood detection using mutual information for the identification of cellular automata*, ***IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)***, vol. 36 (2006), no. 2, pp. 473–479.

CHAIR FOR MATHEMATICAL INFORMATION SCIENCE
ETH ZURICH
SWITZERLAND
E-mail: yanizhang@mins.ee.ethz.ch

CHAIR FOR MATHEMATICAL INFORMATION SCIENCE
ETH ZURICH
SWITZERLAND
E-mail: hboelcskei@ethz.ch