Cellular automata, many-valued logic, and deep neural networks

Yani Zhang

Chair for Mathematical Information Science

ETH zürich

May 2024

joint work with Helmut Bölcskei

Q1: What is the logical rule generating this sequence?

… 1101111001101001011111001111111 …

Q2: Can neural networks learn this logical rule from the data?

Q3: How can the rule then be read out from the trained network?

Cellular automata (CA)

Cellular space: $\mathbb{Z}^d$

State set: $K = \left\{0, \frac{1}{k-1}, \frac{2}{k-1}, \ldots, 1\right\}$

$$|K| = k$$

Neighborhood set: $\mathscr{E}, |\mathscr{E}| = n$

Transition function: $f : K^n \to K$

E.g. 1

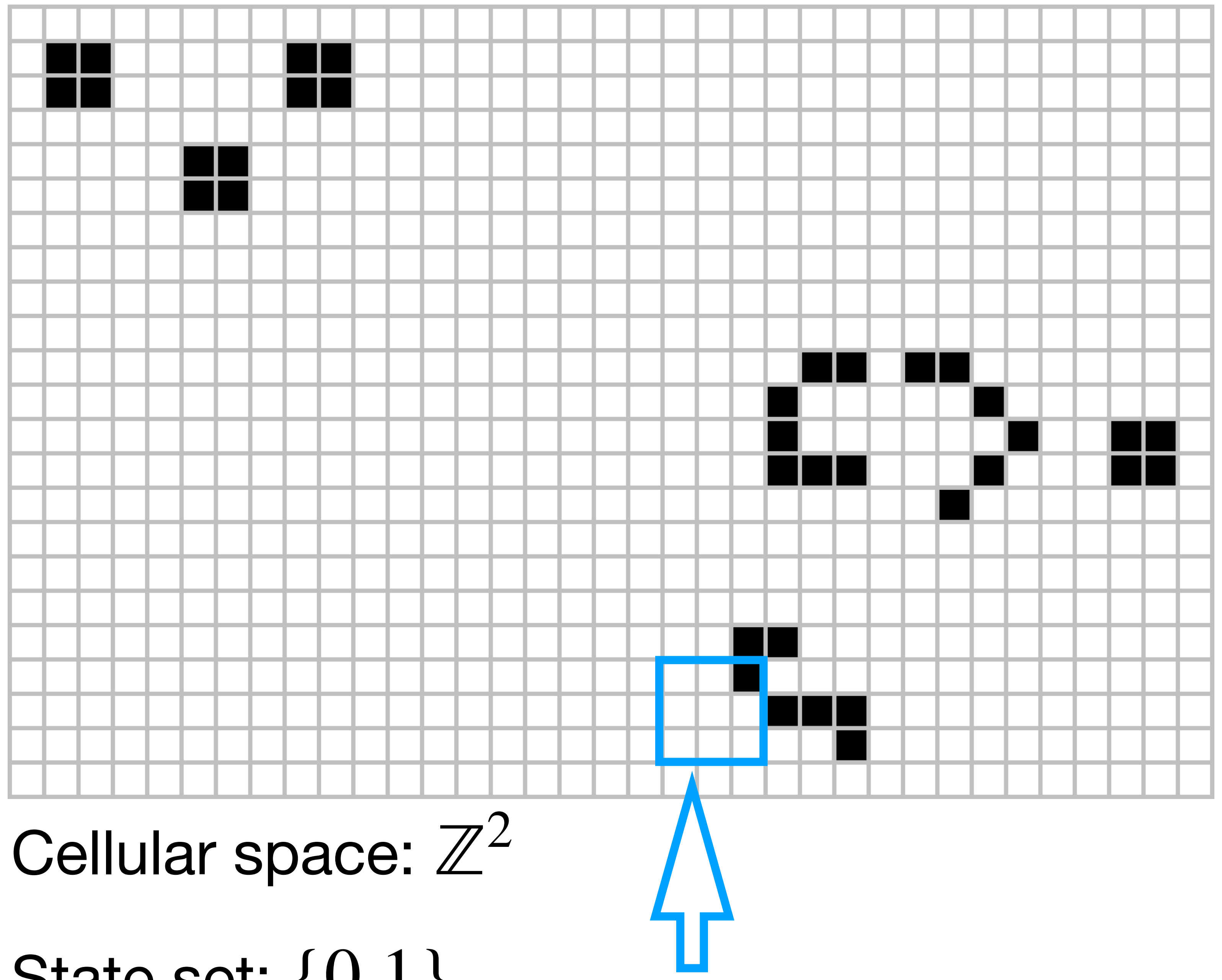
… 11011110011010010111110011111111 …

Cellular space: $\mathbb{Z}$

State set: $\{0,1\}$

Neighborhood: $|\mathscr{E}| = 3$

E.g. 2

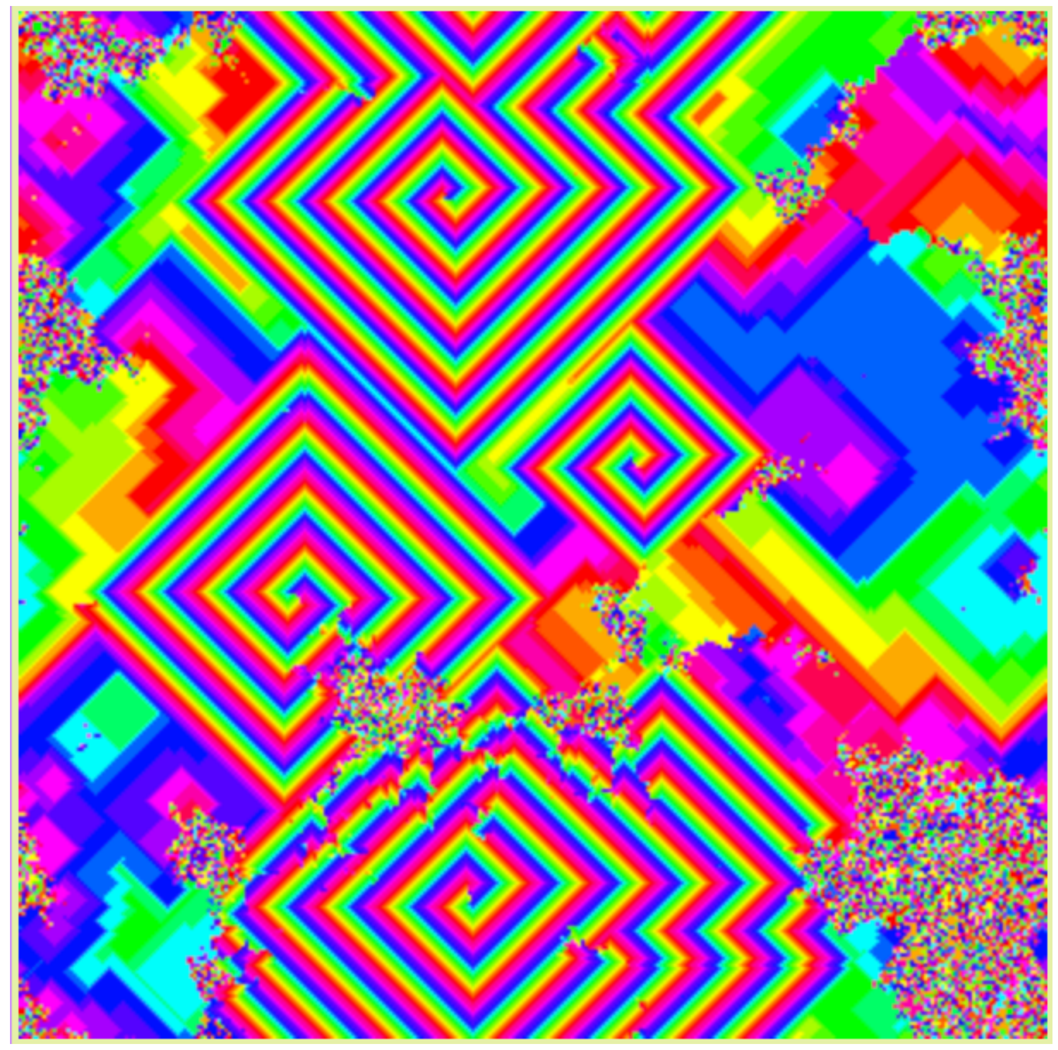


Cellular space: $\mathbb{Z}^2$

State set: $\{0,1\}$

Neighborhood: $|\mathscr{E}| = 9$

E.g. 3



Cellular space: $\mathbb{Z}^2$

State set: $K, k = 16$

Neighborhood: $|\mathscr{E}| = 9$

Cellular automata (CA)

Cellular space: $\mathbb{Z}^d$

State set: $K = \left\{0, \frac{1}{k-1}, \frac{2}{k-1}, \ldots, 1\right\}$

$$|K| = k$$

Neighborhood set: $\mathscr{E}, |\mathscr{E}| = n$

Transition function: $f : K^n \to K$

E.g. 1

… 1101111001101001011111001111111 …

Cellular space: $\mathbb{Z}$

State set: $\{0,1\}$

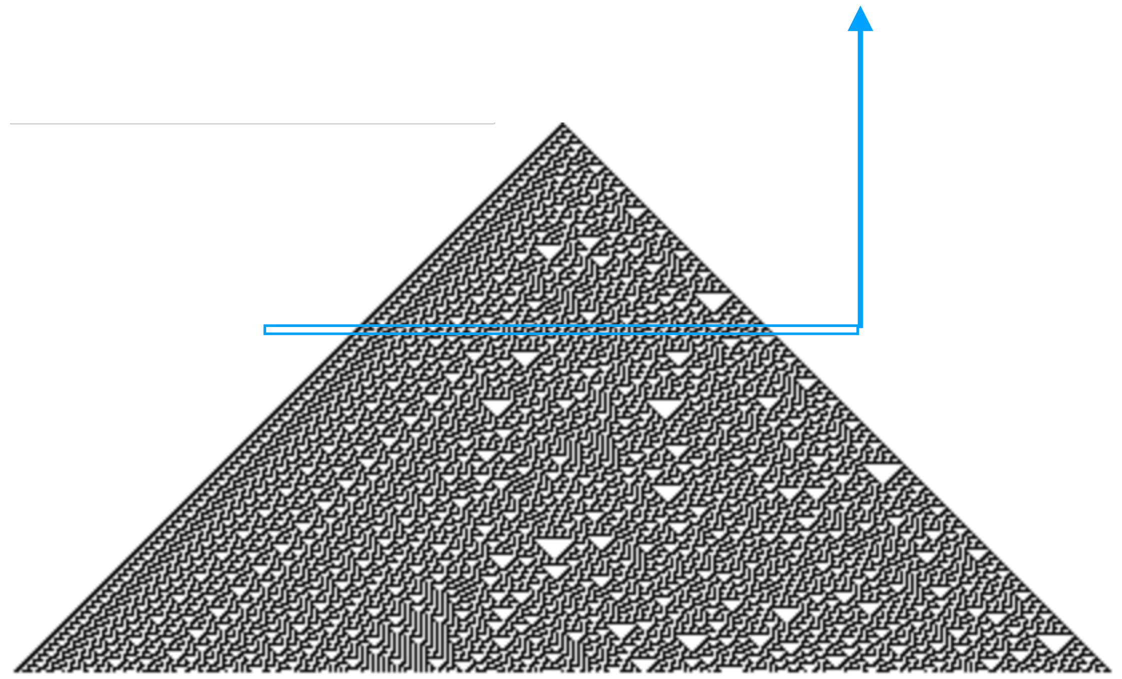
Neighborhood: $|\mathscr{E}| = 3$

$x_{-1}x_0x_1$	111	110	101	100	011	010	001	000
$f(x_{-1}, x_0, x_1)$	0	0	0	1	1	1	1	0

Cellular automata (CA)

… 1101111001101001011111001111111 …

E.g. 1

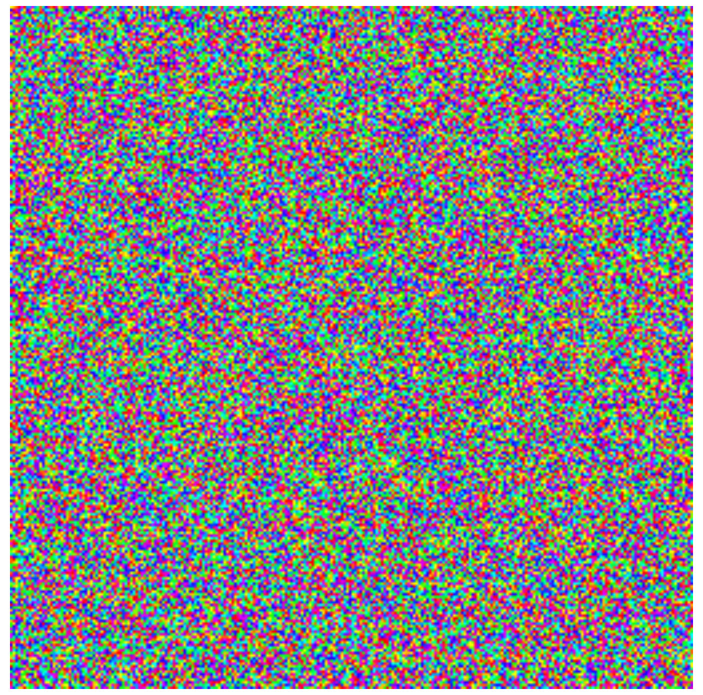


Elementary CA of rule 30

E.g. 2

Game of Life

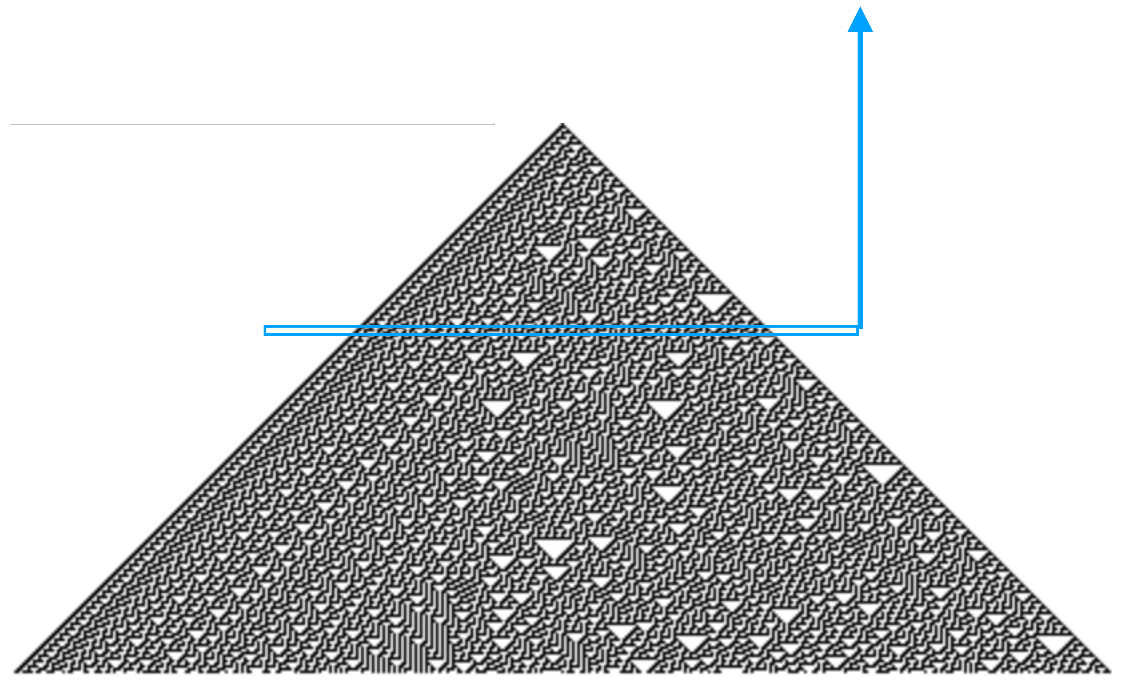
E.g. 3



Colored
cyclic CA

Q1: What is the logical rule generating this sequence?

… 11011110011010010111110011111111 …



Elementary CA of rule 30

$x_{-1}x_0x_1$	111	110	101	100	011	010	001	000
$f(x_{-1},x_0,x_1)$	0	0	0	1	1	1	1	0

$$f = (x_{-1} \odot \neg x_0 \odot \neg x_1) \oplus (\neg x_{-1} \odot x_1) \oplus (\neg x_{-1} \odot x_0)$$

Q1: What is the logical rule generating this sequence?

Theorem:
Every CA is a logical machine, namely in Lukasiewicz propositional logic.

Q2: Can neural networks learn the logical rule from the data?

Theorem:
Neural networks can learn the transition rule from CA evolution data.

Q3: How can the rule then be read out from the trained network?

Proposed an extraction procedure.

Definition [Chang, 1958]:

A **many-valued (MV) algebra** is a structure $\mathscr{A} = \langle A, \oplus, \neg, 0\rangle$ consisting of

- a nonempty set A
- a constant $0 \in A$
- a binary operation $\oplus$
- a unary operation $\neg$

satisfying the following axioms:

$$x \oplus (y \oplus z) = (x \oplus y) \oplus z$$
$$x \oplus y = y \oplus x$$
$$x \oplus 0 = x$$
$$\neg\neg x = x$$
$$x \oplus \neg 0 = \neg 0$$
$$\neg(\neg x \oplus y) \oplus y = \neg(\neg y \oplus x) \oplus x$$

add $x \oplus x = x$: Boolean algebra

Definition:

Let $n \in \mathbb{N}$ and $S_n = \{(,),0,\neg,\oplus,x_1,\ldots,x_n\}$. An *MV term* 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 $x_i, i = 1,\ldots,n$, are MV terms.

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

Examples: $x_1, \neg x_2, x_1 \oplus \neg x_2, \neg\neg x_3$

Definition:

Let $n \in \mathbb{N}$ and $S_n = \{(,),0,\neg,\oplus,x_1,\ldots,x_n\}$. An *MV term* 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, $i = 1,\ldots,n$, are MV terms.

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

Definition:

Let $\tau(x_1,\ldots,x_n)$ be an MV term and $\mathscr{A} = \langle A, \oplus, \neg, 0\rangle$ an MV algebra. The *term function*

$$\tau^{\mathscr{A}} : A^n \to A$$

is obtained by interpreting the symbols $\oplus$ and $\neg$ according to how they are specified in $\mathscr{A}$.

Definition:

Let $n \in \mathbb{N}$ and $S_n = \{(,),0,\neg, \oplus ,x_1, \ldots, x_n\}$. An *MV term* 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 $x_i, i = 1,\ldots, n$, are MV terms.

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

Definition:

Let $\tau(x_1, \ldots, x_n)$ be an MV term and $\mathscr{A} = \langle A, \oplus , \neg,0\rangle$ an MV algebra. The *term function*

$$\tau^{\mathscr{A}} : A^n \to A$$

is obtained by interpreting the symbols $\oplus$ and $\neg$ according to how they are specified in $\mathscr{A}$.

Examples: the Boolean algebra $\mathscr{B} = \{\{0,1\}, \oplus , \neg,0\}$ $\tau^{\mathscr{B}} : \{0,1\}^n \to \{0,1\}$

$x_{-1}x_0x_1$	111	110	101	100	011	010	001	000
$f(x_{-1}, x_0, x_1)$	0	0	0	1	1	1	1	0

Definition:

Consider the unit interval $[0,1]$, 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

$$\mathscr{I} = \langle [0,1], \oplus, \neg, 0, \rangle$$

is an MV algebra. We further define the operation

$$x \odot y := \neg(\neg x \oplus \neg y) = \max\{0, x + y - 1\}.$$

Completeness theorem [Chang, 1958, 1959]:

An equation holds in every MV algebra if and only if it holds in $\mathscr{I}$.

Every **binary** truth table has an associated Boolean formula e.g. [Rosen, 2012]

General functions $f : [0,1]^n \to [0,1]$?

How do term functions in MV logic look like?

How do term functions in MV logic look like?

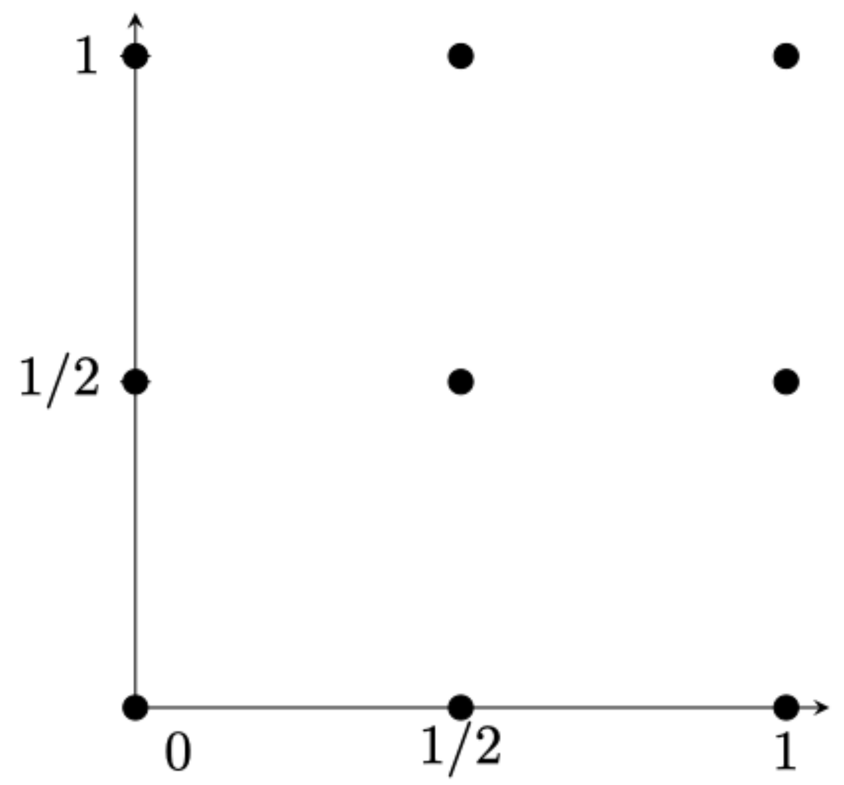
Theorem [McNaughton, 1951]:
Consider the standard MV algebra $\mathcal{I} = \langle [0,1], \oplus, \neg, 0\rangle$. Let $n \in \mathbb{N}$. For a function $f_c : [0,1]^n \to [0,1]$ to have an associated MV term τ such that $\tau^{\mathcal{I}} = f_c$ on $[0,1]^n$, it is necessary and sufficient that

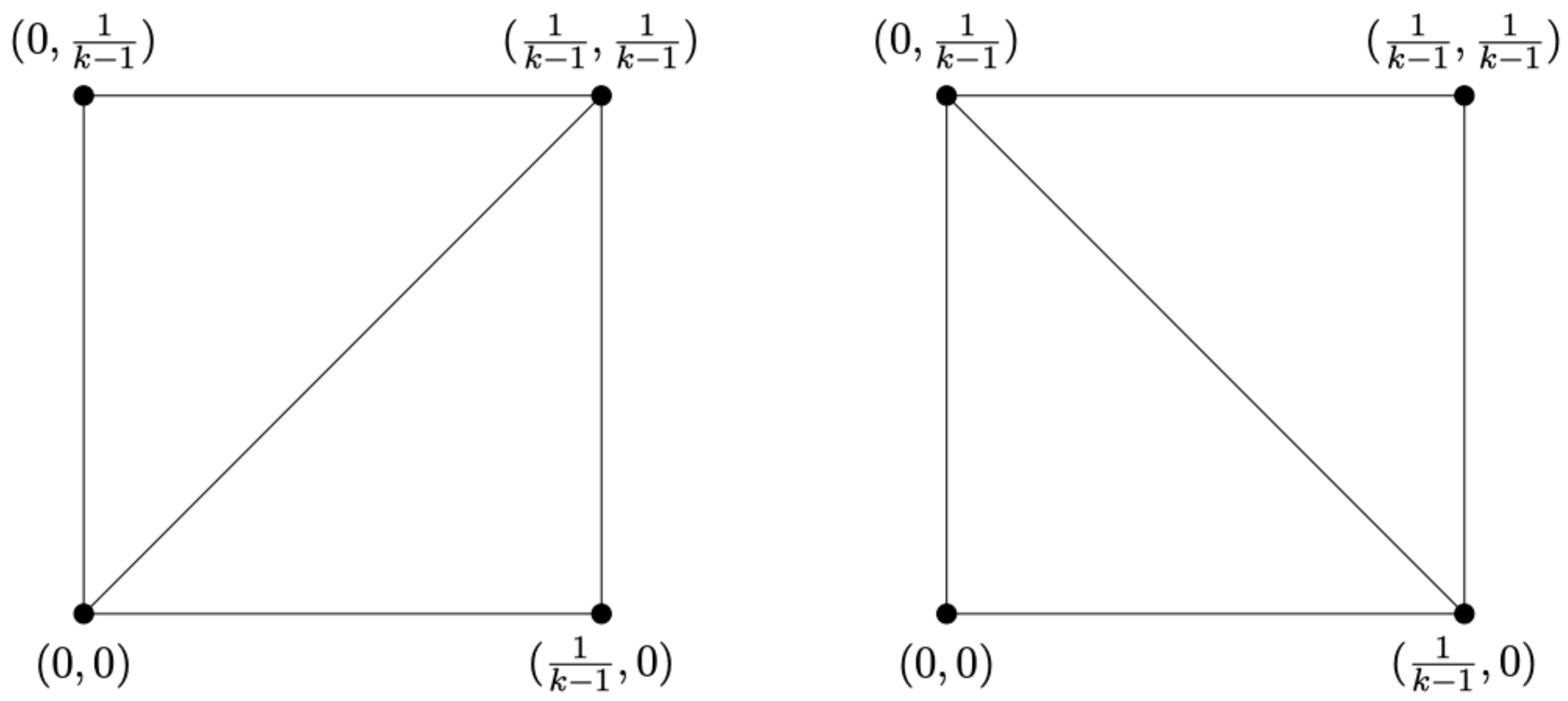
1. f_c is continuous with respect to the natural topology on $[0,1]^n$
2. there exist linear functions $p_1, \ldots, p_\ell$ with integer coefficients, i.e.,

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

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 $f_c(x) = p_j(x)$.

continuous piecewise linear functions with integer coefficients

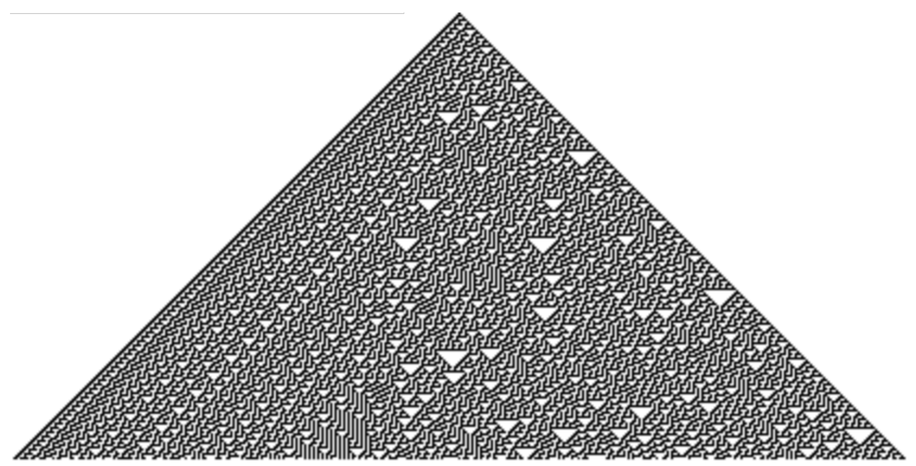


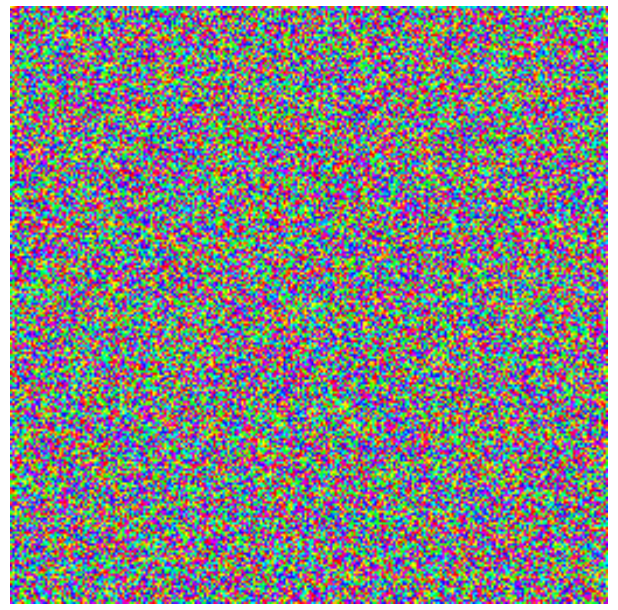


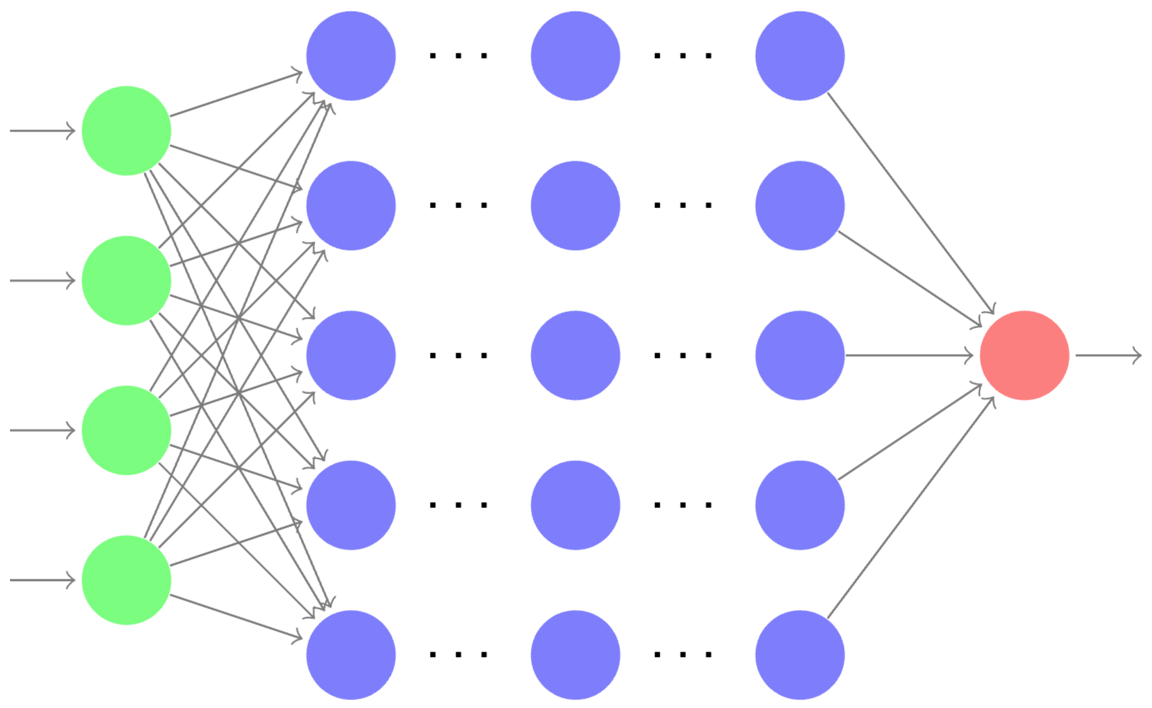
Simplex interpolation

Theorem:

Every CA is a logical machine, namely in Lukasiewicz propositional logic.



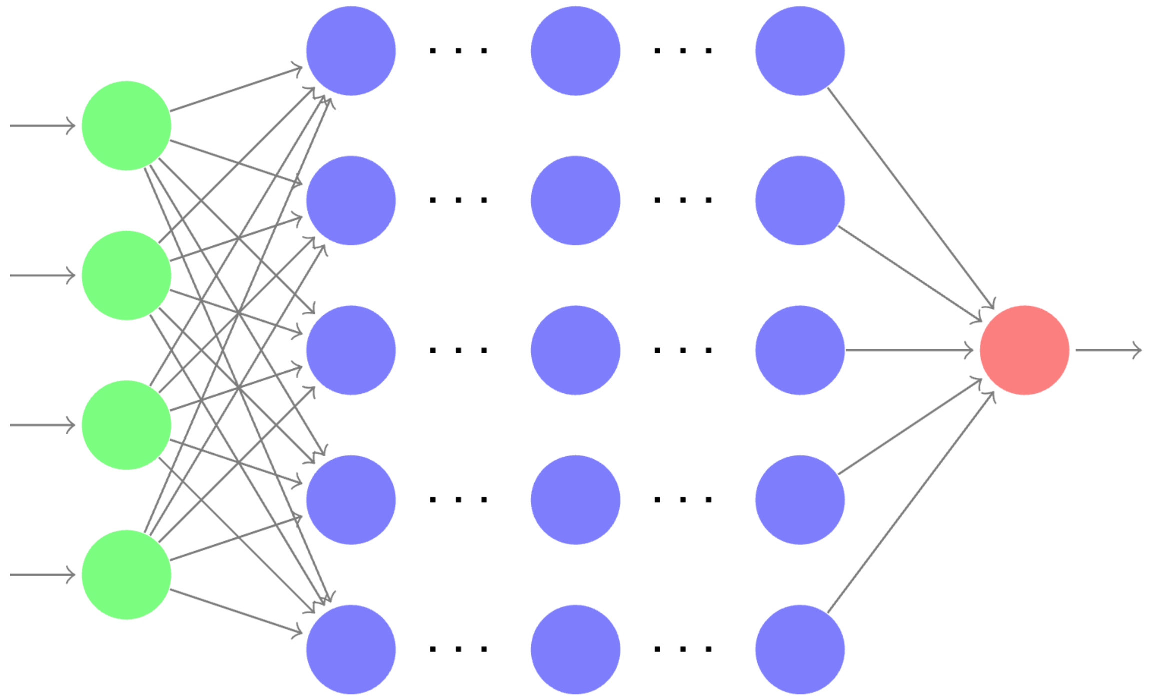




Theorem:

Neural networks can learn the transition rule from CA evolution data.

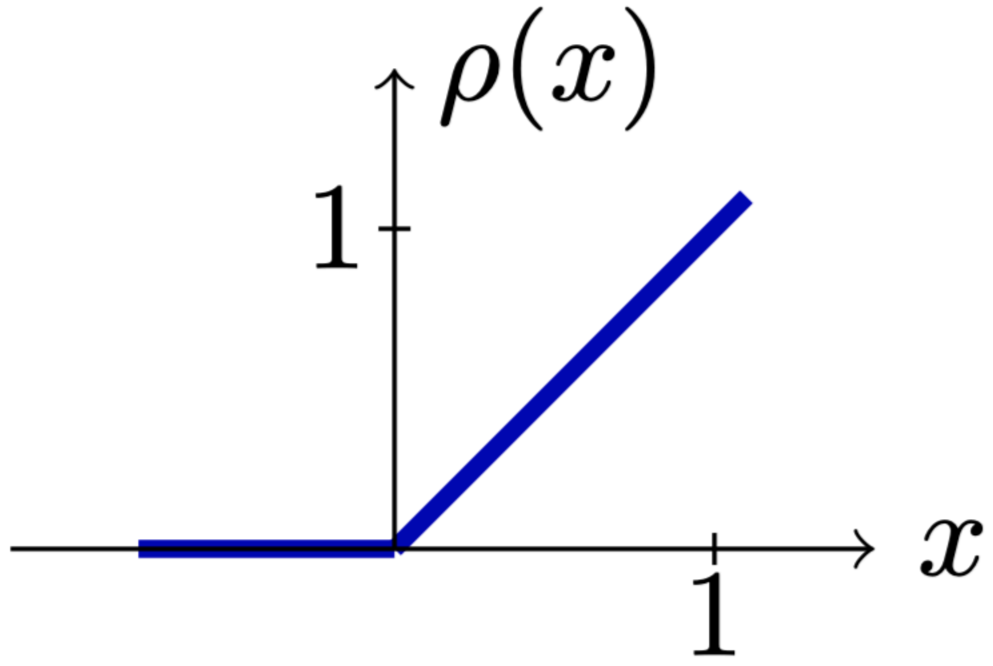
Deep ReLU networks can realize MV term functions

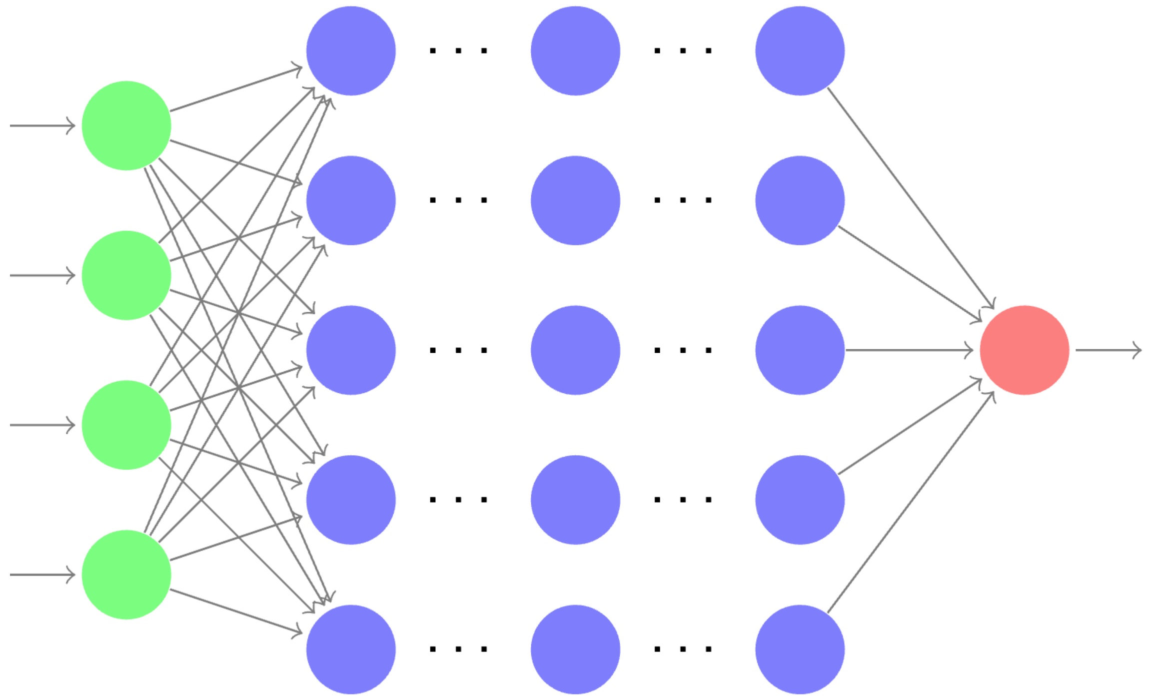


$$\Phi = W_L \circ \rho \circ W_{L-1} \circ \ldots \circ W_2 \circ \rho \circ W_1$$

affine maps: $W_\ell = A_\ell x + b_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}, \ell \in \{1,2,\ldots,L\}$

nonlinearity: $\rho = \max\{0,x\}$





Building blocks:

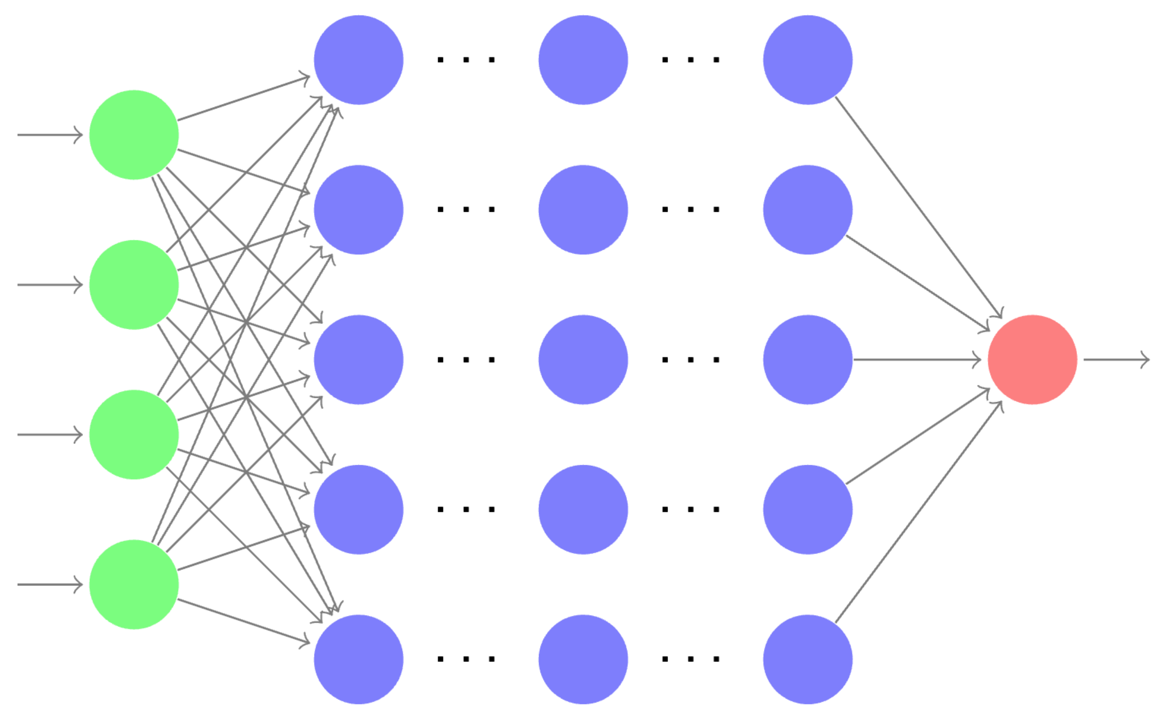
$$\Phi^{\neg} = 1 - x$$

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

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

Compositions of ReLU nets are ReLU nets

$$\underbrace{W_{L_2}^{(2)} \circ \rho \circ \cdots \circ \rho \circ W_1^{(2)}}_{\Phi^{(2)}} \circ \underbrace{W_{L_1}^{(1)} \circ \rho \circ \cdots \circ \rho \circ W_1^{(1)}}_{\Phi^{(1)}}$$



Building blocks:

$$\Phi^{\neg} = 1 - x$$

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

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

Example $\tau = (x \oplus x) \odot \neg y$

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

$$\neg y = -\rho(y) + \rho(-y) + 1$$

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

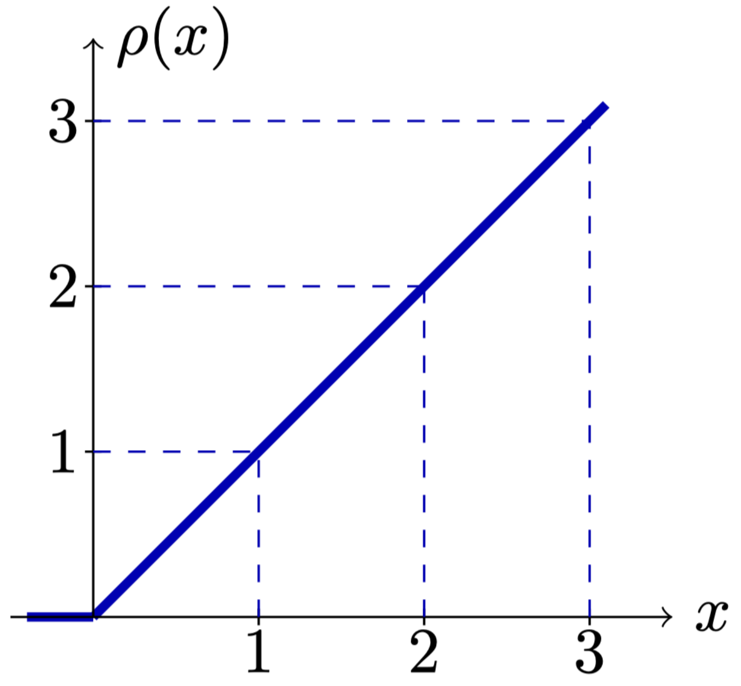
Extract MV terms from trained networks

Convert the learned truth functions to **algebraic formulae**, thereby extracting the ``logic'' behind data

But extraction isn't so easy …

We want to proceed **layer-by-layer, neuron-by-neuron**

to exploit the compositional structure of ReLU networks



MV term functions are $f : [0,1]^n \to [0,1]$, but

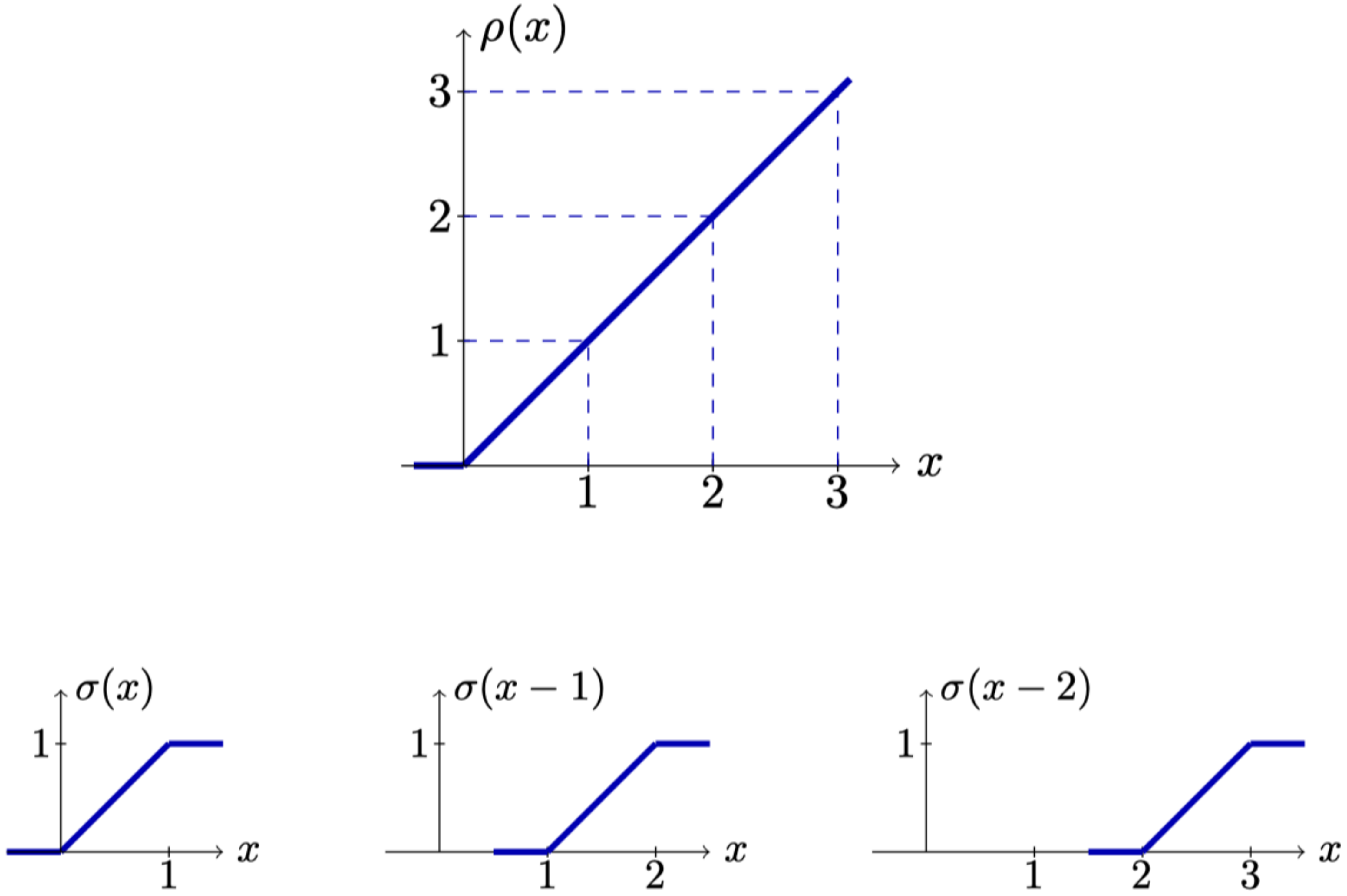
$$\rho \circ W : [0,1]^n \to \mathbb{R}^+,$$

e.g. $\rho(3x) : [0,1] \to [0,3]$

do not have an MV term!

Step 1: From ρ-neurons to σ-neurons

Step 1: From ρ-neurons to σ-neurons

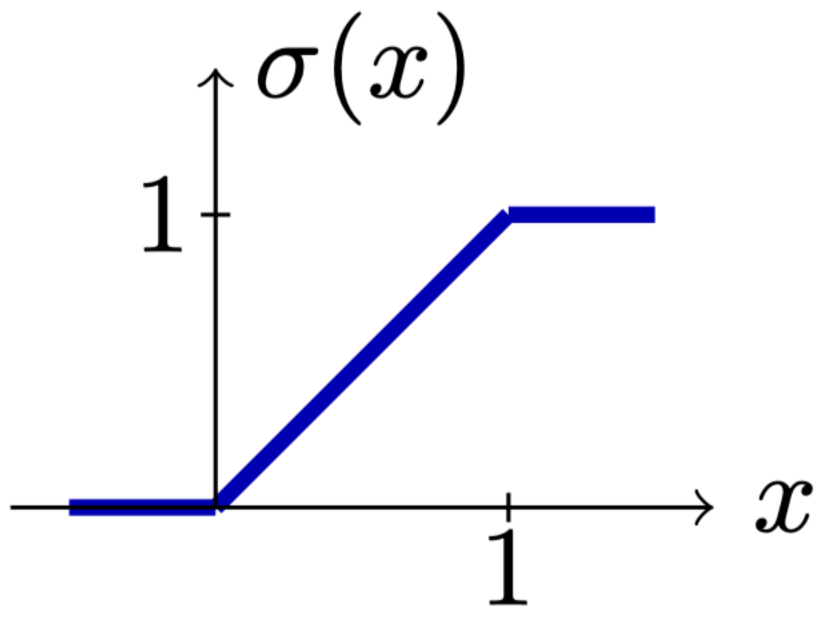


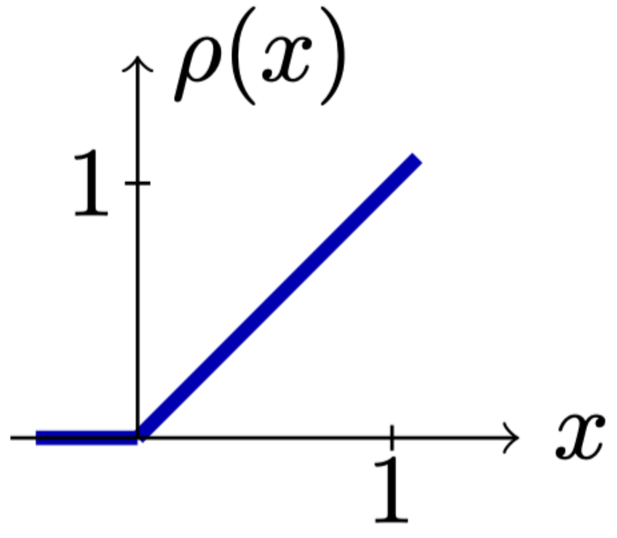
$\rho(x) = \sigma(x) + \sigma(x-1) + \sigma(x-2)$, for $x \in [0,3]$

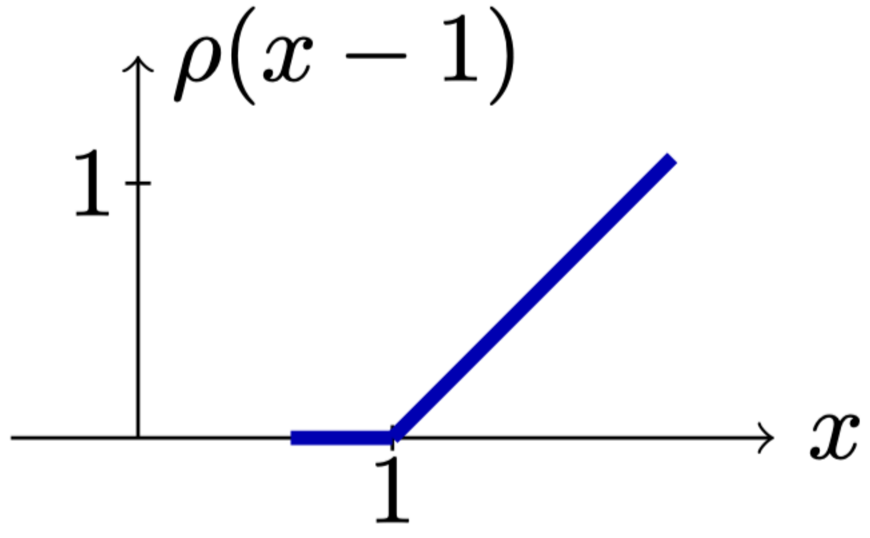
Step 1: From ρ-neurons to σ-neurons

Can always go back

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







Step 2: Extract MV terms from individual σ-neurons

Lemma [Rose and Rosser, 1958; Mundici, 1994]

E.g., $\sigma(x_1 - x_2 + x_3 - 1)$

$$\sigma(x_1 - x_2 + x_3 - 1) = (\sigma(-x_2 + x_3 - 1) \oplus x_1) \odot \sigma(-x_2 + x_3)$$

$$\sigma(-x_2 + x_3 - 1) = (\sigma(-x_2 - 1) \oplus x_3) \odot \sigma(-x_2)$$

$$\sigma(-x_2 + x_3) = (\sigma(-x_2) \oplus x_3) \odot \sigma(-x_2 + 1)$$

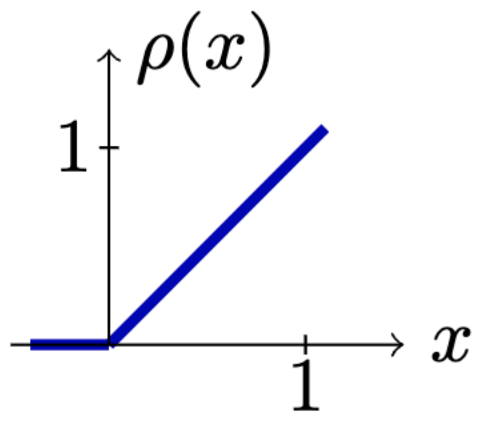
$$\sigma(-x_2 + 1) = 1 - \sigma(x_2) = \neg x_2$$

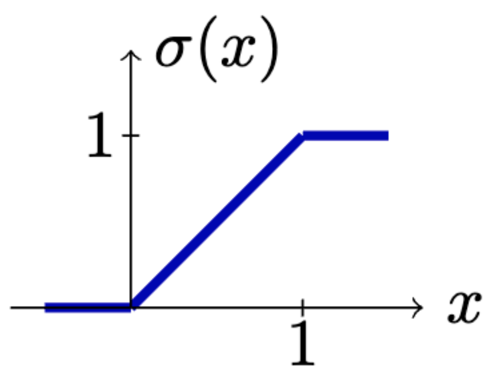
Overall:

$\sigma(x_1 - x_2 + x_3 - 1) : x_1 \odot (x_3 \odot \neg x_2)$

The extraction procedure

Step 1: Convert into equivalent σ-network

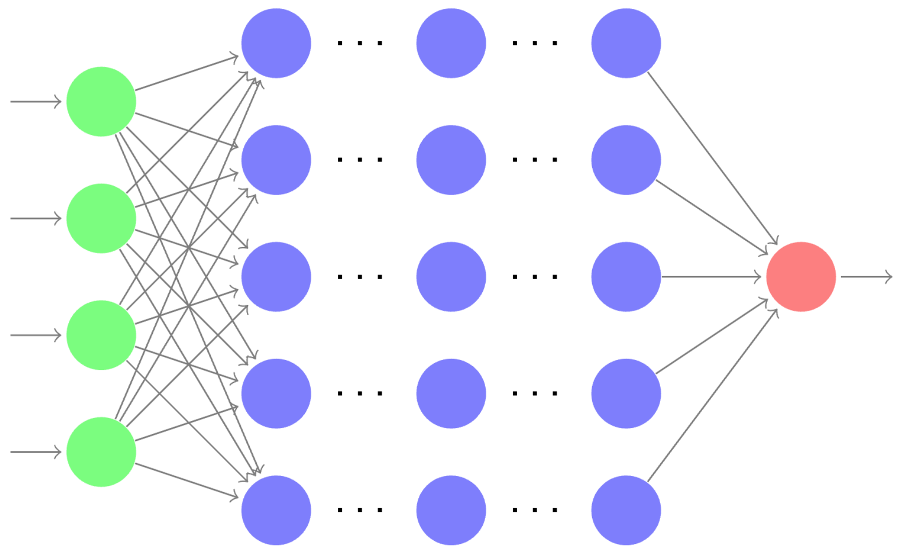




Step 2: Extract MV terms from individual neurons

E.g., $\sigma(2x - y + 1) : x \oplus x \oplus \neg y$

Step 3: Compose



Q1: What is the logical rule generating this sequence?

Theorem:

Every CA is a logical machine, namely in Lukasiewicz propositional logic.

Q2: Can neural networks learn the logical rule from the data?

Theorem:

Neural networks can learn the transition rule from CA evolution data.

Q3: How can the rule then be read out from the trained network?

Proposed an extraction procedure.

Reference

Y. Zhang and H. Bölcskei, “Cellular automata, many-valued logic, and deep neural networks”, arXiv:2404.05259.
Y. Zhang and H. Bölcskei, “Extracting formulae in many-valued logic from deep neural networks”, arxiv:2401.12113.