A Neural Network Performing Boolean Logic Operations

Jacek Mańdziuk and Bohdan Macukow

Faculty of Mathematics and Information Science,
Warsaw University of Technology,
Plac Politechniki 1, 00-661 Warsaw, POLAND

Abstract

A neural network, composed of neurons of two types, able to perform Boolean operations is presented. Based on recursive definition of "basic Boolean operations" the proposed model, for any fixed number of input variables, can either realize all of them paralelly, or accomplish any chosen one of them alone.

Moreover, possibilities of combining a few such networks into more complex structures in order to perform superpositions of basic operations are disscussed.

The general concept of neural implementation of First Order logic (FO) based on the presented network is also introduced.

Key Words : neural networks, logic operations, First Order logic

Published in **Optical Memory and Neural Networks 2(1), 17-35, (1993)**

e-mail: mandziuk@mini.pw.edu.pl
http://www.mini.pw.edu.pl/˜ mandziuk

1 Introduction

In the last years much effort has been devoted to neural networks and great progress in this field has been achieved. Many theoretical results as well as industrial applications concerning neural nets have been presented. However, most publications focused on pattern recognition, e.g. [1-2], content addressable memories, e.g. [3-5], and optimisation problems, e.g. [6-7], some papers also dealt with applications of optical neural networks in realizing Boolean logic operations of a few variables [8-12]. This field of application of neural nets seems very promissing. Massive parallelism, inherent in neural nets allows performing logic operations with high speed. Moreover, small neural nets performing basic logic operations of a few variables [12], used as parts of bigger structures are able to realize more complex tasks. This way may finally lead to building a general-purpose neurocomputer based on logic operations.

In this paper a neural network performing all possible basic Boolean operations of n variables, for given n, is presented. Definition of "basic Boolean operation" is introduced in Section 2. Moreover, in this section the idea of the network based on recursive definition of basic Boolean operations with respect to the number of variables is introduced. In Section 3 a closer look at the network and its formal definition are given. Matrix description of the proposed network is presented in Section 4. Characteristic features of the network such as: its "open" structure, using external bias, and others are disscussed in Section 5.

Sections 6 and 7 are devoted to some extensions of the presented model, such as the way of constructing a network performing any fixed, arbitrarily chosen basic Boolean operation and combining simple networks into more complex structures. Additionally, remarks about neural implementation of First Order logic [13] and its extensions (majority and modulo quantifiers [14]) are placed in Section 7.

Finally, concluding remarks and suggestions of possible further development of the model are covered by Section 8.

2 Preliminaries

Let us define the term "basic Boolean (logic) operation" first.
Definition.
The k-th basic Boolean operation of n variables $x_1, \ldots, x_n$ denoted

$$B_n^k(x_1, \ldots, x_n), \qquad k = 0, \ldots, 2^{2^n} - 1, n \in \mathcal{N}$$

is defined by Tab. 1, where

$$k = f_{2^n-1} \cdot 2^{2^n-1} + f_{2^n-2} \cdot 2^{2^n-2} + \ldots + f_1 \cdot 2^1 + f_0 \cdot 2^0$$

$f_i \in \{0, 1\}, \; i = 0, \ldots, 2^n - 1 \qquad$ that is $\qquad k = (f_{2^n-1}, f_{2^n-2}, \ldots, f_1, f_0)_2.$

x_1	x_2	$\dots$	x_n	$B_n^k(x_1,\dots,x_n)$
1	1	$\dots$	1	f_{2^n-1}
0	1	$\dots$	1	f_{2^n-2}
1	0	$\dots$	1	f_{2^n-3}
0	0	$\dots$	1	f_{2^n-4}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
1	1	$\dots$	0	f_3
0	1	$\dots$	0	f_2
1	0	$\dots$	0	f_1
0	0	$\dots$	0	f_0

Table 1. The "zero-one table" presentation of definition of the k-th basic Boolean operation of n variables $x_1,\dots,x_n$, where $(f_{2^n-1}, f_{2^n-2},\dots,f_1,f_0)_2$ is a binary representation of k.

Certainly it is only a matter of convention in which way we attribute zeros and ones in the table. In our definition variable x_1 changes most often and has 1 in the first row, 0 in the second row, and so on. Variable x_2 has ones in the first two rows, then zeros in the next two rows, then two ones again, and so on. At last x_n changes most slowly and has ones in the upper half of the table and zeros in the lower half.

In definition above we assume that basic logic operation is defined "straight" by $B_n^k()$, not being a superposition of other logic operations. Certainly, any superposition of Boolean operations finally yields to some basic operation, but superpositions of a few operations require a little different treatment (see Section 7).

Henceforth, instead of "basic logic operation" we shall simply write "operation", unless it may lead to misunderstandings. Other, (not basic) logic operations will be called "logic functions", e.g. $B_2^8(\ (B_1^1(x_1),\ x_2)\ )$ or $B_2^4(\ (B_1^3(x_1),\ B_2^7(x_1,x_2)\ )$, etc.

For $n=1$ there are four operations (see Tab. 2):

$$B_1^0(x_1),\ B_1^1(x_1),\ B_1^2(x_1),\text{ and } B_1^3(x_1)$$

equivalent to $FALSE$ (*output always* 0), $\overline{x}_1$ (*not* x_1), x_1 and $TRUE$ (*output always* 1), resp. and for $n=2$ there are sixteen operations (see Tab. 3), etc.

x_1	$B_1^0(x_1)$	$B_1^1(x_1)$	$B_1^2(x_1)$	$B_1^3(x_1)$
1	0	0	1	1
0	0	1	0	1

Table 2. Four possible operations $B_1^0(x_1)$, $B_1^1(x_1)$, $B_1^2(x_1)$, and $B_1^3(x_1)$ for one variable x_1. Those operations are equivalent to $FALSE$, $\overline{x}_1$, x_1 and $TRUE$, respectively.

x_1	x_2	B_2^0	B_2^1	B_2^2	B_2^3	B_2^4	B_2^5	B_2^6	B_2^7	B_2^8	B_2^9	B_2^{10}	B_2^{11}	B_2^{12}	B_2^{13}	B_2^{14}	B_2^{15}
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Table 3. Sixteen possible operations $B_2^0(x_1, x_2), \ldots, B_2^{15}(x_1, x_2)$ for two input variables x_1 and x_2.

For example, operations AND, OR, NAND:

$$x_1 \wedge x_2, \qquad x_1 \vee x_2, \qquad \overline{x_1 \wedge x_2}$$

are in this notation equivalent to

$$B_2^8(x_1, x_2), \qquad B_2^{14}(x_1, x_2) \qquad \text{and} \qquad B_2^7(x_1, x_2),$$

respectively.

The network presented in the paper is based on the following preposition:
Preposition.
$\forall p, n \in \mathcal{N} \qquad \forall x_1, \ldots, x_n$
if $p = (f_{2^n-1}, f_{2^n-2}, \ldots, f_1, f_0)_2, \qquad f_i \in \{0,1\}, \;\; i = 0, \ldots, 2^n - 1$
then $\exists!\; r, s \in \mathcal{N}$ such that

$$B_n^p(x_1, \ldots, x_n) = \Big(\; B_{n-1}^r(x_1, \ldots, x_{n-1}) \wedge x_n \; \Big) \;\vee\; \Big(\; B_{n-1}^s(x_1, \ldots, x_{n-1}) \wedge \overline{x}_n \; \Big)$$

and

$$r = (f_{2^n-1}, f_{2^n-2}, \ldots, f_{2^{n-1}+1}, f_{2^{n-1}})_2, \qquad s = (f_{2^{n-1}-1}, f_{2^{n-1}-2}, \ldots, f_1, f_0)_2$$

Proof.
This preposition is immediate by definition of the operation $B_n^p()$. Actually, since in the "zero-one" table definition of $B_n^p()$ the last variable, i.e. x_n has ones in the upper half and zeros in the lower half of it, then $B_{n-1}^r()$ and $B_{n-1}^s()$ are determined by halfs of that table (upper and lower ones, respectively). Thus

$$B_{n-1}^r(x_1, \ldots, x_{n-1}) = B_n^p(x_1, \ldots, x_{n-1}, 1)$$

$$B_{n-1}^s(x_1, \ldots, x_{n-1}) = B_n^p(x_1, \ldots, x_{n-1}, 0)$$

The conditions for r and s are obvious.

Two examples of such recursive decomposition are presented in Tabs. 4*a* and 4*b*. In Tab. 4*a* there are $n = 2, \;\; p = (1001)_2 = 9$ and thus $r = (10)_2 = 2, \;\; s = (01)_2 = 1$, and in Tab. 4*b* we have $n = 3, \;\; p = (10110101)_2 = 181, \;\; r = (1011)_2, \;\; s = (0101)_2 = 5$.

x_1	x_2	$B_1^r(x_1)$	$B_1^s(x_1)$	α $x_2 \wedge B_1^r(x_1)$	β $\overline{x}_2 \wedge B_1^s(x_1)$	$\alpha \vee \beta$	$B_2^9(x_1, x_2)$
1	1	1	0	1	0	1	1
0	1	0	1	0	0	0	0
1	0	1	0	0	0	0	0
0	0	0	1	0	1	1	1

Table 4a. Recursive decomposition of operation $B_2^9(x_1, x_2)$ based on *Preposition* from Section 2. α and β are used only to streamline the notation.

x_1	x_2	x_3	$B_2^r(x_1, x_2)$	$B_2^s(x_1, x_2)$	α $x_3 \wedge B_2^r(x_1, x_2)$	β $\overline{x}_3 \wedge B_2^s(x_1, x_2)$	$\alpha \vee \beta$	$B_3^{181}(x_1, x_2, x_3)$
1	1	1	1	0	1	0	1	1
0	1	1	0	1	0	0	0	0
1	0	1	1	0	1	0	1	1
0	0	1	1	1	1	0	1	1
1	1	0	1	0	0	0	0	0
0	1	0	0	1	0	1	1	1
1	0	0	1	0	0	0	0	0
0	0	0	1	1	0	1	1	1

Table 4b. Recursive decomposition of operation $B_3^{181}(x_1, x_2, x_3)$ (cf. *Preposition* in Section 2). α and β are used only to streamline the notation.

3 Network definition

The network is composed of two types of neurons. Thus we shall use two threshold functions: g_1 and g_2

$$g_1(z) = \begin{cases} 1 & \text{if} \quad z > 0 \\ 0 & \text{if} \quad z \leq 0 \end{cases}$$

and

$$g_2(z) = g_1(z-1) = \begin{cases} 1 & \text{if} \quad z > 1 \\ 0 & \text{if} \quad z \leq 1 \end{cases}$$

Let us note, for future use, that the conditions

$$\begin{aligned} &p = g_1(p), \qquad \overline{p} = g_1(1-p), \qquad p \vee q = g_1(p+q), \\ &p = g_2(p+p), \qquad p \wedge q = g_2(p+q) \end{aligned} \tag{1}$$

are fulfilled by any Boolean variables p, q. In other words, neurons with the threshold function g_1 (i.e. neurons of type g_1) can perform logical OR operation for the two Boolean inputs. Moreover, with the use of the external bias 1, they can perform logical NOT operation (this feature is only used in the first layer of the network). Similarily, neurons of type g_2 can act as a logical AND gate for the two Boolean inputs.

The network, for n input variables $x_1, \ldots, x_n$, is composed of $2n$ layers numbered 0, 1, 1^+, 2, 2^+, …, $(n-1)^+$, n. The 0-th layer is the input layer and the layer number n is the output one. Layers $1, 2, 3, \ldots, n$ are composed of neurons of type g_1. In layers $1^+, 2^+, \ldots, (n-1)^+$ neurons of type g_2 are used. Layers with superscript " + " are auxiliary ones and the others are main layers.

The example of the network for $n = 2$, i.e. performing all operations $B_2^k(x_1, x_2)$, $k = 0, \ldots, 15$ is presented in Fig. 1. In that case there are four layers: input layer, two hidden layers and the output one. In the input layer except for two input variables x_1, x_2 a node representing external bias 1 is added. Nodes in the first hidden layer represent functions $B_1^0(x_1), B_1^1(x_1), B_1^2(x_1), B_1^3(x_1)$ and variables $x_2, \overline{x}_2$. Nodes in the second hidden layer are created as $B_1^i(x_1) \wedge x_2$ and $B_1^i(x_1) \wedge \overline{x}_2$, $i = 0, \ldots, 3$. In the output layer there are sixteen possible basic logical operations $B_2^0(x_1, x_2), \ldots, B_2^{15}(x_1, x_2)$.

External input bias 1 is neccessary to implement operations that for totally zero input produce a non-zero output, e.g. $B_2^9(0, 0) = 1$. This external node is also indispensable in creating complements of the input variables in the way described in (1). We have taken into account two possible ways of creating these complements:

(a) adding the external node 1 to all but the two last layers in the network,

or

(b) using the external 1 only in the input layer.

Both of the above ways have their advantages as well as weak points. The reason why we chose variant (b) is explained in Section 5, together with the general description of the network features.

The first inconvenience of method (b) is that since we have decided to add external node 1 only in the input layer we have had to generate complements of all input variables yet in the first hidden layer. In fact, according to (1) we create the complement $\overline{x}$ of variable x using the external node 1 (with weight 1) and variable x (with weight -1) as the inputs to the neuron of type g_1 in layer 1.

The other inconvenience is connected with "moving" input variables with bigger indices and their complements through the hidden layers up to their destinations, i.e. the layers in which they are used (e.g. x_3 and $\overline{x}_3$ are used in layer 2^+, x_4 and $\overline{x}_4$ in layer 3^+, etc.). Since in auxiliary layers we use threshold function g_2 which returns 0 for any argument less or equal 1, we had to either use multiple wages or what we did in fact, duplicate some of the nodes. Namely, for every input variable x_i, as well as its complement $\overline{x}_i$, in the main layers $1, 2, \ldots, i-2$ we create two nodes: x_i and x_i' for x_i and $\overline{x}_i$, $\overline{x}_i'$ for $\overline{x}_i$ (see Fig. 2).

In the auxiliary layers there is one node representing x_i (and one representing $\overline{x}_i$) created from the two nodes: x_i and x_i' (respectively $\overline{x}_i$ and $\overline{x}_i'$) from the previous layer. This way in the auxiliary layers we obtain by (1):

$$g_2(x_i + x_i') = x_i \qquad \text{and} \qquad g_2(\overline{x}_i + \overline{x}_i') = \overline{x}_i$$

In other words, nodes x_i and x_i' representing x_i (and similarily nodes $\overline{x}_i$, $\overline{x}_i'$ representing $\overline{x}_i$) created in the main layers $1, 2, \ldots, i-2$ enable prolonging input variable x_i (and its complement $\overline{x}_i$) through auxiliary layers $1^+, 2^+, \ldots, (i-2)^+$ up to the layer $(i-1)^+$, in which x_i and $\overline{x}_i$ are used.

The above described (a little bit confusing) method of avoiding multiple and negative wages is schematically illustrated in Fig. 2. Now let us describe the network layer by layer. For n input variables $x_1, \ldots, x_n$, the following number of nodes is required:

- <u>in layer 0</u>; $n+1$ input nodes:
  $1,\ x_1,\ x_2, \ldots,\ x_n$.

- <u>in layer 1</u>; $4(n-2)+6$ nodes:
  $B_1^0(x_1),\ B_1^1(x_1),\ B_1^2(x_1),\ B_1^3(x_1)$ and x_2, x_2', and
  $x_3, x_3', \overline{x}_3, \overline{x}_3', \ldots, x_n, x_n', \overline{x}_n, \overline{x}_n'$.

- <u>in layer 1^+</u>; $2(n-2)+8$ nodes:
  $B_1^k(x_1) \wedge x_2,\ B_1^k(x_1) \wedge \overline{x}_2,\ k = 0, \ldots, 3$ and $x_3, \overline{x}_3, \ldots, x_n, \overline{x}_n$.
  Henceforth, for any $l \leq m < n$ we denote by
  ${}^+B_m^l(x_1, \ldots, x_{m+1})$ node representing $B_m^l(x_1, \ldots, x_m) \wedge x_{m+1}$ and by
  ${}^-B_m^l(x_1, \ldots, x_{m+1})$ node representing $B_m^l(x_1, \ldots, x_m) \wedge \overline{x}_{m+1}$ (both in layer m^+).

- <u>in layer 2</u>; $4(n-3)+18$ nodes:
  $B_2^k(x_1, x_2),\ k = 0, \ldots, 15$ and x_3, x_3', and $x_4, x_4', \overline{x}_4, \overline{x}_4', \ldots, x_n, x_n', \overline{x}_n, \overline{x}_n'$.

- <u>in layer 2^+</u>; $2(n-3)+32$ nodes:
  ${}^+B_2^k(x_1, x_2, x_3),\ {}^-B_2^k(x_1, x_2, x_3),\ k = 0, \ldots, 15$ and $x_4, \overline{x}_4, \ldots, x_n, \overline{x}_n$.

... ...

- <u>in layer m, $m \leq n$</u>; $2^{2^m} + 4(n-m-1)+2$ nodes:
  $B_m^k(x_1, \ldots, x_m),\ k = 0, \ldots, 2^{2^m}-1$ and $x_{m+1}, \overline{x}_{m+1}$ and
  $x_{m+2}, x_{m+2}', \overline{x}_{m+2}, \overline{x}_{m+2}', \ldots, x_n, x_n', \overline{x}_n, \overline{x}_n'$.

- <u>in layer m^+, $m \leq n-1$</u>; $2 \cdot 2^{2^m} + 2(n-m-1)$ nodes:
  ${}^+B_m^k(x_1, \ldots, x_{m+1}),\ {}^-B_m^k(x_1, \ldots, x_{m+1}),\ k = 0, \ldots, 2^{2^m}-1$ and
  $x_{m+2}, \overline{x}_{m+2}, \ldots, x_n, \overline{x}_n$.

... ...

- <u>in layer $(n-1)^+$</u>; $2 \cdot 2^{2^{n-1}}$ nodes:
  ${}^+B_{n-1}^k(x_1, \ldots, x_n),\ {}^-B_{n-1}^k(x_1, \ldots, x_n),\ k = 0, \ldots, 2^{2^{n-1}}-1$.

- <u>in layer n</u>; 2^{2^n} output nodes:
  $B_n^k(x_1, \ldots, x_n),\ k = 0, \ldots, 2^{2^n}-1$.

To describe the connection weights and structure let us denote by $\{\alpha, \beta\} = \gamma$ the fact that node α from previous layer is connected with node β from the next layer and weight of the connection is equal to γ. Then
1) There are $6(n-1)+9$ connections between nodes from layer 0 to layer 1:

$$\{x_1, B_1^0(x_1)\} = -1, \ \ \{x_1, B_1^1(x_1)\} = -1, \ \ \{x_1, B_1^2(x_1)\} = 1,$$

$$\{x_1, B_1^3(x_1)\} = 1, \ \ \{1, B_1^1(x_1)\} = 1, \ \ \{1, B_1^3(x_1)\} = 1,$$

and

$$\{x_i, x_i\} = 1, \ \ \{x_i, \overline{x}_i\} = -1, \ \ \{1, \overline{x}_i\} = 1, \ \ i = 2, \dots, n,$$

and

$$\{x_i, x_i'\} = 1, \ \ \{x_i, \overline{x}_i'\} = -1, \ \ \{1, \overline{x}_i'\} = 1, \ \ i = 3, \dots, n$$

2) All weights of connections between nodes from layer 1 to layer 1^+, from layer 1^+ to layer 2, and so on up to connections to the output layer are equal to 1, i.e. $\{\alpha, \beta\} = 1$ for all nodes α, β belonging to the two successive layers and α is not the input node.
3) Layer m^+, $1 \le m \le n-1$ is created from layer m in such a way that every node ${}^+B_m^i$, $i = 0, \dots, 2^{2^m}-1$ in layer m^+ is connected with two nodes: B_m^i and x_{m+1} from layer m and every node ${}^-B_m^i$, $i = 0, \dots, 2^{2^m}-1$ from layer m^+ is connected with nodes B_m^i and $\overline{x}_{m+1}$ from layer m.

Moreover, every node x_{m+j} and $\overline{x}_{m+j}$, $j = 2, \dots, n-m$ in layer m^+ is connected with the pair of nodes x_{m+j}, x'_{m+j} and $\overline{x}_{m+j}, \overline{x}'_{m+j}$, respectively from layer m. So, there are $4 \cdot 2^{2^m} + 4(n-m-1)$ connections from layer m to layer m^+:

$$\{B_m^i, {}^+B_m^i\} = 1, \ \ \{B_m^i, {}^-B_m^i\} = 1, \ \ \{x_{m+1}, {}^+B_m^i\} = 1, \ \ \{\overline{x}_{m+1}, {}^-B_m^i\} = 1, \ \ i = 0, \dots, 2^{2^m}-1.$$

and

$$\begin{aligned} &\{x_{m+j}, x_{m+j}\} = 1, \ \ \{\overline{x}_{m+j}, \overline{x}_{m+j}\} = 1, \\ &\{x'_{m+j}, x_{m+j}\} = 1, \ \ \{\overline{x}'_{m+j}, \overline{x}_{m+j}\} = 1, \quad j = 2, \dots, n-m. \end{aligned}$$

4) Layer $m+1$, $m+1 \le n$ is created from layer m^+. Nodes x_{m+2} and $\overline{x}_{m+2}$ are extended from layer m^+ to layer $m+1$. Every other node x_{m+j} as well as $\overline{x}_{m+j}$, $j = 3, \dots, n-m$ from layer m^+ creates two nodes: x_{m+j}, x'_{m+j} and $\overline{x}_{m+j}, \overline{x}'_{m+j}$, respectively, in layer $m+1$, that is

$$\{x_{m+j}, x_{m+j}\} = 1, \ \ \{\overline{x}_{m+j}, \overline{x}_{m+j}\} = 1, \quad j = 2, \dots, n-m,$$

and

$$\{x_{m+j}, x'_{m+j}\} = 1, \ \ \{\overline{x}_{m+j}, \overline{x}'_{m+j}\} = 1, \quad j = 3, \dots, n-m$$

Every node B_{m+1}^k, $k = 0, \dots, 2^{2^{m+1}-1}$ in layer $m+1$ is connected with two nodes from layer m^+: one with superscript " $+$ " and one with superscript " $-$ ". More precisely, the node B_{m+1}^k, $k = 0, \dots, 2^{2^{m+1}-1}$ has the connections

$$\{{}^+B_m^i, B_{m+1}^k\} = 1, \ \ \{{}^-B_m^j, B_{m+1}^k\} = 1,$$

where

$$i = Ent\Big(\frac{k}{2^{2^m}} \Big), \qquad j = k - 2^{2^m} \cdot Ent\Big(\frac{k}{2^{2^m}} \Big)$$

Totally there are $2 \cdot 2^{2^{m+1}} + 4(n-m-2) + 2$ connections between layers m^+ and $m+1$.
5) Layer number n is the output layer. It is composed of 2^{2^n} nodes in which values of operations $B_n^k(x_1, \dots, x_n)$, $k = 0, \dots, 2^{2^n - 1}$ are presented.

4 Matrix notation

Let us consider the network performing all operations $B_n^k(x_1, \dots, x_n)$, $k = 0, \dots, 2^{2^n} - 1$, and define for any Boolean vector $[y_1, \dots, y_m]$, $m \in \mathcal{N}$ casting operations

$$G_i([y_1, \dots, y_m]) = [g_i(y_1), \dots, g_i(y_m)], \qquad i = 1, 2$$

Denote also by IN_i, $i = 0, \dots, n$ and IN_{i^+}, $i = 1, \dots, n-1$, Boolean input vectors to the layers i and i^+, respectively, and similarly by OUT_i and OUT_{i^+} output vectors from these layers. For the input layer:

$$IN_0 = [1, x_1, \dots, x_n] \qquad \text{and} \qquad OUT_0 = IN_0$$

The total input to layer number 1 and output produced by this layer are as follows:

$$IN_1 = OUT_0 * W_0, \qquad \text{and} \qquad OUT_1 = G_1(IN_1),$$

where '$*$' denotes matrix product, and

$$W_0 = \underbrace{\left(\begin{array}{cccccccccccccc} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & \dots & 0 & 0 & 1 & 1 \\ -1 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & \\ & & & & 1 & -1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ & & & & 0 & 0 & 1 & 1 & -1 & -1 & \dots & 0 & 0 & 0 & 0 \\ & 0 & & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ & & & & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 1 & 1 & -1 & -1 \end{array} \right)}_{2^{2^0} + 4(n-2) + 2 \text{ columns}} n+1 \text{ rows}$$

Rows of matrix W_0 correspond to $n+1$ nodes in layer 0 and its columns to $2^{2^0} + 4(n-2) + 2$ nodes in layer 1 (see Section 3 for details).

Similarly the total input to layer 1^+ and output of this layer are equal to:

$$IN_{1^+} = OUT_1 * W_1, \qquad \text{and} \qquad OUT_{1^+} = G_2(IN_{1^+}),$$

where

$$W_1 = \underbrace{\left(\begin{array}{cccccccc|cccc}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & & & & \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & & & & \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & & 0 & & \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & & & & \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & & & & \\
 & & & & & & & & 1 & 0 & \dots & 0 \\
 & & & & & & & & 1 & 0 & \dots & 0 \\
 & & & & & & & & 0 & 1 & \dots & 0 \\
 & & & & & & & & 0 & 1 & \dots & 0 \\
 & & & & 0 & & & & \vdots & \vdots & \ddots & \vdots \\
 & & & & & & & & 0 & 0 & \dots & 1 \\
 & & & & & & & & 0 & 0 & \dots & 1
\end{array}\right)}_{2 \cdot 2^{2^1} + 2(n-2) \text{ columns}} 2^{2^0} + 4(n-2) + 2 \text{ rows}$$

Rows and columns of matrix W_1 correspond to nodes in layers 1 and 1^+, respectively. Equations describing the next layers can be stated in the similar way:

$$IN_{i^+} = OUT_i * W_i, \qquad OUT_{i^+} = G_2(IN_{i^+})$$

and

$$IN_{i+1} = OUT_{i^+} * W_{i^+}, \qquad OUT_{i+1} = G_1(IN_{i+1})$$

For the formal description of matricies W_i and W_{i^+} we introduce some helpful notation: $\Phi^{r,s}$ - matrix $r \times s$ with all elements equal to 0, i.e.

$$\Phi^{r,s}[i,j] = 0, \qquad i = 1, \dots, r, \;\; j = 1, \dots, s$$

J^r - matrix $r \times 2r$ such that

$$J^r[i,j] = \begin{cases} 1 & 2i = j \text{ or } 2i = j+1 \\ 0 & \text{in other cases} \end{cases} \qquad i = 1, \dots, r, \;\; j = 1, \dots, 2r$$

$I^{r,s}$ - matrix composed of s copies of unit matrix $r \times r$ placed one after the other (in a row), that is

$$I^{r,s}[i,j] = \begin{cases} 1 & j - i \mid r \\ 0 & \text{in other cases} \end{cases} \qquad i = 1, \dots, r, \;\; j = 1, \dots, rs$$

where '|' denotes divisibility.

Finally the definition of the more complicated matrix K^r - composed of 2^{2^r+1} rows and 2^{2^r+1} columns such that

$$K^r[i,j] = \begin{cases} 1 & (\; i - 1 \mid 2 \text{ and } \frac{i-1}{2} 2^{2^r} < j \le \frac{i+1}{2} 2^{2^r} \;) \text{ or } (\; j - \frac{i}{2} \mid 2^{2^r} \;) \\ 0 & \text{in other cases} \end{cases}$$

$$i = 1, \dots, 2^{2^r+1}, \;\; j = 1, \dots, 2^{2^r+1}$$

Matrix K^r is presented in Fig. 3.

With above notation each matrix $W_i,\ 1 \le i \le n$ can be depicted as follows:

$$W_i = \left(\begin{array}{cc} J^{2^{2^i}} & \\ & \Phi^{2+2^{2^i},\, 2(n-i-1)} \\ I^{2,\, 2^{2^i}} & \\ \Phi^{4(n-i-1),\, 2^{2^i}+1} & \left(J^{2(n-i-1)}\right)^T \end{array}\right) 2^{2^i}+4(n-i-1)+2 \text{ rows}$$

$$2^{2^i+1}+2(n-i-1) \text{ columns}$$

First 2^{2^i} rows of matrix W_i correspond to nodes $B_i^0,\ B_i^1,\ \ldots,\ B_i^{2^{2^i}-1}$ in layer i. Next two rows correspond to nodes x_{i+1} and $\overline{x}_{i+1}$ in layer i. Finally, $4(n-i-1)$ left ones correspond to nodes $x_{i+2},\ x'_{i+2}, \overline{x}_{i+2}, \overline{x}'_{i+2},\ \ldots, x_n, x'_n, \overline{x}_n,\ \overline{x}'_n$ in this layer.

Columns are responsible for nodes in layer i^+. First 2^{2^i+1} of them correspond to nodes ${}^+B_i^0, {}^-B_i^0, \ldots,\ {}^+B_i^{2^{2^i}-1},\ {}^-B_i^{2^{2^i}-1}$, and the other $2(n-i-1)$ to nodes $x_{i+2}, \overline{x}_{i+2}, \ldots,\ x_n, \overline{x}_n$.

For any auxiliary layer $i^+,\ 1 \le i < n$ we have:

$$W_{i^+} = \left(\begin{array}{ccc} K^i & & \Phi^{2^{2^i+1},\, 4(n-i-2)+2} \\ \Phi^{2,\, 2^{2^{i+1}}} & I^{2,\, 1} & \Phi^{2,\, 4(n-i-2)} \\ \Phi^{2(n-i-1),\, 2+2^{2^{i+1}}} & & J^{2(n-i-2)} \end{array}\right) 2^{2^i+1}+2(n-i-1) \text{ rows}$$

$$2^{2^{i+1}}+4(n-i-2)+2 \text{ columns}$$

where rows correspond to nodes in layer i^+ (see description above), and columns to nodes in layer $i+1$. First $2^{2^{i+1}}$ columns describe nodes $B_{i+1}^0,\ \ldots,\ B_{i+1}^{2^{2^{i+1}}-1}$ and the rest of them nodes $x_{i+2},\ \overline{x}_{i+2},\ x_{i+3},\ x'_{i+3},\ \overline{x}_{i+3},\ \overline{x}'_{i+3},\ \ldots,\ x_n,\ x'_n,\ \overline{x}_n,\ \overline{x}'_n$.

5 Description of the network features

What are characteristic features of the network ?

First, the network has "open" structure. Any main layer, say for example layer number l, can work as the output layer. In its nodes values of 2^{2^l} operations $B_l^k()$, $k = 0, \ldots, 2^{2^l-1}$ of l variables $x_1, \ldots, x_l$ are produced. Thus the network, actually performs all operations of the input variables $x_1, \ldots, x_l$, for every $l \leq n$. Such a structure also provides easy way for constructing a network realizing any single fixed operation, not performing all of them together.

The above feature as well as possibilities of modular combining parts of the network in order to create networks performing more complex tasks are broadly described in next sections.

As mentioned before we cannot avoid external bias 1 to produce a non-zero output from totally zero input, but in construction presented above we need external 1 only in the input layer. The cost of this is that we have to generate complements of all input variables yet in the first hidden layer, though they are used in further layers. Moreover, we have to duplicate some nodes in the main layers to prevent input variables from being reset in the auxiliary layers.

The alternative solution is to prolonge external node 1 from the input layer through all hidden layers up to the last but one of them. This way it is possible to obtain complements of input variables at the last moment. Strictly speaking value of $\overline{x}_l$, $2 \leq l \leq n$ is created in layer $(l-1)^+$. On the other hand, using external node 1 in every layer and applying negative wages in further layers (to obtain complements of the input variables) seems to be "not elegant" and artificial.

Moreover, the chosen model provides very simple mechanism for creation of layers. From layer 1 up to the output layer all weights of connections are equal to 1 which makes structure of the network symmetric and clear. Avoiding artificial external nodes in the hidden layers as well as multiple and negative wages except of a few negative ones between the input layer and the first hidden layer seems promissing for the practical computer implementation of the model. Such an approach is also more intuitive for Boolean zero-one logic as well as for the low-level computer work than the other one.

These are the reasons why we have sticked to the model with only one extra node, added in the input layer. Examples of both networks performing operation $B_3^{181}(x_1, x_2, x_3)$ are presented in Figs. 4*a* and 4*b*.

The main disadvantage of the proposed network is a big number of nodes required. On the other hand, in the network, there are parallelly implemented all basic operations of given number of input variables and the velocity of them grows extremely rapidly (2^{2^n} operations of n variables). Moreover, as stated above, the network performing operations for n input variables $x_1, \ldots, x_n$ additionally performs all operations of the first l variables $x_1, \ldots, x_l$, for every $l \leq n$.

Denoting by $N(k)$ and $N(k^+)$ the number of nodes required in the k-th main layer and the k-th auxiliary layer, respectively we get

$$N(k) = \begin{cases} n+1 & \text{if} \quad k = 0 \\ 2^{2^k} + 4(n-k-1) + 2 & \text{if} \quad 0 \neq k \neq n \\ 2^{2^n} & \text{if} \quad k = n \end{cases}$$

$$N(k^+) = 2^{2^k+1} + 2(n-k-1) \qquad k = 1, \ldots, n-1$$

Thus for N_n - the total number of nodes in the network for n variables we have the equation

$$N_n = \sum_{k=0}^{n} N(k) + \sum_{k=1}^{n-1} N(k^+) = 2^{2^n} + 3n^2 - 6n + 5 + 3\sum_{k=1}^{n-1} 2^{2^k}$$

First four values of N_n are: $N_1 = 6,\ N_2 = 33,\ N_3 = 330,\ N_4 = 2^{16} + 857$.

6 Performing any single, fixed operation

From practical point of view it may be more interesting to answer the question how to construct a network performing any fixed, arbitrarily chosen operation. The answer is very simple, straight from *Preposition.*

Consider $B_n^{\alpha_1}(x_1, \ldots, x_n)$, where $\alpha_1 = (f_{2^n-1}, f_{2^n-2}, \ldots, f_1, f_0)_2$. Which operations of less than n variables must be implemented before $B_n^{\alpha_1}()$ is reached ?

Applaying $n-1$ steps depth decomposition procedure from *Preposition* we obtain:

- <u>first step</u>

$$B_n^{\alpha_1}(x_1, \ldots, x_n) \rightarrow \begin{cases} B_{n-1}^{\alpha_2}(x_1, \ldots, x_{n-1}) & \alpha_2 = (f_{2^n-1}, \ldots, f_{2^{n-1}})_2 \\ B_{n-1}^{\alpha_3}(x_1, \ldots, x_{n-1}) & \alpha_3 = (f_{2^{n-1}-1}, \ldots, f_0)_2 \end{cases}$$

- <u>second step</u>

$$B_{n-1}^{\alpha_2}(x_1, \ldots, x_{n-1}) \rightarrow \begin{cases} B_{n-2}^{\alpha_4}(x_1, \ldots, x_{n-2}) & \alpha_4 = (f_{2^n-1}, \ldots, f_{2^n-2^{n-2}})_2 \\ B_{n-2}^{\alpha_5}(x_1, \ldots, x_{n-2}) & \alpha_5 = (f_{2^n-2^{n-2}-1}, \ldots, f_{2^{n-1}})_2 \end{cases}$$

$$B_{n-1}^{\alpha_3}(x_1, \ldots, x_{n-1}) \rightarrow \begin{cases} B_{n-2}^{\alpha_6}(x_1, \ldots, x_{n-2}) & \alpha_6 = (f_{2^{n-1}-1}, \ldots, f_{2^{n-2}})_2 \\ B_{n-2}^{\alpha_7}(x_1, \ldots, x_{n-2}) & \alpha_7 = (f_{2^{n-2}-1}, \ldots, f_0)_2 \end{cases}$$

$$\ldots \qquad \ldots \qquad \ldots$$

- <u>$k-$th step</u>

$$B_{n-k+1}^{\alpha_{2^{k-1}}}(x_1, \ldots, x_{n-k+1}) \rightarrow \begin{cases} B_{n-k}^{\alpha_{2^k}}(x_1, \ldots, x_{n-k}) \\ B_{n-k}^{\alpha_{2^k+1}}(x_1, \ldots, x_{n-k}) \end{cases}$$

$$\alpha_{2^k} = (f_{2^n-1}, \ldots, f_{2^n-2^{n-k}})_2, \qquad \alpha_{2^k+1} = (f_{2^n-2^{n-k}-1}, \ldots, f_{2^n-2^{n-k+1}})_2$$

$$\ldots$$

$$B_{n-k+1}^{\alpha_{2^k-1}}(x_1,\ldots,x_{n-k+1}) \rightarrow \begin{cases} B_{n-k}^{\alpha_{2^{k+1}-2}}(x_1,\ldots,x_{n-k}) \\ \\ B_{n-k}^{\alpha_{2^{k+1}-1}}(x_1,\ldots,x_{n-k}) \end{cases}$$

$$\alpha_{2^{k+1}-2} = (f_{2^{n-k+1}-1},\ldots,f_{2^{n-k}})_2, \qquad \alpha_{2^{k+1}-1} = (f_{2^{n-k}-1},\ldots,f_0)_2$$

$$\ldots \qquad \ldots \qquad \ldots$$

- $\underline{(n-2)\text{−th step}}$

$$B_3^{\alpha_{2^{n-3}}}(x_1,x_2,x_3) \rightarrow \begin{cases} B_2^{\alpha_{2^{n-2}}}(x_1,x_2) & \alpha_{2^{n-2}} = (f_{2^n-1},f_{2^n-2},f_{2^n-3},f_{2^n-4})_2 \\ \\ B_2^{\alpha_{2^{n-2}+1}}(x_1,x_2) & \alpha_{2^{n-2}+1} = (f_{2^n-5},f_{2^n-6},f_{2^n-7},f_{2^n-8})_2 \end{cases}$$

$$\ldots$$

$$B_3^{\alpha_{2^{n-2}-1}}(x_1,x_2,x_3) \rightarrow \begin{cases} B_2^{\alpha_{2^{n-1}-2}}(x_1,x_2) & \alpha_{2^{n-1}-2} = (f_7,f_6,f_5,f_4)_2 \\ \\ B_2^{\alpha_{2^{n-1}-1}}(x_1,x_2) & \alpha_{2^{n-1}-1} = (f_3,f_2,f_1,f_0)_2 \end{cases}$$

- $\underline{(n-1)\text{−th step}}$

$$B_2^{\alpha_{2^{n-2}}}(x_1,x_2) \rightarrow \begin{cases} B_1^{\alpha_{2^{n-1}}}(x_1) & \alpha_{2^{n-1}} = (f_{2^n-1},f_{2^n-2})_2 \\ \\ B_1^{\alpha_{2^{n-1}+1}}(x_1) & \alpha_{2^{n-1}+1} = (f_{2^n-3},f_{2^n-4})_2 \end{cases}$$

$$\ldots$$

$$B_2^{\alpha_{2^{n-1}-1}}(x_1,x_2) \rightarrow \begin{cases} B_1^{\alpha_{2^n-2}}(x_1) & \alpha_{2^n-2} = (f_3,f_2)_2 \\ \\ B_1^{\alpha_{2^n-1}}(x_1) & \alpha_{2^n-1} = (f_1,f_0)_2 \end{cases}$$

Finally, in order to implement $B_n^{\alpha_1}()$ there must be implemented 2^n-2 previous operations (of less than n variables). Note that for bigger values of n many of them are redundant (not unique). For example in case $B_3^{181}(x_1,x_2,x_3)$, i.e. $n=3$, $\alpha_1 = 181 = (10110101)_2$ (see Fig. 4*a* and Tab. 4*b*) we need

$$B_1^{\alpha_1}(x_1,x_2,x_3) \rightarrow \begin{cases} B_2^{\alpha_2}(x_1,x_2) \rightarrow \begin{cases} B_1^{\alpha_4}(x_1) \\ \\ B_1^{\alpha_5}(x_1) \end{cases} \\ \\ B_2^{\alpha_3}(x_1,x_2) \rightarrow \begin{cases} B_1^{\alpha_6}(x_1) \\ \\ B_1^{\alpha_7}(x_1) \end{cases} \end{cases}$$

where

$$\alpha_2 = (1011)_2 = 11, \qquad \alpha_3 = (0101)_2 = 5,$$

$$\alpha_4 = (10)_2 = 2, \qquad \alpha_5 = (11)_2 = 3, \qquad \alpha_6 = \alpha_7 = (01)_2 = 1$$

In the first step we divide $(10110101)_2$ into $(1011)_2$ and $(0101)_2$, that is α_2 and α_3, respectively. In the second step we obtain $(10)_2$ and $(11)_2$ (α_4 and α_5) from α_2 and $(01)_2$, $(01)_2$ (α_6, α_7) from α_3.

7 Performing more complex tasks

Once we are able to perform all operations $B_n^k()$ (for fixed n) parallelly, or any single one of them alone, we can accomplish some more complex tasks. One of the promissing features of the network is its ability to be combined into more complex structures. Suppose for example, that a network which for any Boolean input variables x_1, x_2, x_3 outputs value of

$$B_2^8(\ B_2^4(x_1, x_2),\ B_3^{200}(x_1, x_2, x_3)\ ) \tag{2}$$

is required (see Tab. 5).

x_1	x_2	x_3	$B_2^4(x_1,x_2)$	$B_3^{200}(x_1,x_2,x_3)$	$B_2^8(B_2^4(x_1,x_2), B_3^{200}(x_1,x_2,x_3))$
1	1	1	0	1	0
0	1	1	1	1	1
1	0	1	0	0	0
0	0	1	0	0	0
1	1	0	0	1	0
0	1	0	1	0	0
1	0	0	0	0	0
0	0	0	0	0	0

Table 5. Presentation of logical function $B_2^8(\ B_2^4(x_1, x_2),\ B_3^{200}(x_1, x_2, x_3)\ )$ by a zero-one table.

A diagram of the network performing logical function (2) is presented in Fig. 5. In the first step, operations $B_2^4(x_1, x_2)$, $B_3^{200}(x_1, x_2, x_3)$ are parallelly realized in two separate networks, having only input in common. Then outputs of those networks plus additional external 1 work as inputs to the next network performing operation $B_2^8(p, q)$, where p, q are Boolean variables.

The proposed network can also be used for neural implementation of First Order logic (FO) [13]. Since the details of this implementation require further research we would only like to introduce here the general concept.

Let us assume that we have a neural network able to sequentially generate all possible sequences of zeros and ones (zero-one mappings) of length n, where n is fixed, that is:

$$\underbrace{(0, 0, \ldots, 0, 0)}_{n \text{ times}},\ \underbrace{(0, 0, \ldots, 0, 1)}_{n \text{ times}},\ \ldots\ ,\underbrace{(1, 1, \ldots, 1, 0)}_{n \text{ times}},\ \underbrace{(1, 1, \ldots, 1, 1)}_{n \text{ times}}$$

Such a neural device (treated as a "black-box" here) together with the network proposed in the paper allow to implement classical FO over the language

$$\mathcal{L}_n = \left\langle\ B_n^k(x_1, \ldots, x_n)\ :\ n - \text{ fixed},\ 0 \le k < 2^{2^n}\ \right\rangle$$

What is to be implemented are quantifiers $\forall$ and $\exists$.

The existential quantifier ($\exists$) can be realized as the sum of results of the performed operation for all possible zero-one sequences (there are 2^n of them, of course), with threshold function g_1.

Similarly, the general quantifier ($\forall$) can be implemented as the logical product of these 2^n results, or more simply by the sum of all of them, decremented by 1 each, plus additional 1 (with threshold function g_1).

As the example let us consider Boolean formulas:

$$\forall x_1 \; \forall x_2 \quad B_2^3(x_1, x_2) \qquad \text{and} \qquad \exists x_1 \; \exists x_2 \quad B_2^3(x_1, x_2)$$

For the two variables x_1 and x_2 there are four possible zero-one mappings: $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$. The condition

$$\forall x_1 \; \forall x_2 \quad B_2^3(x_1, x_2)$$

can be realized as

$$g_1(\ (B_2^3(0,0)-1)\ +\ (B_2^3(0,1)-1)\ +\ (B_2^3(1,0)-1)\ +\ (B_2^3(1,1)-1)\ +\ 1\ )$$

and

$$\exists x_1 \; \exists x_2 \quad B_2^3(x_1, x_2)$$

is equivalent to

$$g_1(\ B_2^3(0,0)\ +\ B_2^3(0,1)\ +\ B_2^3(1,0)\ +\ B_2^3(1,1)\ )$$

The general quantifier can also be accomplished according to the rule: "work until non-zero output is generated". This way the answer $FALSE$ is achieved when for any zero-one mapping the output of the network (performing operation $B_2^3(x_1, x_2)$, in our example) is equal to 0, otherwise the answer is $TRUE$.

Certainly, the rule for the existential quantifier is: "work until the output is equal to zero". The result is $TRUE$ if for any mapping non-zero output is generated, otherwise the result is $FALSE$.

As one can see it is easy to implement open formulas and formulas with exactly one type of quantifiers (general or existential). Implementing formulas with mixed quantifiers is not so natural and is of present research.

Since we are able to accomplish the classical FO we may think about its extensions by adding majority or modulo quantifiers [14]:

$$FO + MAJ_n \qquad \text{or} \qquad FO + Qmod_a$$

Let us remain that the condition

$$MAJ_n x_1, \ldots, x_n \quad f(x_1, \ldots, x_n) \quad \text{is } TRUE \quad \text{if and only if} \tag{3}$$
$$\overline{\overline{\{(x_1, \ldots, x_n) :\ f(x_1, \ldots, x_n) \text{ is } TRUE\}}} > \overline{\overline{\{(x_1, \ldots, x_n) :\ f(x_1, \ldots, x_n) \text{ is } FALSE\}}}$$

where double-line over the set denotes its power.

The majority quantifier can be carried out by the doubled sum of all results decreased by 1 each, with 1 substracted from the whole (with threshold function g_1). For example

$$MAJ_2 x_1, x_2 \qquad B_2^3(x_1, x_2)$$

can be realized by the formula

$$g_1(\ (2B_2^3(0,0) - 1)\ +\ (2B_2^3(0,1) - 1)\ +\ (2B_2^3(1,0) - 1)\ +\ (2B_2^3(1,1) - 1)\ -\ 1\ ) \tag{4}$$

When in (3) the weak inequality is allowed then the majority quantifier is nothing else but the general one with threshold equal to $\frac{1}{2}$ and therefore the condition (4) must be modified to

$$g_1(\ (2B_2^3(0,0) - 1)\ +\ (2B_2^3(0,1) - 1)\ +\ (2B_2^3(1,0) - 1)\ +\ (2B_2^3(1,1) - 1)\ +\ 1\ )$$

The authors have also tried to implement the modulo quantifier $Qmod_a$:

$$Qmod_a \quad f(x_1, \ldots, x_n) \quad \text{is } TRUE \quad \text{if and only if} \tag{5}$$
$$\overline{\overline{\{(x_1, \ldots, x_n) : f(x_1, \ldots, x_n) \text{ is } TRUE\}}} \ \mid\ a$$

This task is more complicated and so far we are unable to answer how to carry it out.

Quantifiers described above may also be implemented simplier in case the threshold functions other than g_1 are used, especially when the threshold depends on n. In language $\mathcal{L}_n$, the general quantifier can be simulated as a multiple-AND gate (threshold equal to 2^n), the existential one as multiple-OR gate (threshold equal to 1 - as in our implementation) and the majority quantifier as a multiple-AND gate with threshold equal to 2^{n-1} (with or without external -1, depends on the inequality in (3)).

We suppose that the modulo quantifier (5) can be accomplished by a combination of multiple gates with threshold equal to a but the precise answer is unknown to us.

8 Conclusions

Neural networks are finding many areas of applications. Although they are particularly well suited for applications related to associative recall such as content addressable memories, neural nets can be used in various other applications ranging from logic operations to solving difficult optimisation problems.

The proposed network model can either work as the independent universal logic module or be combined with neural network associative memories to accomplish complex combinations of data and logic operations on corrupted input data.

It is well suited in the electronical (hardware) context. Except for a few connection weights between the input layer and the first hidden layer all of weights are equal to 1. Thus, weights in the network can be looked at as electronical switches.

The problem of performing *any* desired logical operation for *any*, arbitrarily constrained, number of input variables can be handled by the universal hardware, based on the full network, with the programmable switches. This way is much more effective than using specific hardware dedicated to every single operation.

The network is still under research. We work on full implementation of First Order logic (accomplishing formulas with mixed quantifiers), and on implementation of the modulo quantifier. We also plan to define the network and its behaviour strictly in the logical terminology.

Acknowledgments

Thanks are due to Dorota Krauze for her careful reading of the manuscript and suggestions concerning implementation of First Order logic and its extensions.

References

[1] G. A. Carpenter, S. Grossberg, C. Mehanian, "Invariant Recognition of Cluttered Scenes by a Self-Organizing ART Architecture: Cort-X Boundary Segmentation", Neural Networks, **2**, 169-181, 1989

[2] A. Rajavelu, M. T. Musavi, M. V. Shirvaikar, "A Neural Network Approach to Character Recognition", Neural Networks, **2**, 387-393, 1989

[3] T. Kohonen, "Adaptive, associative, and self-organizing functions in neural computing", Applied Optics, **26**, 4910-4918, 1987

[4] H. H. Arsenault, B. Macukow, "Neural network model for fast learning and retrieval", Optical Engineering, **28**, 506-512, 1989

[5] X. Xu, W. T. Tsai, "Constructing Associative Memories Using Neural Networks", Neural Networks, **3**, 301-309, 1990

[6] A. R. Bizzarri, "Convergence properties of a modified Hopfield-Tank model", Biological Cybernetics, **64**, 293-300, 1991

[7] J. Mańdziuk, B. Macukow, "A neural network designed to solve the N-Queens Problem", Biological Cybernetics, **66**, 375-379, 1992

[8] M. H. Hassoun, R. Arrathoon, "Logical signal processing with optically connected threshold gates", Optical Engineering, **25**, 56-68, 1986

[9] M. J. Murdoca, A. Huang, J. Jahns, N. Streibl, "Optical design of programmable logic arrays", Applied Optics, **27**, 1651-1660, 1988

[10] P. S. Guilfoyle, W. J. Wiley, "Combinational logic based on digital optical computing architectures", Applied Optics, **27**, 1661-1673, 1988

[11] B. M. Macukow, H. H. Arsenault, "Neural networks for logic operations", Proceedings SPIE, **1134**, 40-43, 1989

[12] B. M. Macukow, H. H. Arsenault, "Logic operations with neural networks", Control and Cybernetics, **20**, 116-133, 1991

[13] J. R. Shoenfield, *Mathematical logic*, Reading, MA: Addison - Wesley, 1967

[14] D. A. Barrington, N. Immerman, H. Straubing, "On uniformity within NC^1", Proceedings 3rd Annual Conference Structure in Complexity Theory, IEEE, 47-59, 1988

Figure captions.

Figure 1. The example of proposed neural network model for two variables. Lines with arrowheads denote connection weights equal to -1, while the rest of them is equal to 1.

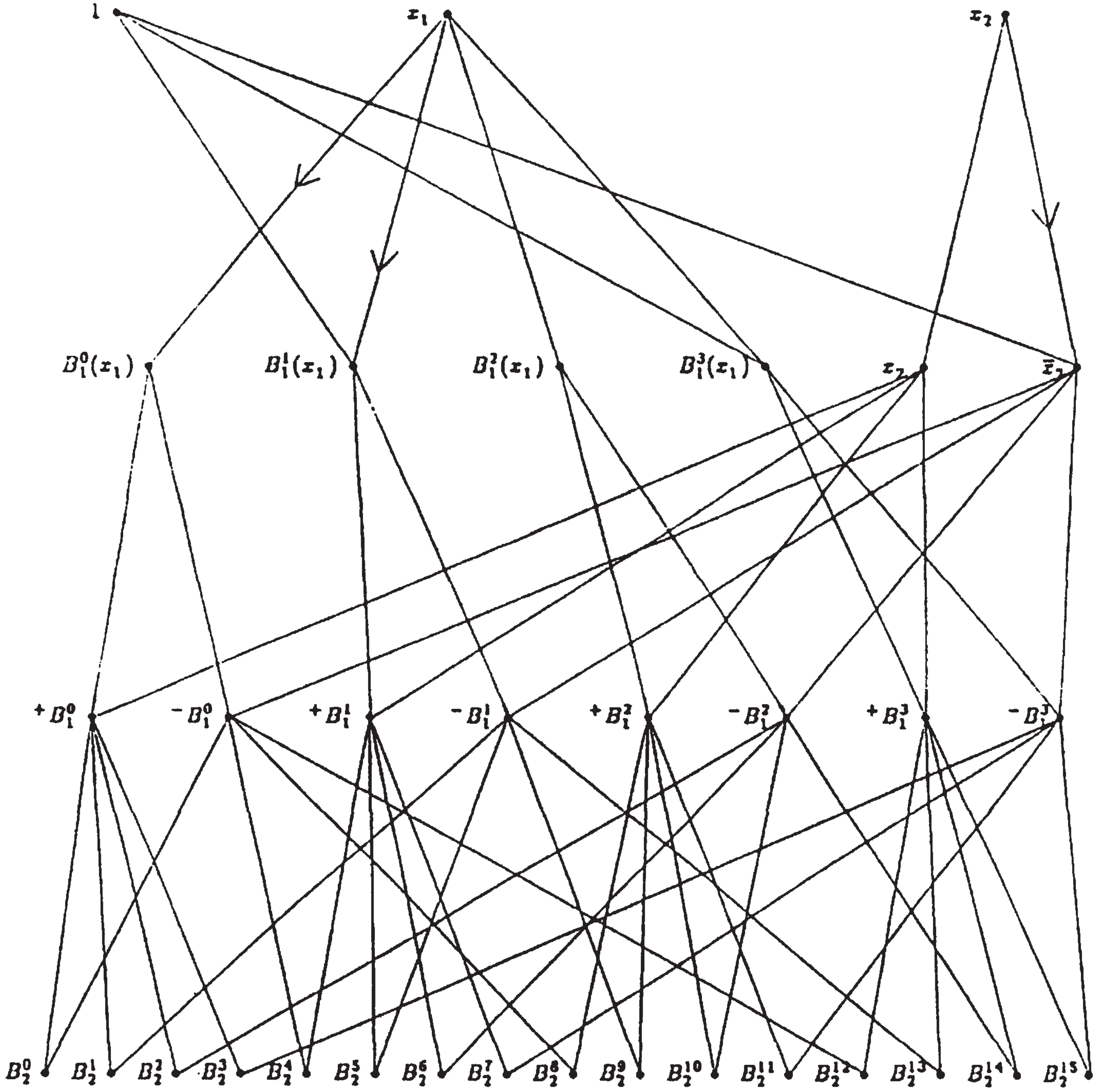
Figure 2. Presentation of the structure of nodes representing first four input variables $x_1, \ldots, x_4$ (and their complements) in the successive layers of the network.

Figure 3. A general form of matrix K^r, for $r \in \mathcal{N}$. In the picture columns are divided into 2^{2^r} groups, separated by characters $' \,|'$. Each group consists of 2^{2^r} columns.

Figures 4a and 4b. In Figs. $4a$ and $4b$ there are presented two networks performing operation $B_3^{181}(x_1, x_2, x_3)$ (see Section 5). In the model proposed in Fig. $4a$ the external node 1 appears only in the input layer. In the alternative network (Fig. $4b$) external node 1 is prolonged from the input layer to the next ones. In both pictures lines with arrowheads denote connection weights equal to -1, while the rest of them is equal to 1.

Figure 5. A schematical diagram of the network performing logical function $B_2^8(\ B_2^4(x_1, x_2)$, $B_3^{200}(x_1, x_2, x_3)\ )$ (see Section 7).

Figure 1



1
x_1
x_2
$B_1^0(x_1)$
$B_1^1(x_1)$
$B_1^2(x_1)$
$B_1^3(x_1)$
x_2
$\bar{x}_2$
$+B_1^0$
$-B_1^0$
$+B_1^1$
$-B_1^1$
$+B_1^2$
$-B_1^2$
$+B_1^3$
$-B_1^3$
B_2^0
B_2^1
B_2^2
B_2^3
B_2^4
B_2^5
B_2^6
B_2^7
B_2^8
B_2^9
B_2^{10}
B_2^{11}
B_2^{12}
B_2^{13}
B_2^{14}
B_2^{15}

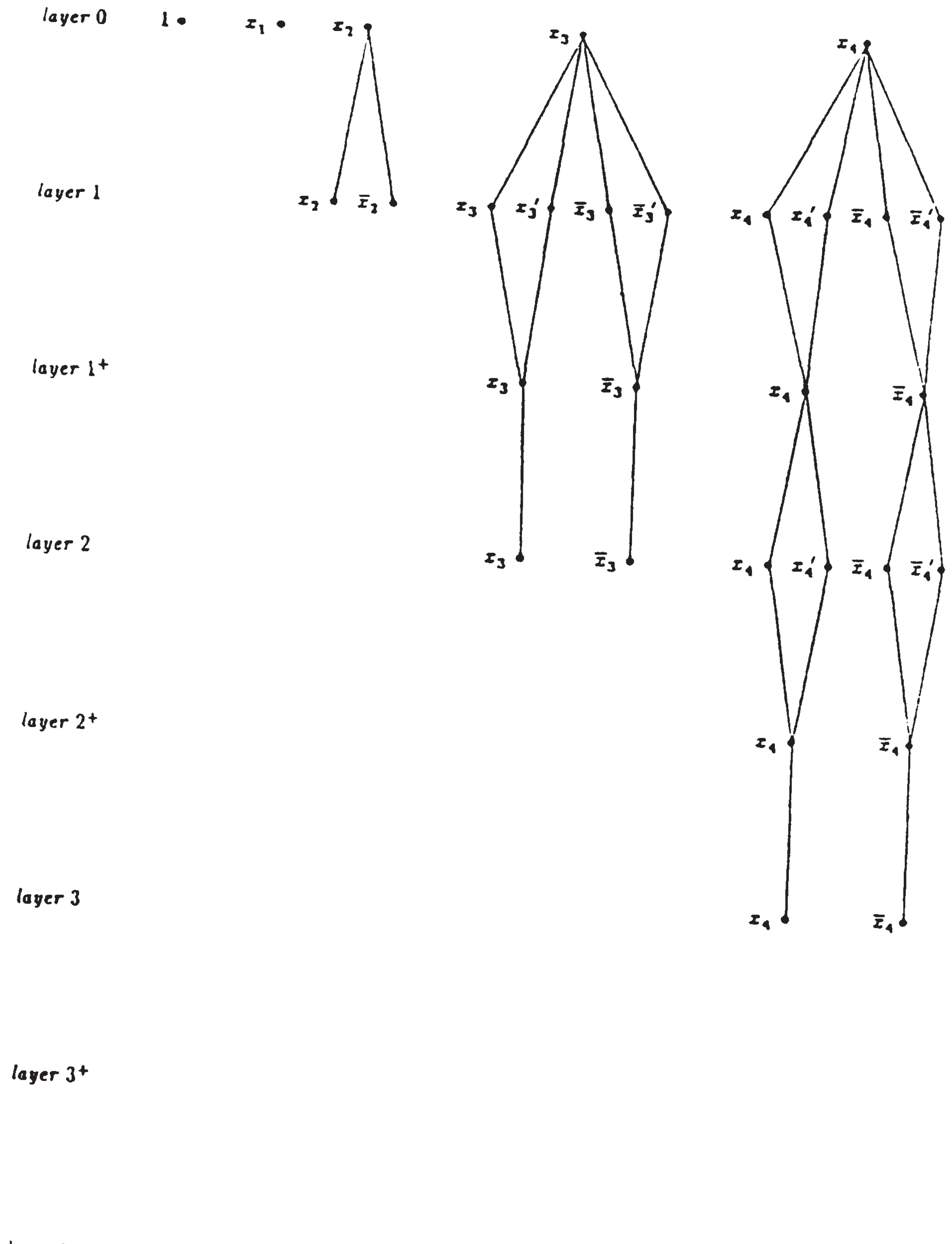

layer 0
1
x_1
x_2
x_3
x_4
layer 1
x_2
$\bar{x}_2$
x_3
x_3'
$\bar{x}_3$
$\bar{x}_3'$
x_4
x_4'
$\bar{x}_4$
$\bar{x}_4'$
layer 1^+
x_3
$\bar{x}_3$
x_4
$\bar{x}_4$
layer 2
x_3
$\bar{x}_3$
x_4
x_4'
$\bar{x}_4$
$\bar{x}_4'$
layer 2^+
x_4
$\bar{x}_4$
layer 3
x_4
$\bar{x}_4$
layer 3^+
layer 4

Figure 2

$$K^r = \begin{pmatrix}
1 & 1 & \dots & 1 & 1 & | & 0 & 0 & \dots & 0 & 0 & | & 0 & 0 & \dots & 0 & 0 & | & \cdots & | & 0 & 0 & \dots & 0 & 0 \\
1 & 0 & \dots & 0 & 0 & | & 1 & 0 & \dots & 0 & 0 & | & 1 & 0 & \dots & 0 & 0 & | & \cdots & | & 1 & 0 & \dots & 0 & 0 \\
0 & 0 & \dots & 0 & 0 & | & 1 & 1 & \dots & 1 & 1 & | & 0 & 0 & \dots & 0 & 0 & | & \cdots & | & 0 & 0 & \dots & 0 & 0 \\
0 & 1 & \dots & 0 & 0 & | & 0 & 1 & \dots & 0 & 0 & | & 0 & 1 & \dots & 0 & 0 & | & \cdots & | & 0 & 1 & \dots & 0 & 0 \\
 & & \vdots & & & | & & & \vdots & & & | & & & \vdots & & & | & \ddots & | & & & \vdots & & \\
0 & 0 & \dots & 0 & 0 & | & 0 & 0 & \dots & 0 & 0 & | & 0 & 0 & \dots & 0 & 0 & | & \cdots & | & 1 & 1 & \dots & 1 & 1 \\
0 & 0 & \dots & 0 & 1 & | & 0 & 0 & \dots & 0 & 1 & | & 0 & 0 & \dots & 0 & 1 & | & \cdots & | & 0 & 0 & \dots & 0 & 1
\end{pmatrix}$$

Figure 3

Figure 4a

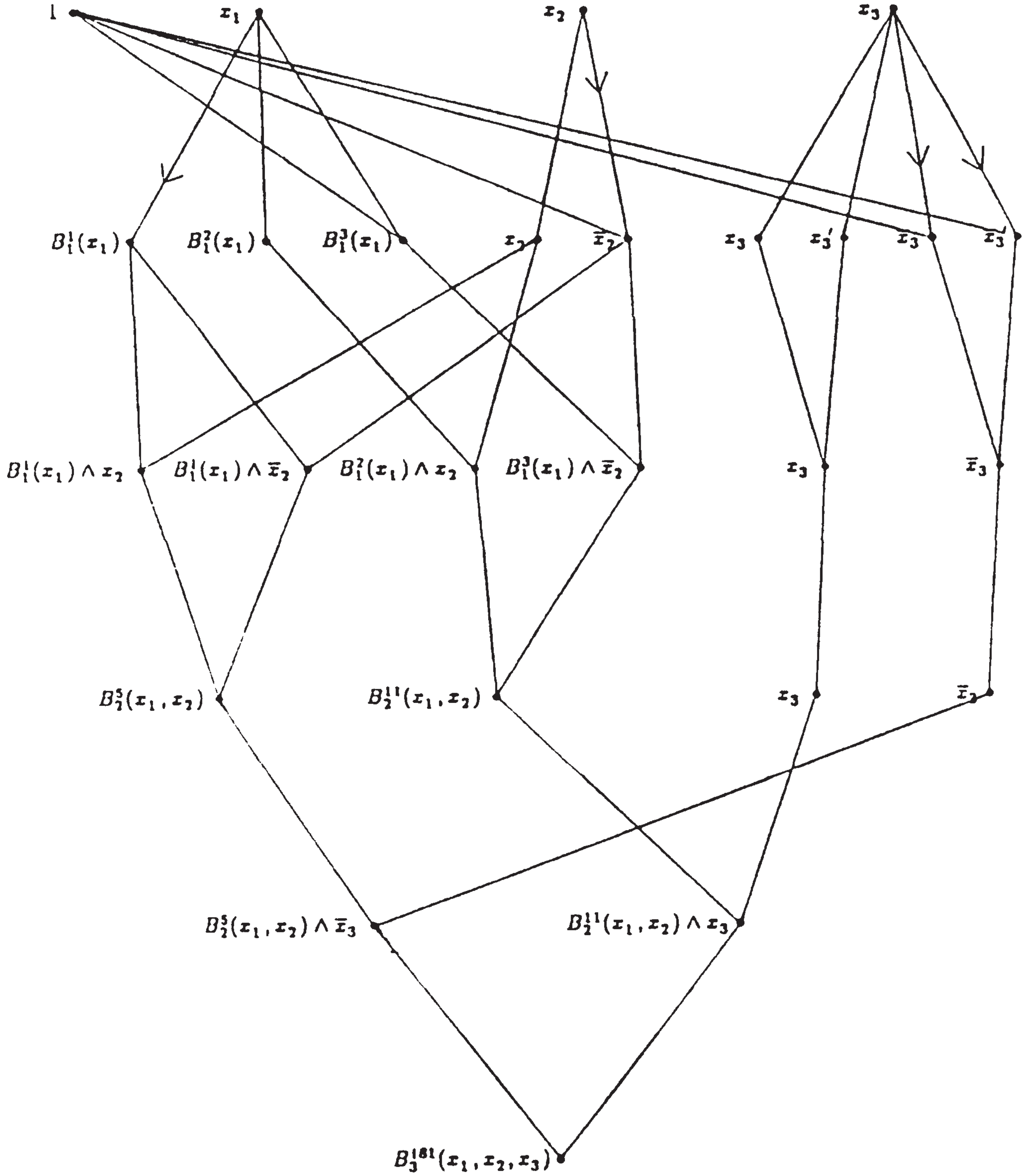
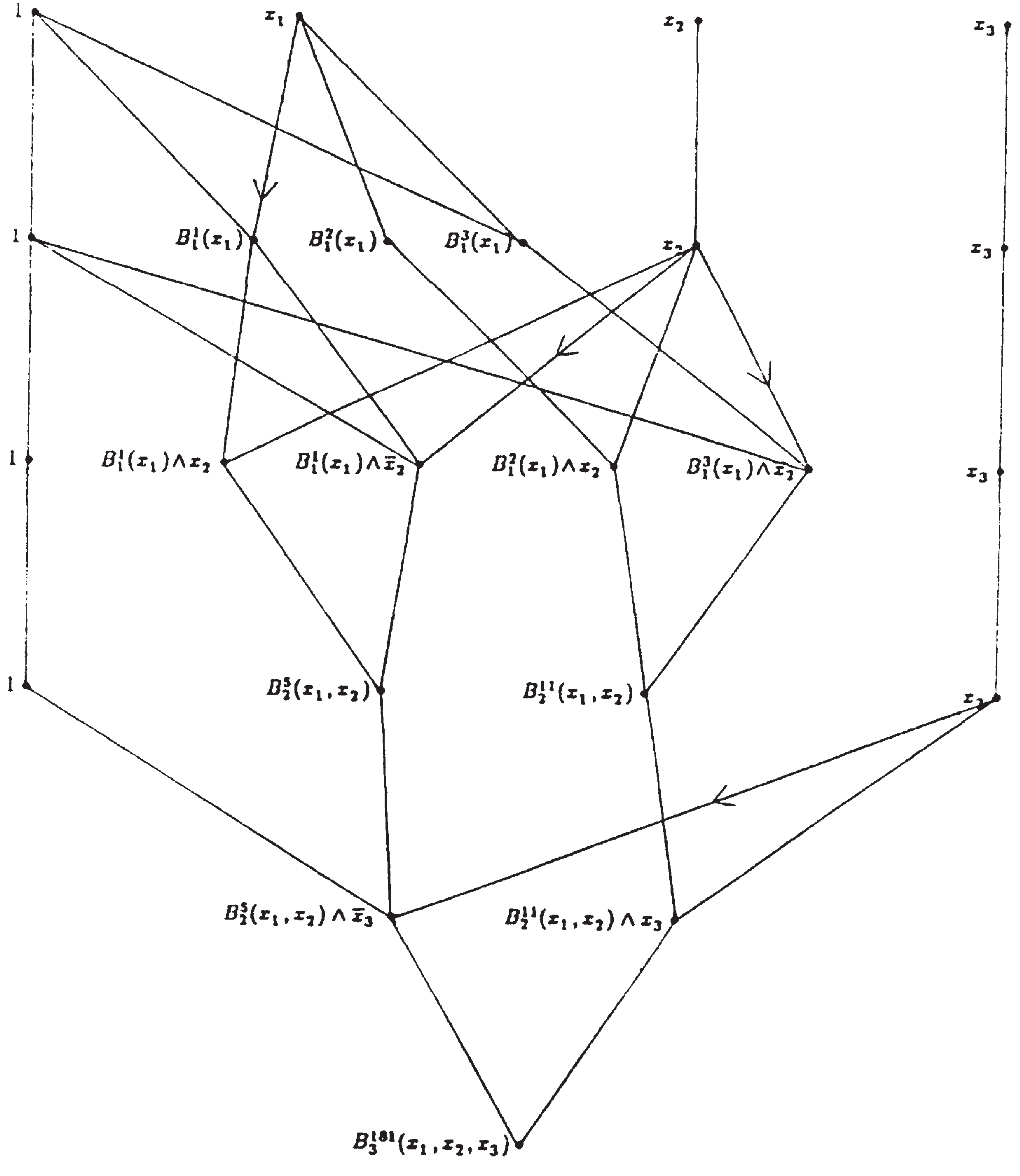

1
x_1
x_2
x_3
$B_1^1(x_1)$
$B_1^2(x_1)$
$B_1^3(x_1)$
x_2
$\bar{x}_2$
x_3
x_3'
$\bar{x}_3$
$\bar{x}_3'$
$B_1^1(x_1) \wedge x_2$
$B_1^1(x_1) \wedge \bar{x}_2$
$B_1^2(x_1) \wedge x_2$
$B_1^3(x_1) \wedge \bar{x}_2$
x_3
$\bar{x}_3$
$B_2^5(x_1, x_2)$
$B_2^{11}(x_1, x_2)$
x_3
$\bar{x}_3$
$B_2^5(x_1, x_2) \wedge \bar{x}_3$
$B_2^{11}(x_1, x_2) \wedge x_3$
$B_3^{181}(x_1, x_2, x_3)$

Figure 4b


1
x_1
x_2
x_3
1
$B_1^1(x_1)$
$B_1^2(x_1)$
$B_1^3(x_1)$
x_2
x_3
1
$B_1^1(x_1) \wedge x_2$
$B_1^1(x_1) \wedge \bar{x}_2$
$B_1^2(x_1) \wedge x_2$
$B_1^3(x_1) \wedge x_2$
x_3
1
$B_2^5(x_1, x_2)$
$B_2^{11}(x_1, x_2)$
x_3
$B_2^5(x_1, x_2) \wedge \bar{x}_3$
$B_2^{11}(x_1, x_2) \wedge x_3$
$B_3^{181}(x_1, x_2, x_3)$

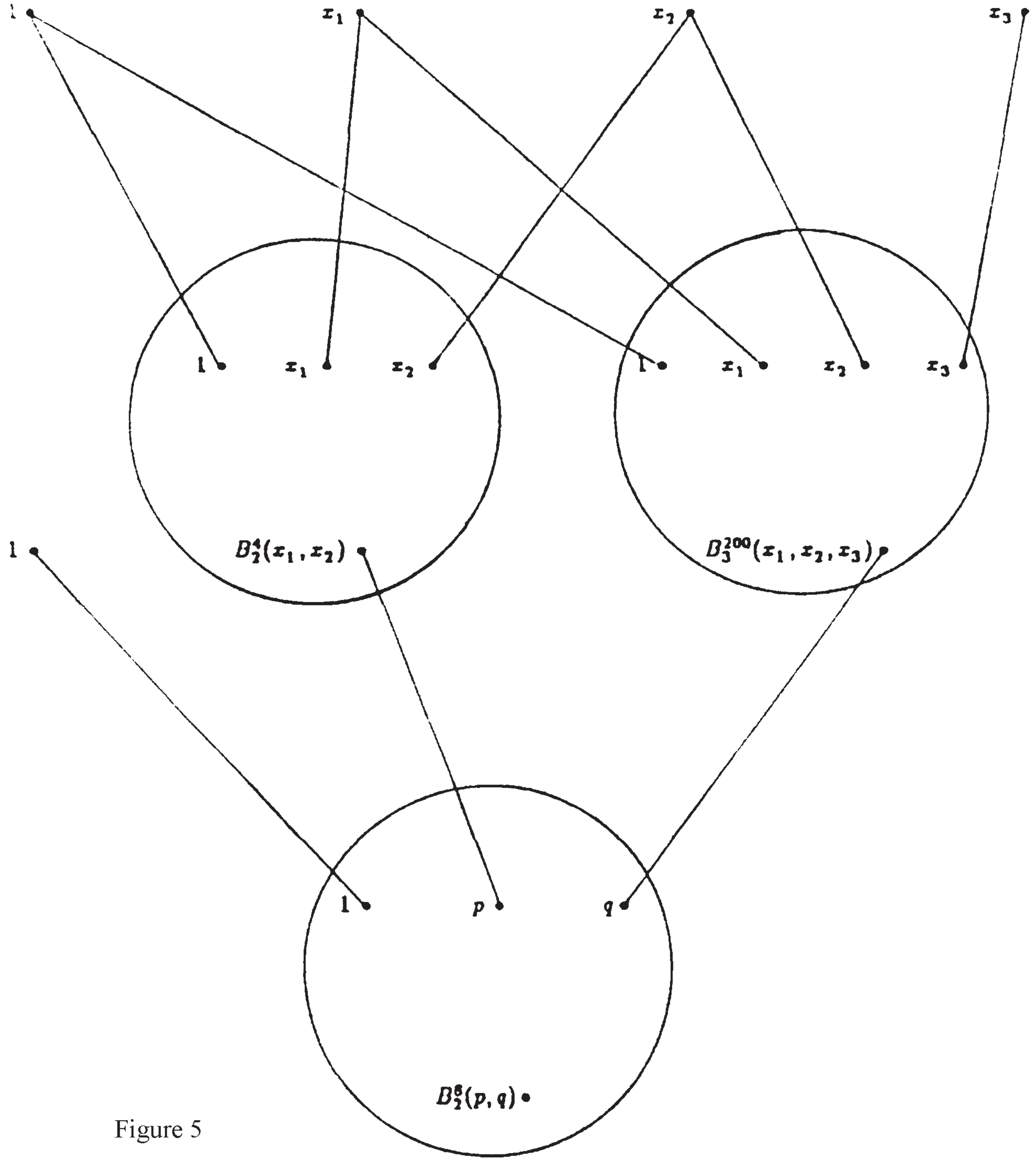


Figure 5