Flexible Reasoning of Boolean Constraints in Recurrent Neural Networks with Dual Representation

Wonil Chang[1], Hyun Ah Song[2], and Soo-Young Lee[3]

[1] Electronics and Telecommunications Research Institute, Daejeon 305-700, Republic of Korea
[2] Korea Institute of Science and Technology, Seoul, Republic of Korea
[3] Department of Electrical Engineering, KAIST, Daejeon 305-701, Republic of Korea
{chang.wonil,hi.hyunah}@gmail.com, sylee@kaist.ac.kr

Abstract. In this paper, we propose a recurrent neural network that can flexibly make inferences to satisfy given Boolean constraints. In our proposed network, each Boolean variable is represented in dual representation by a pair of neurons, which can handle four states of true, false, unknown, and contradiction. We successfully import Blake's classical Boolean reasoning algorithm to recurrent neural network with hidden neurons of Boolean product terms. For symmetric Boolean functions, we designed an extended model of Boolean reasoning which can drastically reduce the hardware cost. Since our network has only excitatory connections, it does not suffer from oscillation and we can freely combine multiple Boolean constraints.

Keywords: Boolean constraint, Boolean reasoning, symmetric Boolean function, recurrent neural network.

1 Introduction

Everyday we do logical thinking; we interpret given conditions, reason based on them, and draw a conclusion. We are also able to fill in missing variables given the rest, which is called " problem solving . " These complex yet flexible human reasoning shows properties that are quite different from ordinary computation processes in machines. Then how can we mimic these flexible reasoning processes?

There are a number of studies on how humans actually conduct reasoning processes. In [1], the authors carefully observe how humans interpret given conditionals and understand causal relations when reasoning tasks are given. In [6], the authors designed several different types of thinking processes based on decision theory by neural networks with adequate weights according to given conditions.

Logical reasoning can take place on different level of cognitive task, from evaluation of Boolean functions to natural language processing. This paper focuses on biological computation model of Boolean reasoning. Majority of biological reasoning models are based on artificial neural networks (ANN). Initial reasoning models are restricted to compute logical functions in feed-forward network structure [8, 14, 15]. Advanced models use recurrent neural networks in Boolean reasoning so that it can cover both backward and forward boolean inference [11, 16]. For example, these networks not only compute AND operation of two binary input, but they also infer that given true output, all input must be true accordingly. The boolean reasoning systems tries to find a set of unknown binary variables that satisfy a number of given Boolean formula. This is a well-known satisfiability (SAT) problem [7] in computer science.

Many reasoning models adopt three-state logic to deal with unknown variables [4,11,13,16]. The three states are commonly encoded as positive (true), negative (false), and zero (unknown) activation of a single neuron. If we assign proper excitatory and inhibitory connectivity between neurons, it is possible to infer unknown values other neurons.

However, conventional recurrent neural networks are problematic in combining multiple boolean constraints into the unified system. The accumulation of synaptic weights from different constraints might cause wrong inference or oscillation of network. Therefore, most successful ANN-based inference models [11,16] are implemented to support only limited type of boolean operations such as AND, OR, and negation. And networks can not be fully parallelized due to undesirable interferences.

In this paper, we propose a flexible Boolean reasoning network that can include any kind of boolean constraints without restriction. Our network is able to handle four states(true, false, unknown, and contradiction) by dual representation of Boolean variables. The encoding of a boolean variable using a pair of neurons is not a new idea [12,14,15]. However, to our best knowledge, this paper presents the first boolean network model that can reason on all possible ways using dual representation. Since the network has only excitatory synapses, it is free from interference problem and oscillation.

The organization of paper is as follows. In Section 2, we explain our design of dual representation of variables. The structure of network for general Boolean constraints based on full extraction of prime implicants, or Blake Canonical Form (BCF) [3] is explained in Section 3. In Section 4, we introduce a simplified implementation of the network for symmetric Boolean constraints. We close our paper in Section 5.

2 Dual Representation

Our network adopts dual representation, where a Boolean variable is encoded by two neurons, each standing for 'true' and 'false,' respectively. A pair of neurons represent four possible states. If one of two neurons is active, it means that the variable is either true or false; If both of true and false neurons are active, it means given conditionals are in contradicting situation that we cannot satisfy the constraint; If both of the neurons are silent, it means not enough information is given and we cannot draw a conclusion from it. The description of four possible state representations are shown in Fig. 1.

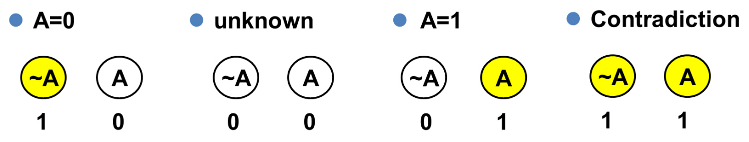

Fig. 1. Dual representation of a Boolean variable in four possible states

In [11] and [16], boolean reasoning is performed on the basis of delicately-designed excitatory and inhibitory connections. When a neuron is involved in multiple constraints, the interference between neural interactions might lead to wrong conclusion. Dual representation has an important advantage over conventional single-neuron representation in that matter; we do not need to inhibit any neuron to inactivate. Instead, its counterpart is excited. Since our network has excitatory connections only, it does not suffer from interference problem. Also, it always converges fast without oscillation.

3 Reasoning Network for General Boolean Constraints

The neural connectivity is described as 'A conjunction of neurons → A neuron.' For example, consider a function $C = A$ AND B. Regardless of directions that distinguish inputs from outputs, we can derive multi-directional inference rules such as:

- If $C = 1$, $A = 1$ ($C \rightarrow A$)
- If $A = 1$ and $B = 1$, $C = 1$. ($AB \rightarrow C$) $\cdots$

Our goal is to make inferences based on the set of variables that satisfies a Boolean constraint

$$f(x_1, x_2, ..., x_N) = 0 \tag{1}$$

and then to make inferences based on the true configuration information.

In the example, its Boolean constraint is re-formulated as

$$f(A, B, C) = (A \text{ AND } B) \text{ XOR } C = 0. \tag{2}$$

Any kind of Boolean function can be transformed into Boolean constraint. We can make a table of configuration of variables and the constraint formula f:

A	B	C	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

The optimal structure of the the network is derived from Blake's canonical expressions [2], that uses prime implicants to represent the Boolean formula f in abstract form. An implicant of a Boolean formula f is a product term P_i such that $P_i \leq f$. A prime implicant of f is an implicant that cannot be included by a more general implicant. In (2), term $\bar{A}BC$ is an implicant of $f(A,B,C)$ but not a prime implicant since it is covered by another implicant $\bar{A}C$. Blake's canonical form (BCF), or complete sum of f is the sum of all prime implicants F_i in f.

$$BCF(f) = \sum_i F_i = 0 \tag{3}$$

Variable configuration set that satisfies f can be simply solved by checking BCF. If Boolean variables in the given conditions satisfy the Boolean formula $f = 0$, then $F_i = 0$ for all i. We can connect activation synapes to a varible x_i from other variables by

$$F_i/x_i \rightarrow \bar{x}_i, \tag{4}$$

where F_i/x_i is a quotient of F_i with respect to x_i. (For example, $\bar{A}B/B = \bar{A}$.) An example of the whole reasoning network for AND operation is shown in Fig. 2.

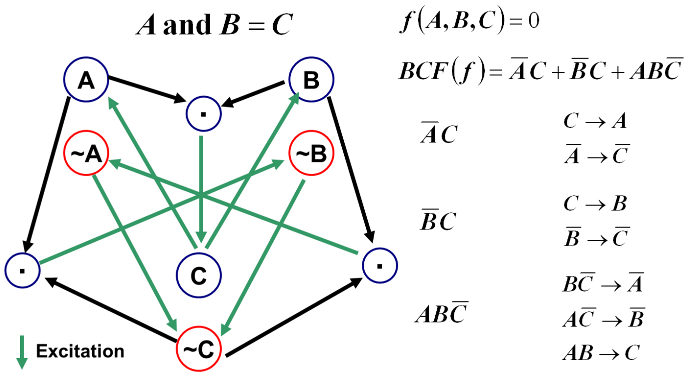
A and $B = C$
A
B
~A
~B
C
~C
Excitation
$f(A,B,C) = 0$
$BCF(f) = \bar{A}C + \bar{B}C + AB\bar{C}$
$\bar{A}C$
$C \rightarrow A$
$\bar{A} \rightarrow \bar{C}$
$\bar{B}C$
$C \rightarrow B$
$\bar{B} \rightarrow \bar{C}$
$AB\bar{C}$
$B\bar{C} \rightarrow \bar{A}$
$A\bar{C} \rightarrow \bar{B}$
$AB \rightarrow C$

Fig. 2. Structure of network connection of AND operation

The product of boolean variables $(x_1, x_2, \ldots, x_N)$ is implemented by a single perceptron

$$h = \varphi\left(\sum_{i=1}^{N} w_i x_i - \theta\right) \tag{5}$$

with uniform pre-synaptic weight $w_i = 1$ for all i, threshold $\theta = N$, and the activation function

$$\varphi(\alpha) = \begin{cases} 1 & \text{if } \alpha \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

For Boolean equations with four variables or less, we can easily find prime implicants using Karnaugh-Map [5]. Extraction of prime implicants in general cases has been extensively studied for digital circuit optimization after the foundational work of Quinn and McCluskey [9]. However, it is known that the time complexity of full search for arbitrary boolean function is exponential with respect to the number of input variable [17]. And the number of prime implicants for a boolean function with N inputs is upper-bounded by $O(2^N)$ [10].

4 Optimized Reasoning Network for Symmetric Boolean Function

Symmetric Boolean function is a boolean function $f(x_1, x_2, \ldots, x_N)$ that satisfies

$$f(x_1, x_2, \ldots, x_N) = f\left(\pi(x_1, x_2, \ldots, x_N)\right) \tag{7}$$

for an arbitrary permutation π.

Symmetric boolean functions are expressed as functions in terms of total sum of input variables:

$$f(x_1, x_2, \ldots, x_N) = g\left(\sum_i x_i\right). \tag{8}$$

Note that $\sum_i x_i$ is the arithmetic sum (not Boolean sum) in this equation. And g is an arbitrary function.

By using summation of the variables as the unit to check truth table for a function, the time complexity of network design is reduced to polynomial order. We construct and search truth table of summation of variables, not value of variables themselves.

For example, take example of OR operation with three input $y_1 = \mathrm{OR}(x_1, x_2, x_3)$. The truth table based on the summation of variables can be constructed as shown below.

$\sum_i x_i$ \ $\sum_i y_i$	0	1
0	T	F
1	F	T
2	F	T
3	F	T

We can reformulate the original symmetric Boolean function in terms of alternative type of prime implicants $(a \le \sum_i x_i \le b) \cdot (c \le \sum_i y_i \le d)$, which we call "inequality prime implicants."

According to the truth table, $f(x_1, x_2, x_3, y_1) = g\left(\sum_i x_i, \sum_i y_i\right) = (0 \le \sum_i x_i \le 0) \cdot (1 \le \sum_i y_i \le 1) + (1 \le \sum_i x_i \le 3) \cdot (0 \le \sum_i y_i \le 0) = 0$.

Like (4), we can search for the connection between variables. Generalized rules is defined as follows:

If $(a \le \sum_i x_i \le b) \cdot (c \le \sum_i y_i \le d)$ is given,

$$\left.\begin{array}{l}(\sum_i y_i \ge c)\cdot\\(\sum_i y_i \le d)\cdot\\(\sum_{j\neq i} x_j \ge a-1)\cdot\\(\sum_{j\neq i} \bar{x_j} \ge N-b)\end{array}\right\} \to \bar{x_i} \quad \left.\begin{array}{l}(\sum_i y_i \ge c)\cdot\\(\sum_i y_i \le d)\cdot\\(\sum_{j\neq i} x_j \ge a)\cdot\\(\sum_{j\neq i} \bar{x_j} \ge N-b-1)\end{array}\right\} \to x_i$$

$$\left.\begin{array}{l}(\sum_i x_i \ge a)\cdot\\(\sum_i x_i \le b)\cdot\\(\sum_{j\neq i} y_j \ge c-1)\cdot\\(\sum_{j\neq i} \bar{y_j} \ge N-d)\end{array}\right\} \to \bar{y_i} \quad \left.\begin{array}{l}(\sum_i x_i \ge a)\cdot\\(\sum_i x_i \le b)\cdot\\(\sum_{j\neq i} y_j \ge c)\cdot\\(\sum_{j\neq i} \bar{y_j} \ge N-d-1)\end{array}\right\} \to y_i. \qquad (9)$$

Inequality terms and their products in (9) are implemented in the same way as in (5). The number of inequality prime implicants for N-input symmetric Boolean functions is upper-bounded by $O(N)$. Compared to the previous model in section 3, the computational cost is significantly reduced.

5 Concluding Remarks

We proposed a flexible neural network that successfully handles Boolean constraint problems in multi-directions. Dual representation enables simple implementation of delicate reasoning rules. We described the learning process of the network using conventional Blake's reasoning algorithm. We also proposed an alternative type of prime implicant for abstract representation of symmetric Boolean constraints, which significantly reduced computational cost. Since the network use only excitatory synaptic connections, we guarantee stability of the process even for complex functions consisting of heavy constraints. We expect that our proposed network to be applied to solve complex and multi-directional Boolean constraint problems free of computational burden in more flexible manner.

Acknowledgments. This research was funded by the MSIP(Ministry of Science, ICT & Future Planning), Korea in the ICT R&D Program 2013 (Wonil Chang). Also, it was supported by Public Welfare & Safety Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2012-0006551, Hyun Ah Song).

References

1. Beller, S., Kuhnmünch, G.: What causal conditional reasoning tells us about people's understanding of causality. Thinking & Reasoning 13(4), 426–460 (2007)
2. Blake, A.: Canonical expressions in Boolean algebra. University of Chicago (1938)
3. Brown, F.M.: Boolean reasoning: the logic of Boolean equations. Courier Dover Publications (2003)

4. Dietz, E.-A., Hölldobler, S., Ragni, M.: A computational logic approach to the suppression task. In: Proceedings of the 34th Annual Conference of the Cognitive Science Society, pp. 1500–1505 (2012)
5. Karnaugh, M.: The map method for synthesis of combinational logic circuits. Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics 72(5), 593–599 (1953)
6. Kowalski, R.A.: The logical way to be artificially intelligent. In: Toni, F., Torroni, P. (eds.) CLIMA 2005. LNCS (LNAI), vol. 3900, pp. 1–22. Springer, Heidelberg (2006)
7. Malik, S., Zhang, L.: Boolean satisfiability from theoretical hardness to practical success. Communications of the ACM 52(8), 76–82 (2009)
8. Mandziuk, J., Macukow, B.: A neural network performing boolean logic operations. Optical Memory and Neural Networks 2(1), 17–35 (1993)
9. Mccluskey, E.J.: Minimization of Boolean functions. The Bell System Technical Journal 35(5), 1417–1444 (1956)
10. McMullen, C., Shearer, J.: Prime implicants, minimum covers, and the complexity of logic simplification. IEEE Transactions on Computers 100(8), 761–762 (1986)
11. Spears, W.M.: A nn algorithm for boolean satisfiability problems. In: IEEE International Conference on Neural Networks, vol. 2, pp. 1121–1126. IEEE (1996)
12. Stenning, K., Lambalgen, M.: Semantic interpretation as computation in nonmonotonic logic: The real meaning of the suppression task. Cognitive Science 29(6), 919–960 (2005)
13. Stenning, K., Van Lambalgen, M.: Human reasoning and cognitive science. The MIT Press (2008)
14. Tan, C.L., Quah, T.S., Teh, H.H.: An artificial neural network that models human decision making. Computer 29(3), 64–70 (1996)
15. Teh, H.H.: Neural Logic Networks: A New Class of Neural Networks. World Scientific (1995)
16. Wang, G., Shi, H.: Tmlnn: triple-valued or multiple-valued logic neural network. IEEE Transactions on Neural Networks 9(6), 1099–1117 (1998)
17. Wegener, I.: The complexity of boolean functions (1987)