

- 308 s.
17. Franko, IYa 1976. Zibrannya tvoriv : u 50 t (Collection of works: in 50 volumes). AN URSSR, In-t lit. im. T.G. Shevchenka, Kiyiv: Naukova dumka, T. 1. s. 65-66.

DOI 10.33930/ed.2019.5007.32(4)-2

УДК 164

FROM BOOLE'S LOGIC TO BOOLEAN APPLICATIONS IN COMPUTER SCIENCE

ВІД БУЛЕВОЇ ЛОГІКИ ДО ЇЇ ЗАСТОСУВАННЯ У КОМП'ЮТЕРНИХ НАУКАХ

Mohamad Awwad

Urgency of the research. We consider crucial to distinguish between the original Boole's logic and its modern development as Boolean logic in order to understand their theoretical and applied capacities. In addition, the development of Boolean logic constitutes a strong example about the relationships between logic, an algebraic language of thoughts, and computation.

Target setting. Boole's logic is generally perceived as equivalent to Boolean logic despite their mathematical and conceptual differences. The influences of Boolean logic on computer science are recognized especially in their applications in circuits design. However, it appears that the influence of this logic on Cryptography is less cited in the literature when studying the relationships between logic and computer science.

Actual scientific researches and issues analysis. Studies of cryptographic Boolean functions and applications have been realized by scholars such as T. Cusick, and P. Stanica. Boolean methods in Mathematics are studied by Y. Crama and P. L. Hammer. Some differences between the original Boole's logic and the modern Boolean logic have been pointed out and promoted by N. J. Wildberger. Boole's algebra as a mathematical language of

Актуальність теми дослідження. Ми вважаємо вирішальним розмежувати оригінальну Булеву логіку та її сучасний розвиток з метою розкриття її теоретичних та прикладних можливостей. Розвиток Булевої логіки є цінним прикладом взаємозв'язків між логікою, алгебраїчною мовою та обчисленнями реальних міркувань.

Постановка проблеми. Булева логіка, як правило, сприймається як рівноцінна Булевим логікам, незважаючи на їх математичні та концептуальні відмінності. Вплив Булевої логіки на інформатику визнаний особливо в їх застосуванні при проектуванні схем. Однак, схоже, вплив цієї логіки на криптографію менше згадується в літературі при вивченні взаємозв'язків між логікою та інформатикою.

Аналіз останніх досліджень і публікацій. Дослідження криптографічних Булевих функцій та додатків проводили такі вчені, як Т. Кусік та П. Станіка. Булеві методи в математиці вивчають Ю. Крама та П. Л. Хаммер. На деякі відмінності між початковою логікою Буля та сучасною булевою логікою вказував і пропагував Н. Дж. Вайлдбергер. Булева алгебра як математична мова думки була представлена в багатьох

thought has been presented in many fundamental handbooks such as those written or edited by D. M. Gabbay, J. Woods, W. Kneale, M. Kneale, and L. Haaparanta.

The research objective. The first objective of this research is to formally explain the theoretical differences between Boole's logic and the modern Boolean logic. The second objective is to show the immense role of Boolean logic in computer science through the developments of circuits design and cryptography.

The statement of basic material. The developments of an algebraic logical language of thoughts by G. Boole are considered using historical and theoretical perspectives. The technical implementations of Boolean logic in combinational circuits and in modern cryptography show strong influences of a 19th century logic on the latest technologies of computing.

Conclusions. In this research we came to the following results: 1) Boolean logic is different but derived from the original logic of Boole; it has been obtained due to a transformational process that led to a two-valued logic. 2) This two-valued logic made possible its digital implementation into a binary system. 3) The use of Boolean logic in circuits' analysis and design constitutes a strong example of the influences of logic on computer science. 4) The use of Boolean logic in cryptography constitutes another strong example of the role of logic in modern computation.

Keywords: Logic, Computer Science, Boole's Algebra of logic, Boolean Logic, Circuit analysis and Design, Cryptography, Error Correcting Code.

фундаментальних довідниках, таких як написані або відредаговані Д. М. Габбей, Дж. Вудс, В. Кніл, М. Кніл та Л. Хаапаранта.

Постановка завдання. Головною метою цього дослідження є пояснення теоретичних відмінностей між Булевою логікою та сучасними Булевими логіками. Додатковою метою дослідження є демонстрація ролі Булевої логіки в інформатиці через розробки схеми проектування та криптографії.

Виклад основного матеріалу. Розробки алгебраїчної логічної мови міркувань Буля розглядаються з використанням історичних та теоретичних перспектив. Технічні реалізації Булевої логіки в комбінаційних схемах і в сучасній криптографії демонструють сильний вплив логіки 19 століття на новітні технології обчислень.

Висновки. У цьому дослідженні ми дійшли до таких результатів: 1) Булеві логіки відрізняються, але походять від вихідної логіки Буля; це було отримано завдяки трансформаційному процесу, який призвів до двозначної логіки. 2) Ця двозначна логіка зробила можливим її цифрову реалізацію у двійковій системі. 3) Використання логічної логіки в аналізі та проектуванні схем є вагомим прикладом впливу логіки на інформатику. 4) Використання Булевої логіки в криптографії є ще одним вагомим прикладом ролі логіки в сучасних обчисленнях.

Ключові слова: логіка, комп'ютерні науки, Булева логічна алгебра, Булева логіка, аналіз і розробка мікросхем, криптографія, код виправлення помилок.

Urgency of the research. This study is relevant in many respects. Firstly, it unveils the incorrect, sometimes assumed, equivalence between the original Boole's logic (or Boole's algebra) and the modern Boolean logic (or Boolean algebra) by precisely providing the differences between them. Secondly, it emphasizes the crucial role of Boolean logic in Computer science as 1) an algebraic language used to make logical deductions and proofs, and consequently increasing the amount of knowledge within the logical system 2) an engineering tool in order to design digital circuits existing in a wide range of modern technologies 3)

theoretical models and methods to be applied in modern Cryptography and Error-correcting code. In addition, this study is relevant regarding the question of the nature of Computer science as basically a logical development along with other scientific and physical nature.

Target setting. The investigation of the role of Boole's logic in Computer science is a part of a more general problem of studying the influences of logic on Computer science and other fields of knowledge. Understanding the role of logic in Computer science is directly related to grasp the foundational developments of this discipline as well as its nature in the philosophical sense of the term. In this specific study, it is interesting to observe that Boolean logic is connected to Computer science as a Mathematical language able to algebraically perform logical calculations, but also well adapted to be used as a framework to implement hardware computer components. In a wider perspective, the understanding of the inherent connections between Logic, Computer science, languages of thoughts, and digital implementations constitute the nuts and bolts of the problem.

Actual scientific researches and issues analysis. The study of the applications of Boolean functions in circuits design became classic since the work of C. Shannon to more recent developments in the VLSI domains. However, a great number of new discoveries in cryptography has been achieving since the 1970s. For example, a modern comprehensive reference on cryptographic Boolean functions has been published by T. Cusick and P. Stanica [1]. This work is foundational-oriented and presents the major use of Boolean methods and functions in modern cryptography. Moreover, the reference Boolean Models and Methods in Mathematics, Computer Science, and Engineering (edited by Y. Crama and P. L. Hammer) [2] is very relevant in the investigation of the impact of Boolean functions on these disciplines. It is an extensive work on Boolean functions covering, among other topics, their algebraic structures, learning theory and cryptography, graph representations and neural networks, and engineering applications.

The research objective. Our main objective in this paper is to investigate the importance of Boolean logic in Computer Science. We are interested in understanding the impacts of Boolean logic on the design and analysis of the combinational logic circuits as well as on Cryptography.

To better support this investigation, our first task is the study of the foundational concepts of Boole's algebra of Logic. We consider this task crucial in order to extract the main substance constituting the modern Boolean Logic by finding its differences comparing to the original Boole's logic.

In addition, we seek justifying how Cryptography and Error Correcting Code theory, sub-areas of computer science, have largely benefited from the development of Boolean Logic.

The statement of basic material.

From a language of thoughts to combinational circuits.

Originally, G. Boole used algebraic symbols on classes (denoted by x, y, z, \dots) that could be related by three main operations: the negation, the union, and the intersection. Given a class denoted by x , its negation, \bar{x} , represents all elements that are not members of the class x . The union of two classes x and y and expressed as $x + y$, is the class containing all elements in the class x or the class y . Similarly, the intersection of the classes x and y , denoted by xy , is the class containing the elements belonging to the class x and the class y . In addition, Boole used the sign $-$ between two classes x and y as $x - y$ to represent the members of

x excluding the members of y . Notice that this operation was defined only when the class y is a part of the class x . On the other hand, the sign '=' is fundamental in Boole's system. In fact, it is used to replace the symbol *different* (represented by '! =' or ' ≠ ') by the symbol '='. This substitution was possible by adding another concept we discuss below in the text [3–5].

In this system, we need to define two special classes 1) the universe or the 'universe of discourse' to which the variables of interest belong to (i. e. humans, cars, numbers) 2) the empty class (or null) to which nothing belongs to. In his algebra, Boole used the symbols 1 and 0 to represent, respectively these two classes. Apparently, the use of 1 and 0 for these two classes was influenced by the laws of probability. However, it is easy to understand that given an element x , the fact that x belongs to the universe is always true (or 1), and the fact that x belongs to the empty or the null class is always false (or 0).

Notice that the intersection of an element x with the universe is naturally x , and the intersection between an element x with the null class is null. These two facts give respectively the equations $1x = x$ and $0x = 0$. The two mentioned classes, the universe and the null, are often considered a novelty in the symbolic algebra of Boole. There is a natural analogy between the intersection of classes and the product of numbers. In this context, Boole has defined a simple but crucial law over the intersection and had important consequences in his Algebra of Logic. It can be said that the intersection of the class denoted by x with itself is exactly x , and the corresponding equation is $xx = x$. This idea leads to the equation $x^n = x$, called the idempotent law, meaning that the intersection of x with itself repeated n times is always x and no new elements can be added to x . Note that the division operation is meaningless in the Algebra of Boole for many reasons. For example, such hypothetic division in the last equation by x would make the equation $x^{n-1} = 1$ meaning the intersection of x with itself repeated $n - 1$ times are equivalent to the universe. Many other algebraic reasons make also the division nonsense [3].

Additionally, there is an important restriction that Boole established regarding the union of two classes. It is not allowed to talk about the union of two classes except when they are *mutually exclusive*. For this reason, and in order to guarantee this condition, Boole often wrote $x + \bar{x}y$ instead of $x + y$ to exclude all potential members belonging simultaneously to the classes denoted by x and y . This restriction constitutes one of the fundamental differences between the original Algebra of Boole and what we call today the Boolean algebra. Other differences between these two terms will be discussed in the text.

As we have mentioned, Boole introduced the sign $-$ (minus) as the inverse of the addition sign. In this context, he originally used $1 - x$ as the complement of x (the elements in the universe without the element of x) before using the sign \bar{x} . The precedent use of the sign $-$ permits to Boole to introduce the law $x(1 - x) = 0$ meaning the intersection of a class with its complement is null.

The introduced symbolic notations can represent the main four propositions of the traditional logic as follows [3]:

- Every x is y : $x(1 - y) = 0$.
- No x is y : $xy = 0$.
- Some x is y : $xy! = 0$.
- Some x is not y : $x(1 - y)! = 0$.

In this model, the inequality sign present in the last two propositions is avoided by introducing the famous symbol v representing a meaning of "some". In fact,

when we say “some x is y ”, it means that some members of the class x are also members of the class y . Though, these members constitute another class denoted by v and having at least one member. The last two propositions become [3]:

- Some x is y : $xy = v$.
- Some x is not y : $x(1 - y) = v$.

Based on the preceding definitions, the following laws constitute the main premises of the Algebra of Boole that largely inspired from the classical algebra [3–5]:

- $xy = yx$
- $x + y = y + x$
- $x(y + z) = xy + xz$
- $x(y - z) = xy - xz$
- $(x = y) \rightarrow (xz = yz)$
- $(x = y) \rightarrow ((x + z) = (y + z))$
- $(x = y) \rightarrow ((x - z) = (y - z))$
- $x(1 - x) = 0$

Notice again that the last premise works algebraically only if x is exclusively 0 or 1. This fact does not mean that the original Boole’s system is a 0-1 system. Boole’s system is a more general algebra of classes where the combinations of operations on these classes can give “values” or sub-classes different to the null and the universe classes (0 and 1 respectively). However, his original system can be turned to become a 0-1 system by adding the restriction “*Either $x = 0$ or $x = 1$* ”. This restriction, among others, was fundamental in transforming the original algebra of Boole into a Boolean algebra.

In any case, what really matters in this development is to establish an Algebra of Logic using the closest set of axioms and premises to those of traditional algebra (or numerical algebra) in order to perform the needed calculations in a similar way. However, what would really distinguish them are the interpretations that can be stated and not the used form or the *language* in order to apply these operations. The main objective of his system is to generate abstractly conclusions or consequences independently of the interpretation or the meaning that can be established at the end. In line with this idea, this mechanical process is called a *development* which can be analogue to the development followed upon solving a linear equation or a functional expression. We often consider this Boolean mechanical approach as the basis of the called *decision procedure* [3].

Now, it is time to formally distinguish between the original Boole’s logic and the modern Boolean logic. Boolean logic, a binary truth-functional logic, could be roughly considered as a special system of Boole’s algebra of logic presented above with the following *three main differences* that should be taken into consideration:

1. All variables in the Boolean logic (called classes in the Algebra of logic) are either 0 or 1. As consequence of this assumption, the value v denoting the idea of “some” in the Algebra of logic does simply not exist in the Boolean logic.
2. The addition of two variables (called union of classes in the Algebra of logic) is not mutually exclusive in the Boolean algebra; this means that $1 + 1 = 1$ (in opposite to $1 + 1 = 0$ when the addition is exclusive in the Algebra of logic).

3. The negative sign ($-$) is not used in the Boolean logic as well as the symbol *different* (\neq).

As we have already seen, the symbol (\neq) was earlier avoided by Boole by adding the value v ; in Boolean logic, these two symbols have absolutely no place. Because any Boolean variable is either 0 or 1, the value v is not necessary, and for the same reason if a variable x is not 0, it is necessarily 1 and vice versa; consequently the symbol (\neq) is totally useless.

Another remark that we should state here is the natural corresponding between the operators *complement*, *addition*, and *multiplication* used in Boole's Algebra of logic and their respective equivalence with the *negation*, *disjunction*, and *conjunction* as exactly the Boolean operators NOT, OR, and AND (often denoted by \neg , \vee , and \wedge).

One of the most useful domains of applications of Boolean logic is the circuit analysis and design. In 1938, C. Shannon proved that Boolean logic can be used in electronic circuits to perform binary calculations [6]. This idea became and is still fundamental in designing any modern computing device. A Boolean function (any algebraic combination of Boolean variables under the operators (\neg , \vee , and \wedge), can be represented as truth table expressing its characteristics. Based on these three principal Boolean operators (NOT, OR, AND), it is possible to construct any logical function, and consequently its corresponding circuit.

In more detail, the main Boolean operations are designed in a physical manner using transistors. A transistor is simply a tiny switching device used as elementary unit in the construction of more complex electronic circuits. It is important to mention that a transistor works on voltages: high and low, which can easily represent respectively the two Boolean logical values true and false (1 and 0). In order to construct a NOT gate, the following simplified procedure on the transistor is performed: if an input 1 as a high voltage is operated, the output will be 0 due to a direct connection with the ground that represent a low voltage. If the input value is 0, the value 1 is passed due to its connection to the power through a resistor. In both cases the negation of the input is received as output and the whole mechanism represents the gate NOT. Generally, this basic idea of implementing a NOT gate using transistors can be expanded to build the AND gate and the OR gate. Given that the three main operators are now designed as building blocks gates, any combinational logic circuit can be implemented using the corresponding three gates. A combinational circuit is a set of connected gates receiving 0-1 inputs and producing a 0-1 output as the result of a Boolean function. Any connection between gates or between the input/output and the gates is done by wires [7].

From the general implementation point of view, the designers have built different useful gates of extended Boolean operations based on the three elementary gates. For example, the XOR gate for the exclusive OR, the NAND gate for the negation of the AND, and the NOR gate for the negation of the OR.

Boolean function Cryptography and Error Correcting Code.

The main objective of Cryptography, as a cross-discipline of Computer Science and Mathematics, is to manage a digital communication between entities in a secure manner. It tries to prevent the recovering of the original message (plaintext) and not the coded message sent over the network (ciphertext). The encryption (enciphering) method is the conversion of the original text to an encrypted text using a private key (conventional cryptography) or a public key (public cryptography). The decryption (deciphering) is the recovering of the encrypted text to the original plaintext using a secret key. Error correcting codes is

the procedure of detecting the errors produced upon a communication over a noisy network in order to find and correct these errors at the reception of the message [2].

The Boolean functions play an important role in cryptography and error correcting code. Evidently, the original messages, the keys, and any data transformation during the process of encryption/decryption are represented in binary. Thus, the cryptographic methods treat these data purely as Boolean functions [1]. For instance, a composition of non-linear Boolean functions is used in cryptographic transformations of both stream and block ciphers (two different encryption methods). In fact, the S-boxes (for Substitution-box), parts of substitution algorithms of block ciphers, are simply vectorial Boolean functions. In addition, Boolean operations such as the exclusive OR (XOR) are largely applied in many encryption algorithms. Moreover, several properties of Boolean functions are exploited in cryptography to make its methods more effective. For example, the Correlation Immunity, the resilience, and the Nonlinearity using the distance of the Boolean functions' truth tables are widely considered. Finally, the confusion and the diffusion, important principles in cryptography, are intimately related to the degree of complexity of the used Boolean functions. In fact, it is possible to quantify the degree of resistance in a cryptosystem to the known attacks using fundamental characteristics of the adopted Boolean functions. These characteristics can be directly associated to the confusion and diffusion principles [2].

Conclusions of this research and prospects for further exploration. We have shown in this paper how an Algebra of Logic created by Boole in the middle of 19th century turned out to be of immense importance in Computer Science one century later. We explained in this research the transformation of Boole's original logic into a modern logic able to be used as a platform to build combinational circuits. The same modern Boolean developments allowed generating highly effective cryptographic functions and models to assure secure digital communication. In general, we tried to prove that Boolean logic has had great influences in the applied aspects of Computer science as well as in its theoretical and mathematical foundational aspects. More precisely, we have investigated in some details how it is possible to use Boolean logic in the analysis and design of electronic hardware components widely implemented in modern technologies. This hardware development has been extended in the 1970s to build VLSI circuits algebraically represented by Boolean functions.

Boole's main objective was to perform an algebraic logical mechanism on thoughts similarly to the mechanism used in classic algebra. We emphasized how this algebraic reasoning that operates on signs can be classified depending on their functions in order to make logical calculations. Indeed, we know today that any propositional calculus expression can be written in Boolean algebra.

There are many others important repercussions of Boolean algebra on Computer science as well as on other area of knowledge that we did not study in this paper. Perhaps every role in this context deserves a particular exploration. For example, the theoretical aspects of Boolean algebra improved and extended by Jevons, Peirce, and Schroder were particularly important in the creation of predicate logic and, generally, in mathematical logic developments. These transformational processes leading from one kind of logic to another more powerful deserve further investigation. In addition, we consider relevant to explore the essential role of Boolean algebra in statistics and set theory. In fact, the

Algebra of sets is a Boolean algebra and, due to Stone's proof, every Boolean algebra is isomorphic to a field of set.

Finally, it is useful to notice that the domain of Boolean circuit connects the time complexity to circuit complexity, and the Boolean Satisfiability Problem (SAT) is always considered a reference problem in theoretical Computer science of being the first discovered NP-problem.

References:

1. Cusick, TW & Stanica, P 2017. Cryptographic Boolean functions and applications, *Academic Press*.
2. Crama, Y & Hammer, PL 2010. Boolean models and methods in mathematics, computer science, and engineering, *Cambridge University Press*, Vol. 2.
3. Kneale, W & Kneale, M 1962. The development of logic, *Oxford University Press*.
4. Boole, G 1957. The laws of thought, Dover, *New York* (original edition 1854).
5. Boole, R 1947. The Mathematical Analysis of Logic, *Philosophical library*.
6. Shannon, CE 1938. A symbolic analysis of relay and switching circuits, *Electr. Eng.* Vol. 57, № 12. P. 713–723.
7. Foundations of Computer Science/Computing Machinery 2017. *Wikibooks, open books for an open world*. Available from: <https://en.wikibooks.org/wiki/Foundations_of_Computer_Science/Computing_Machinery> [05 May 2021].