How to tell the truth without knowing what you are talking about
George Boole and the Boolean algebra

Luca Anselma, Davide Cavagnino
Dipartimento di Informatica – Università degli Studi di Torino

> ἔστω δὲ ὁ λόγος ὑμῶν ναὶ ναί, οὒ οὔ: τὸ δὲ περισσὸν τούτων ἐκ τοῦ πονηροῦ ἐστιν.
> Matthew 5:37

> But let your communication be, Yea, yea; Nay, nay: for whatsoever is more than these cometh of evil.
> Matthew 5:37 (King James version)

Introduction

Is it possible to verify whether a statement is true, even without knowing the subject of the discussion? More precisely, is it possible to extract secure conclusions from a set of premises describing what you may not know? The answer is positive provided that the reasoning, from the accepted set of premises to the conclusions, is correct. The problem of verifying whether a reasoning is correct is one of the aims of logic since Aristotle. In particular, modern logic, that is logic since the Renaissance, aims at improving human reasoning and overcoming its fallacies, significantly by mechanizing reasoning. To this aim, it is necessary to improve logic passing from the syllogistic doctrine towards more powerful forms of reasoning and to devise a precise logical language free from the ambiguities of natural language: modern logic witnessed the birth of *symbolic logic*. Whereas Aristotelian and scholastic logic has a catalog approach, based on classification and categorization, modern logic is a calculus. In fact, in modern logic a symbolism must be devised, together with inference rules for manipulating symbols to discover new truths or to verify whether a conclusion can be derived from the premises.

Today logic is considered a branch of mathematics and this mathematical entry into logic dates to the notable precursory works of Leibniz, who aimed at devising a *lingua characteristica universalis*, a universally characteristic language, and a *calculus ratiocinator*, a calculus of reason. However, Leibniz's works on logic were published only at the end of the 19th century, so the influence of his ideas on the emergence of modern logic has been very limited.

In the 19th century, there was a main paradigm shift in mathematics: the perception that algebra could deal not only with numbers but, more abstractly, with symbols. It was in this milieu that George Boole arrived at a mathematical treatment of logic by proposing an algebra not of numbers, but of logic. As Boole writes in the introduction of his *The Mathematical Analysis of Logic*: "*... the validity of the processes of analysis does not depend upon the interpretation of the symbols which are employed, but solely upon the laws of combination.*"

Symbolic logic and in particular Boole's Boolean algebra has had major repercussions on computer science. In fact, in the first half of the twentieth century many scientists showed that the Boolean logic could be easily implemented in electronic circuits and could be used to build the various functions of a computer. Moreover, it is used in search engines and in database management systems to query data.

As an example to illustrate Boolean algebra, we will show that it is possible to solve the following simple detective mystery case and establish the truth as a valid conclusion by applying only Boolean algebra.

> ***A Boolean mystery***
> *A murder has been committed. The detective questions three persons: Alice, Bob and Cormac.*

> *Let us suppose that a liar always lies, i.e., all that s/he says is false and that a truth-teller always tells the truth, i.e., all that s/he says is true. Each person is either a liar or a truth-teller.*
> *Alice says: "Cormac committed the murder and Bob is a liar". Bob says: "Cormac is a liar". Cormac says: "Alice and Bob are both liars".*
> *Did Cormac commit the murder?*

This paper is structured as follows: the next section is devoted to a brief presentation of the life of George Boole while the following section presents the Boolean algebra, the principal research result of Boole. Before drawing some conclusions at the end of the paper, we will analyze the previous example and we will solve it by means of the Boolean algebra.

George Boole

One of the most important people of the 19th century, whose pioneering and revolutionary ideas still influence our everyday life (the logic of computers, for example), is Professor George Boole.
Born on November 2, 1815 in Lincoln, a town in the north-east of England, he grew up there, studying by himself and with the help of his father, a shoe vendor but also a person interested in scientific matters. Anyway, Boole's first studies were not dominated by mathematical-scientific topics; his interests were towards classical studies: he learned Latin and Greek, and some of the most important European languages of his period, namely Italian, French and German. It is not by chance that he loved reading Dante Alighieri's works, in particular the well known *Divina Commedia*.
His father's financial problems forced him to leave his studies during his youth, but this did not prevent him from deepening his culture by studying autonomously and obtaining the excellent result of becoming a teacher at various levels.
George Boole did not forget his modest origins and his religious beliefs, even when he achieved a brilliant career. To help poor people to get an education he founded a school, The Mechanics Institute, and The Female Penitents Home, where prostitutes could change their lives for the better.
George Boole, being a self-taught person, was influenced by some casual contacts, not only by the relationship with his friend and mentor Duncan Gregory, an English mathematician who encouraged Boole as a teenager to study mathematics, but also by the controversy, which lasted several years, over the quantification of the predicate between the philosopher William Hamilton and the mathematician Augustus De Morgan.
The scientific community became aware of the attention of Boole for mathematics when, in 1840, he published his paper *Researches on the Theory of Analytical Transformations with a special application to the reduction of the general equation of the second order* in the Gregory's Cambridge Mathematical Journal, and when he received, at the age of twenty-nine, the Royal Society Medal for his work on differential equations.
These results, along with many others, allowed him to become professor of mathematics in 1849 at the Queen's College of Cork (today University College of Cork). Some years later, Boole married Miss Mary Everest (whose name associates with the highest mountain in the world, named in honour of her uncle, the geographer Colonel George Everest), and they had five daughters.
In 1854 he published *An Investigation of the Laws of Thought, on which are founded the Mathematical Theories of Logic and Probabilities*, following and improving his previous publication *The Mathematical Analysis of Logic, being an Essay Towards Calculus of Deductive Reasoning* (1847). Briefly, the objective of these works was to define a reasoning model that could be expressed using symbols and a mechanical method of deduction based on rules through the solution of algebraic equations. In other words, Boole found that logic, provided with an adequate representation, is a branch of algebra, thus freeing logic from the tradition of philosophy.

In addition to those previously cited, Boole published many other works and articles, until his sudden death on December 8, 1864 in Cork, where he was buried near the church of St. Michael. In the following sections we will present an introduction to some aspects of the works of 1847 and of 1854.

Boolean algebra

Boole developed an algebra defined by variables and operations with some particular properties. Boolean algebra is applied to two values, namely 0 and 1. A natural interpretation of these values may be obtained considering the value 1 as TRUE and the value 0 as FALSE.
The unary operator of negation (or NOT), and the binary operators of conjunction (or AND) and of disjunction (or OR) may be applied to these values as shown in the following.

Negation

The simplest operator that one finds in the Boolean algebra is the negation (or complement). Negation is a unary operator, i.e., it is applied to one term only. The negation of a term *a* is written as $\neg a$ or, alternatively, NOT *a*.
Intuitively, the negation inverts the truth value of a term: if *a* is TRUE, NOT *a* is FALSE, and if *a* is FALSE, NOT *a* is TRUE.
The behaviour of negation is represented in the following so-called truth table[1].

a	**NOT *a***
0	1
1	0

For example, let *a* be "I did study"; NOT *a* corresponds to "I did not study". If "I did study" is TRUE, then "I did not study" is FALSE, and vice versa.
It is worth noting that it is possible to combine, in a formula, more operators. For example it is possible to have a double negation: NOT NOT *a*. Since NOT inverts the truth value of a term, a double negation inverts twice the value of a term, therefore it affirms it and it is possible to eliminate a double negation, i.e., NOT NOT $a = a$. This is similar to a particular use of double negation in Standard English ("I do not want nothing" means that I want something), but different from some other natural languages, such as Italian or French, where it is customary that a double negation negates ("Non voglio niente" and "Je ne veux rien" both mean that I do not want anything), or some dialects of English ("Whatcha got ain't nothin' new" says a Texan sheriff in the movie *No country for old men*, meaning "What you have received is nothing new").
If we apply the NOT operator to the title of this paper, it would result in "how to *not* tell the truth (even) without knowing what you are talking about", which is, apparently, the job of someone else, not of the logician.

Conjunction

Unlike negation, conjunction is a binary operator, i.e. it is applied to two terms. The conjunction of two terms *a* and *b* is written as *ab* or $a \wedge b$ or, alternatively, *a* AND *b*. Intuitively, the conjunction of two terms is TRUE if (and only if) both terms are TRUE. In other words, if either *a* or *b* or both are FALSE, then *a* AND *b* is FALSE; if both *a* and *b* are TRUE, then *a* AND *b* is TRUE.
The behaviour of conjunction is represented in the following truth table.

[1] Truth tables are an intuitive method for defining operations in Boolean algebra. They were introduced by George Boole in *The Mathematical Analysis of Logic* and made popular by Ludwig Wittgenstein in his *Tractatus Logico-Philophicus*.

a	***b***	***a* AND *b***
0	0	0
0	1	0
1	0	0
1	1	1

For example, let *a* be "I did not study" and *b* be "I did not pass the exam". *a* AND *b* is "I did not study and I did not pass the exam". We can verify whether *a* AND *b* is true by examining the truth value of *a* and of *b*. For example, if I did study and did pass the exam, then *a* is FALSE and *b* is FALSE, and, therefore, also *a* AND *b* is FALSE.

It is possible to express the principle of non-contradiction as the true proposition "NOT(*a* AND NOT *a*)", i.e., it is not possible that *a* is both true and false. By applying the rules of conjunction and of negation, it is possible to verify that in Boolean algebra the principle of non-contradiction is a tautology, i.e., it is necessarily true.

It is interesting to notice that, in order to evaluate the truth value of *a* AND *b*, it is sometimes not necessary to know both the values of *a* and of *b*. For example, if *a* is FALSE, *a* AND *b* is necessarily false, independently of the value of *b*. In the above example, if "I did not study" is FALSE, then "I did not study and I did not pass the exam" is FALSE, both whether "I did not pass the exam" is TRUE or FALSE. However, if the value of *a* is TRUE, it is necessary to test also the value of *b* to evaluate *a* AND *b*. This feature is known as short-circuit evaluation of a Boolean formula.

Disjunction

The disjunction of two terms *a* and *b* is written as *a*+*b* or $a \vee b$ or, alternatively, *a* OR *b*.

Intuitively, the disjunction of two terms is FALSE only when *a* and *b* are both FALSE. In other words, if *a* is TRUE or *b* is TRUE or both *a* and *b* are TRUE, then *a* OR *b* is TRUE. Thus, the OR operator is an inclusive disjunction resembling the Latin *vel*.

The behaviour of disjunction is represented in the following truth table.

a	***b***	***a* OR *b***
0	0	0
0	1	1
1	0	1
1	1	1

For example, let *a* be "I did not study" and *b* be "The exam questions were easy". *a* OR *b* is "I did not study or the exam questions were easy". If both "I did not study" (i.e., *a*) is TRUE and "The exam questions were easy" (i.e., *b*) is TRUE, then we can conclude that "I did not study or the exam questions were easy" (i.e., *a* OR *b*) is TRUE.

It is possible to express the Aristotelian *tertium non datur* or law of excluded middle using disjunction and negation as the tautology "*A* OR NOT *A*", i.e., *A* is either true or not true.

As for the conjunction, also for the disjunction we can apply the short-circuit evaluation, i.e., there are cases where we can evaluate the truth value of a formula even if we do not know the truth values of all the terms. If *a* is TRUE, then necessarily, *a* OR *b* is TRUE, both whether *b* is TRUE or FALSE. In the example, if "I did not study" is TRUE, then "I did not study or the exam questions were easy" is necessarily TRUE.

As mentioned above, the main discovery of George Boole was a connection between logic and arithmetic. In fact, it is possible to reduce the evaluation of a Boolean formula to a mathematical formula, using the basic arithmetical operators. In particular, using the equivalence between 0 and

FALSE, and between 1 and TRUE, we see that the AND operation may be computed as a product, the OR operation as a sum and the NOT operation may be computed as a complement or difference. More in detail, *a* AND *b* corresponds to the product of *a* and *b*, or logical multiplication. In fact, *a* AND *b* is TRUE if and only if both *a* and *b* are TRUE, as the product of *a* and *b*, i.e., *ab*, is 1 if and only if both *a* and *b* are 1; and *a* AND *b* is FALSE if and only if either *a* or *b* or both are FALSE, as the product of *a* and *b* is 0 if and only if either *a* or *b* or both are 0.

The disjunction is analogous: *a* OR *b* corresponds to the sum of *a* and *b*, or logical addition. In fact, *a* OR *b* is FALSE if and only if both *a* and *b* are FALSE, as the sum of *a* and *b*, i.e., *a*+*b*, is 0 if and only if both *a* and *b* are 0; and *a* OR *b* is TRUE if and only if either *a* or *b* or both are TRUE, as the sum of *a* and *b* is 1 (or above) if and only if either *a* or *b* or both are 1. However, in this case, one must take particular care, in fact one can obtain as a result a number different from 0 and 1, as in 1+1=2 corresponding to TRUE OR TRUE. In this case it is necessary to remain in the set of admissible values, and consider that 1+1=1 and not 2.

For the negation, we can compute the result of NOT *a* as 1–*a*. In fact, NOT *a* is TRUE if and only if *a* is FALSE, as 1–*a*=1 if *a*=0; and NOT *a* is FALSE if and only if *a* is TRUE, as 1–*a*=0 if *a*=1.

The careful reader may have noticed that the descriptions of conjunction and disjunction share some analogies. In fact, in the above sentences, if one exchanges TRUE with FALSE, 0 with 1, and "either" with "both", the descriptions of the operators are identical. This observation reflects the duality property of Boolean formulas, called De Morgan's laws, named after Augustus De Morgan. In fact, it is possible to express disjunction by means of conjunction (and negation), and, vice versa, it is possible to express conjunction by means of disjunction (and negation):

NOT *a* AND NOT *b* = NOT(*a* OR *b*)
NOT *a* OR NOT *b* = NOT(*a* AND *b*).

Let us consider the second equivalence. Intuitively, *a* and *b* are not both TRUE (right-hand side of the equivalence) if and only if *a* is not TRUE or *b* is not TRUE or both *a* and *b* are not both TRUE (left-hand side). For example, if it is not true that I did study and the exam questions were easy, then either I did not study or the exam questions were not easy (or both).

It is possible to employ De Morgan's laws to reduce the principle of non-contradiction to the law of excluded middle, in fact:

NOT(*a* AND NOT *a*) = NOT *a* OR NOT NOT *a* = NOT *a* OR *a*.

One may define other logical operators besides the three basic ones. In fact, it may be easily shown that it is possible to define 2^4=16 different operators of two variables that may assume the values 0 and 1. OR and AND are two of such operators; six other operators are the trivial ones that output only the first or the second argument, their negations, always TRUE or always FALSE. Of the remaining operators, it is interesting to illustrate the operations of implication, bi-implication and exclusive disjunction.

Implication

The implication operation between two terms *a* and *b* is written as *a* → *b*.

Intuitively, it corresponds to a certain way of using "if… then…" in English. On the one hand, if the antecedent (the "if" part) is true, then the consequent (the "then" part) must be true. This is equivalent to the syllogistic rule of inference called *modus ponens* in syllogistic theory. On the other hand, if the antecedent is false, then the consequent can be either true or false; this is equivalent to the rule of inference *ex falso quodlibet*, i.e., from a false proposition, anything follows. An anecdote recalls that during a public lecture Bertrand Russell was making a remark that from a contradiction everything can be deduced. A man from the public challenged him to prove that if 4=5 then he was the Pope. And he brilliantly did it (and we leave to the reader the pleasure to find out how!).

The implication is defined as:

a	***b***	***a* → *b***
0	0	1
0	1	1
1	0	0
1	1	1

One may note that only the case where *a*=TRUE and *b*=FALSE does not satisfy the formula *a* → *b*. For example, let *a* be "I did not study" and *b* be "I did not pass the exam". *a* → *b* is "if I did not study, then I did not pass the exam". If "I did not study" is TRUE, also "I did not pass the exam" must be true; on the other hand, if I did study (i.e., "I did not study" is FALSE), then I am not guaranteed that I would have passed the exam (i.e., "I did not pass the exam" can be either true or false).
Implication may be written by means of basic operators as:
a → *b* = NOT *a* OR *b*, or, equivalently, *a* → *b* = NOT (*a* AND NOT *b*),
that is, *a* → *b* is equivalent to "*a* is FALSE or *b* is TRUE" or, equivalently, "it may not happen that *a* is TRUE and *b* is FALSE".
It is interesting to note that this operation is *not* commutative, in the sense that the resulting value depends on the order of the input values. Instead, the other operators described in this paper are commutative because changing the order of the input values does not change the result of the operation (as can be easily seen from the respective truth tables).

Bi-implication
Another useful operation is the bi-implication, written as *a* ↔ *b*, that outputs 1 if and only if the inputs are equal, i.e.:

a	***b***	***a* ↔ *b***
0	0	1
0	1	0
1	0	0
1	1	1

This operation is also called equivalence because for the operation to be TRUE the value of one of the inputs implies the value of the other and vice versa. In other words, the bi-implication corresponds to the conjunction of two implications: *a* ↔ *b* = (*a* → *b*) AND (*b* → *a*).
For example, let *a* be "I did not study" and *b* be "I did not pass the exam". *a* ↔ *b* is "I did not study if and only if I did not pass the exam". If "I did not study" is TRUE, also "I did not pass the exam" must be TRUE; on the other hand, if I did study (i.e., "I did not study" is FALSE), then I am guaranteed that I pass the exam (i.e., "I did not pass the exam" must be FALSE).

Exclusive disjunction
Another interesting operation is the one called exclusive disjunction, written as *a* XOR *b*, that outputs 1 if and only if the two input values are different, i.e.:

a	***b***	***a* XOR *b***
0	0	0
0	1	1
1	0	1
1	1	0

Intuitively, the XOR operation models the “either … or …” construct of English, or the *aut aut* in Latin: the formula is true if (and only if) one and only one of the terms is true. For example, let *a* be “you do it straight away” and *b* be “there will be trouble”, *a* XOR *b* is “either you do it straight away or there will be trouble”, meaning that, if you do it straight away, there will not be trouble, and, if you do not do it straight away, there will be trouble.
The exclusive disjunction can be written by means of disjunction, conjunction and negation:
a XOR *b* = (*a* OR *b*) AND NOT(*a* AND *b*),
that is, *a* XOR *b* is equivalent to “*a* or *b*, but not both *a* and *b*”.

An application of Boolean algebra
As an application of Boolean algebra we illustrate the solution of the detective mystery previously introduced.

A solution to the Boolean mystery
We solve the puzzle in two ways: by means of inferences and by means of Boolean algebra. First, we express the information provided in the mystery as a Boolean formula, by conveniently formalizing the relevant information as Boolean terms.
For example, let us call *M* the term expressing the fact that “Cormac committed the murder”. *M* is true if Cormac committed the murder, and false otherwise. *A* is the term expressing the fact that “Alice is a truth-teller”, and the same for *B* and Bob, and for *C* and Cormac. The (conjunction of the) following formulas represent the mystery:
A ↔ NOT *B*, i.e., Alice says: “Bob is a liar”;
B ↔ NOT *C*, i.e., Bob says: “Cormac is a liar”;
C ↔ (NOT *A* AND NOT *B*), i.e., Cormac says: “Alice and Bob are both liars”;
A ↔ *M*, i.e., Alice says: “Cormac committed the murder”.

By applying the inference rules relative to the proper operators, we can show that Cormac did not commit the murder by trying to assign a truth value to a term and by applying the proper inference rules for the relative operators as outlined above. For example, we assign TRUE to the term *M*, that is we suppose that Cormac committed the murder and, as in *reductio ad absurdum*, we show that it is inconsistent.
If Cormac committed the murder (and thus the term *M* evaluates to TRUE), as a consequence of the bi-implication operator, also *A* must be true, that is Alice is a truth-teller. Therefore, NOT *B* must be true, i.e., *B* must be false and Bob is a liar. If *B* is false, also NOT *C* is false and Cormac is a truth-teller.
If Cormac is a truth-teller, also NOT *A* AND NOT *B* must be true so, from the definition of the conjunction, both NOT *A* and NOT *B* must be true, i.e., Alice and Bob are liars. This is a contradiction, since above we have concluded that Alice is a truth-teller.
We could repeat the reasoning above supposing that Cormac did not commit the murder (i.e., *M* is FALSE) and we would find that this is the only consistent scenario.

By applying the Boolean algebra, the mystery corresponds to the following formula:
(A ↔ NOT *B)* AND *(B ↔* NOT *C)* AND *(C ↔* (NOT *A* AND NOT *B*)*)* AND *(A ↔ M)*
To simplify the formula, we can observe that bi-implication, being an equivalence, allows us to substitute its left-hand side with its right-hand side. Thus, substituting *A* with *M*, NOT *B* with *A* (and, thus, with *M*), and NOT *C* with *B* (and, thus, with NOT *M*), the formula becomes
*(M ↔ M) AND (*NOT *M* ↔ NOT *M)* AND *(M ↔* (NOT *M* AND *M*)*)* AND *(M ↔ M)*.

In the formula, the first, the second and the fourth conjuncts are trivially true, because they are tautologies. Thus, thanks to the short-circuit evaluation property for the conjunction, the formula is true if and only if the third conjunct is true.
In order to evaluate this conjunct, it is useful to observe that, since $a \leftrightarrow b$ corresponds to ($a \rightarrow b$) AND ($b \rightarrow a$), i.e., to (NOT a OR b) AND (NOT b OR a), we can write the expression as:
(NOT M OR (NOT M AND M)) AND (NOT(NOT M AND M) OR M),
which corresponds to the following algebraic expression:
$((1-M) + ((1-M) M)) (1-((1-M) M) + M)$
which must be equal to 1. The algebraic solution of the equation leads to M=0, therefore M must be false and Cormac did not commit the murder.

It is worth noting that we have been able to solve the case with no prior knowledge about the facts and that this reasoning can also be performed automatically.

Conclusions

In this paper we have briefly illustrated the Boolean algebra, which was proposed by George Boole in the 19th century. Thanks to Boolean algebra it is possible to reduce both the syllogistic theory and propositional logic to simple algebraic calculations.
Boolean algebra has been an important step in the rise of symbolic logic, thanks to the significant influence in the mathematization of logic, and to the repercussions in the design of digital electronic circuits, its use in significant computer applications, and to mathematics and computer science (e.g., set theory, probability theory, information theory). In particular, in the 1930s a 21-year-old master's student called Claude Shannon, after attending a course on the works of George Boole, showed in his master's thesis that it is possible to use electromechanical relays to solve Boolean algebra problems, thus laying the foundations for the designing of computers and actually achieving the goals of Leibniz's *calculus ratiocinator*.
In the application of Boolean logic to computer circuits the connotational overtones of TRUE/FALSE tend to be substituted with 1/0, high/low, yes/no, or ON/OFF, etc.
Further developments in logic and axiomatization of Boolean algebra allowed to answer some appealing questions such as whether Boolean logic is "enough". In other words, can Boolean logic capture all information that is "interesting" to use? It has been proven that Boolean algebra (and its equivalent propositional logic) is both sound and complete, i.e., in Boolean algebra one can derive all and only "truths" expressible in Boolean algebra. Moreover, Boolean algebra is one of the simplest calculi in logic. Logicians have devised other and more powerful logical systems, such as first-order logic (e.g., "All dogs are animals") and higher-order logics, modal logics (e.g., "It is possible that a dog is black, but it is necessary that a dog is an animal"), many-valued logics and fuzzy logic (that contemplate many degrees of truth and vague concepts), intuitionistic logic (which rejects *tertium non datur* and double negation elimination) and many others.

Acknowledgments

The authors thank Prof. A. E. Werbrouck and Dr. S. Bresso for the significant help they gave in the preparation of this paper.

References

G. Boole, *The Calculus of Logic*, Cambridge and Dublin Mathematical Journal, vol. 3, pages 183-198, 1948.
P. Odifreddi, *Il diavolo in cattedra*, Einaudi, 2003.

D. MacHale, *George Boole: his life and work*, Boole Press (http://www.boolepress.com), 1985.

W. Reville, *The greatness of George Boole*, http://www.ucc.ie/academic/undersci/pages/sci_georgeboole.htm visited on December 30, 2009; the article was first published in *The Irish Times* on May 20, 1996.

L. de Ledesma, A. Perez, D. Borrajo, L. M. Laita, *A computational approach to George Boole's discovery of mathematical logic*, Artificial Intelligence, 91(2), Scientific Discovery, pages 281-307, April 1997, ISSN 0004-3702, DOI: 10.1016/S0004-3702(97)00017-9.

L. M. Laita, *Influences on Boole's logic: The controversy between William Hamilton and Augustus De Morgan*. Annals of Science, 36(1), pages 45-65, 1979, doi:10.1080/00033797900200121.

D. M. Gabbay, J. Woods (eds.), Handbook of the History of Logic, volume 3: The Rise of Modern Logic: From Leibniz to Frege, Amsterdam et al.: Elsevier-North-Holland, 2004.

V. Peckhaus, *Leibniz's Influence on 19th Century Logic*, The Stanford Encyclopedia of Philosophy (Fall 2009 Edition), Edward N. Zalta (ed.), URL: http://plato.stanford.edu/archives/fall2009/entries/leibniz-logic-influence/