The Organon and the logic perspective of computation

Panagiotis Katsaros, Nick Bassiliades, Ioannis Vlahavas
e-mail: {katsaros, nbassili, vlahavas}@csd.auth.gr
Dept. of Informatics, Aristotle University of Thessaloniki, Greece

The works of Aristotle on logic are collectively known as the *Organon*, that is, the "instrument" or "tool" of thought. In the "Prior Analytics", Aristotle introduced a list of inference rules that concern with the relation of premises to conclusion in arguments (syllogisms). His aim was to determine which kinds of arguments are valid. The validity of an argument is characterized and inferred based on its *logical form* (deduction) and for this reason Aristotle is considered as the father of formal logic. A few years later, the Stoic philosophers developed a logic of propositions as the primary bearers of truth-values, aiming at reasoning over the language by using the logical connectives "not", "and", "or" and "if . . . then". However, Aristotle's conception for logic was the dominant form of Western logic with no breakthroughs until the 19th century advances in mathematical logic.

These started with George Boole, who found an analogy between propositional operators and the operators of set theory, thus giving an extensional interpretation to propositional logic. A few years later, Gottlob Frege developed a synthesis of the logic of Aristotle with the Stoics' propositional logic. This enabled the representation of relations, whereas Aristotle's logic was limited to predicates applying to a single subject. Frege's "Begriffsschrift" introduced the first symbolic logic language, called predicate logic, with precisely defined syntax rules that handle all combinations of quantifiers and negation, as well as conjunction, disjunction and conditionals.

The notion of computation came to the foreground due to the decision problem (known as "Entscheidungsproblem") that was posed by David Hilbert: is there an *effective* or *mechanical* method – called decision procedure – that can be applied to any proposition to determine whether it is provable in predicate logic? Hilbert's motivation was a new deductive system, in which the grammar and the inference rules are *explicitly stated* in syntactical terms. Kurt Gödel quoted that "this deductive system should refer only to the outward structure of the formulas, not their meaning, so that they can be applied by someone who knew nothing about mathematics, or by a *machine*". Such deductive systems are syntactic devices for establishing semantic properties and are related to the general *models of computation*, later introduced by Alan Turing and Alonzo Church, which aim to a purely mechanical transformation procedure to sequences of symbols.

In this article, we attempt an epitome of the fundamental principles of the theory of syllogisms and their impact to the foundations of modern logic reasoning and in particular the study of computation during the 20th century.

1. Introduction

At a moment of culmination of the philosophical thought, ancient Greek philosophers were focused on a rational explanation of the world based exclusively on evidence and reasoning. In this era, Aristotle saw the need for a separate discipline to study and develop the act of pure reason, irrespective of what it is about. He wrote five treatises, known collectively as the Organon, that are still the object of study and of inspiration for scholars around the world, towards the quest of new logical insights [1-12].

Having seen the Organon as a tool used by all the sciences, Aristotle introduced a theory of *deductive inferences*, which are known as *syllogisms*. The syllogism is defined as "a *logos* (speech) in which, certain things having been supposed, something different from the things supposed results of necessity because of their being so". However, logical inference is not a matter of the contents of the supposed things. Schemes with dummy letters (variables) are used to stand for the terms (subjects and predicates) in the premises, like e.g. that "All A are B" or that "No B is C". A scheme cannot be a syllogism, if there are terms to substitute for the letters that make the premises true and the conclusion false. This characterization of an argument's *validity* based on its *logical form* was first achieved by Aristotle, who is therefore considered as the father of *formal logic*. The syllogisms were the dominant form of logical reasoning with no major breakthroughs until the 19th century advances in mathematical logic, where the focus of interest is the logical reasoning in artificial languages.

In this work, we trace back the principles of Aristotle's theory of syllogisms and their impact to the developments in mathematical logic. The groundbreaking vision of G. Leibniz towards the reduction of reasoning into calculation (Characteristica Universalis) [13] had inspired continuous efforts for mechanizing the performance of logical inference. Through the idea of *calculus*, logic was developed into a system with a precisely defined vocabulary and two types of formal rules: *formation rules* stating the kind of sequences of symbols that are well formed and *transformation rules*, which are the basis on which logical reasoning is performed like calculating. Two important milestones in this direction were G. Boole's logic algebra [14] and G. Frege's predicate logic [15]. Through the formulation of the *decision problem for predicate logic* by D. Hilbert – known as the "Entscheidungsproblem" – [16] the first rigorous definitions of the notion of an *effective* or *mechanical procedure* to compute functions were eventually achieved [17], i.e. the general *models of computation* by A. Turing and A. Church, which were eventually found to be equivalent. Apart from the historic connection between logical reasoning and computation, both of them focus on the *explicit formalization* of rule-based inference by the mediation of some *language* that fulfills certain requirements of preciseness. We therefore examine the structure and the properties of the language of syllogisms, in comparison with the languages of the classical logics and the languages of the more recent theory of computation.

In another interesting direction, the Curry - Howard isomorphism is a generalization of a syntactic analogy between systems of formal logic and programs that establishes a direct relationship between them and an operational interpretation of modern and classical mathematical logics. The widespread characterization of logic as the "Calculus of Computation" [18] refers to the emerged *computational logics*, which aim to develop efficient *decision procedures*, for the proof of formulas in logical theories and logic-based programming paradigms. To this respect, we discuss the recent computational formulations of Aristotle's

logic, the development of *logic programming* and the emergence of *automated theorem proving*.

By turning our gaze to the past, the Organon was not just a "tool" of thought, but it was also the source of inspiration for the development of thought towards a mechanistic view of cognition and to the foundation of the modern *theory of computation*.

The current article is structured as follows. In Section 2, we discuss the birth of formal deductive reasoning, due to the theory of syllogisms, its fundamental principles and properties, the expressed criticisms from the perspective of mathematical logic and the consequent rebuttal arguments. In Section 3, the principles of Aristotelian logic are tracked into the classic languages of mathematical logic. Section 4 refers to the long-standing challenge of mechanizing logical inference, the negative answer to the decision problem for predicate logic and the multifaceted connection between the logical reasoning and the notion of computation. Section 5 is focused on the development of computational logics and the paper concludes with a discussion on the overall impact of the Organon to the logic perspective of computation.

2. The Organon and the birth of formal deductive reasoning

2.1. Definitions

To facilitate our presentation, we adopt a number of definitions for standard concepts found in all logical theories. The precise meaning of most of these concepts was first adopted and substantiated in the context of the Organon.

An *argument* consists of one or more propositions, known as *premises*, offered as *a reason* for a further proposition, known as *conclusion*, *to be true*. The meaning for a proposition to be true is that it corresponds to reality. A *deductive argument* aims to show that its conclusion *must*[1] be true, i.e. that *if the premises are true, then the conclusion is true as well*. An *inductive argument* aims to show that its conclusion is *probably* true although not certainly true.

Valid argument is a deductive argument, for which it *is* the case that if the premises are true, then the conclusion *must* be true. Notice that an argument can be valid, even though it has false premises and a false conclusion. It is clear that in logic, “valid” does not mean true. On the other hand, an *invalid argument* is a deductive argument in which it is *not* the case that if the premises are true then the conclusion must be true.

Validity is not the sole concern in reasoning, since we are also interested for the truth of our arguments, i.e. their correspondence with reality. An argument is said to be *sound*, if all of its premises are true and it is also valid.

Thus, a logical theory is useful if it is sound, but apart from the validity of arguments we want to be able to decide whether two propositions stand in logical conflict or they are characterized by another possible logical relation. Two propositions are said to be *consistent*, if and only if it *is possible* both are true. Two propositions are *inconsistent*, if and only if it *is not possible* that both are true. One proposition *implies* a second proposition, if and only if it

[1] Recall the characterization “results of necessity” in Aristotle’s definition of the syllogism.

is not possible that the first proposition is true and the second is false. Finally, two propositions are *equivalent*, if and only if the first one implies the second and the opposite is also true.

2.2. Categorical syllogisms

Aristotle's theory of syllogisms is based on a language for *categorical arguments*, like the following:

i. No birds are mammals.
ii. Some birds are pets.
iii. So, some pets are not mammals.

Aristotle studied this sort of arguments, by breaking them into their essential parts. Every categorical proposition expresses a relation between two categories of things by stating either that all, none, or some of one category, belong to, or do not belong to, a second category of things. It is broken into a *quantifier*, a *subject term*, a *predicate term* and a *copula* that joins the subject and the predicate and states whether the quantity refers to things that belong to the predicate category or that do not belong to it. There are only four basic types of propositions defined using dummy letters (according to [1], Aristotle was the first who introduced variables in logic reasoning):

Universal affirmative (proposition *A*[2]): All *X* are *Y*.

Universal negative (proposition *E*): No *X* are *Y*.

Particular affirmative (proposition *I*): Some *X* are *Y*.

Particular negative (proposition *O*): Some *X* are not *Y*.

Our example argument is now generalized[3] to the following form:

i. No *X* are *Y*.
ii. Some *X* are *Z*.
iii. Some *Z* are not *Y*.

Every *categorical syllogism* is thus defined as an argument composed of two premises and one conclusion, all of which are categorical propositions expressed with only three different terms. Each term appears exactly twice. The predicate of the conclusion is called the *major term*, whereas its subject is known as the *minor term* (the *middle term* appears in both premises). The premise introducing the major term is the *major premise* and the premise with the minor term is the *minor premise*. A categorical syllogism is said to be in *standard form*, if the major premise appears first and the conclusion appears last. The arrangement of the middle term determines the *figure* of the syllogism, e.g. if the middle term appears as a subject term of the first premise and as predicate term of the second premise, then we have a syllogism with figure 1. There are four possible placements for the middle term, and thus four possible figures[4]. The listing of the letters for a syllogism's propositions determine its *mood*.

[2] The labelling of the four types of propositions was introduced by scholars in the Middle Age.
[3] In fact, there is no example with concrete terms in the Prior Analytics.
[4] Aristotle originally identified the three of the four possible figures; the last one was added later.

These definitions eventually result in determining the *logical form* of a syllogism; our example's mood is EIO and its figure is 3. From all possible 256 forms, only 24 are unconditionally *valid* patterns of reasoning.

The theory of syllogisms was the first logic system based on axioms known as *laws*, which are mentioned in the Metaphysics (and in the Analytics) and are used for justifying valid inference. The *law of identity* states that:

- "All *A* are *A*" and
- "Some *A* are *A*"

The *law of excluded middle* or *excluded third* says that for every proposition, either its positive or negative form is true. Finally, the *law of non-contradiction* in the Metaphysics, states that "the most certain of all basic principles is that contradictory propositions are not true simultaneously". Propositions which are "opposed" to one another are such that either if they were asserted to be true together, or if they were asserted to be false together, a contradiction would result. If a contradiction results only when they are asserted true together, they are called *contraries*; if a contradiction results only when they are asserted false together, they are called *sub-contraries*; if a contradiction results in both cases they are called *contradictories*. The *square of opposition* in Aristotelian logic is a diagram that helps to show why equivalences and implications are necessary, since logical necessity means *"necessary to avoid a contradiction*". Additionally, through the square of opposition we can get an equivalent of a proposition *p*, given as the denial of the contradictory of *p*: the necessary consequent of *p*'s denial is *p*'s contradictory.

2.3. Deductive proofs and algorithms

Syllogistic is a powerful proof system. Aristotle defines the notion of *proof* in Posterior Analytics as "a syllogism which creates scientific knowledge, that is such a syllogism owing to which, if we only have the proof, we possess that knowledge". And he continues: "Hence if knowledge is such as we have stated, then the premises of demonstrative knowledge must be true, primitive, direct, better known (than the conclusion) and must be its cause".

Aristotle explicitly distinguishes between proofs in two categories, the *perfect* or *complete deductions* and the *imperfect* or *incomplete* ones. A perfect deduction "needs no external term in order to show the necessary result", whereas an incomplete deduction "needs one or several terms in addition that are necessary because of the terms supposed but were not assumed through the premises". Aristotle identified six different forms of perfect deductions, for which he felt that the premises are stated in such a way that the conclusion follows necessarily and there is no need to prove it.

For imperfect deductions, Aristotle gives proofs, in which he "reduces"[5] each case to one of the perfect forms and they are thus "completed" or "perfected". Two approaches may be followed to "complete" a proof, namely *direct deduction* or *through the impossible* (reductio ad absurdum). Direct deductions start with a number of propositions assumed as premises and the steps from the premises to the conclusion may be either inferences from two previous steps by perfect first-figure moods used as deduction rules or *conversions*. The laws of

[5] This is the original term translated from the Greek verb «ανάγει».

conversion are simple implications, for instance: "If all *B* are *A*, then some *A* are *B*". Aristotle has identified three such implications regarded as true for all values of the variables A and B. In each conversion, a new proposition is inferred from an existing proposition, which has the subject and predicate interchanged. As an example, we provide Aristotle's direct deduction of the figure 2 syllogism AEE:

i.	All *C* are *B*.	(major premise)
ii.	No *A* are *B*.	(minor premise)
iii.	No *A* are *C*.	(conclusion)

Proof:	Step 1 – All *C* are *B*	(proposition A)
	Step 2 – No *A* are *B*	(proposition E)
	Step 3 – No *B* are *A*	(E from step 2 by conversion of E)
	Step 4 – No *C* are *A*	(from steps 3 & 1 by figure 1 perfect deduction EAE)
	Step 5 – No *A* are *C*	(conclusion from step 4 by conversion of E)

In a reductio ad absurdum proof, the conclusion is inferred from a pair of premises by assuming as a third premise the denial of the conclusion, in order to deduce from it and one of the original premises the contrary of the other premise.

For premise – conclusion combinations that cannot be valid syllogisms, Aristotle adopts the *proof by counterexample*, a method that we still use in modern logic: given a pair of premises of an invalid form, it should be possible to construct examples in which premises of that form are true and a conclusion of any of the four possible forms is false.

The syllogistic also features detailed instructions for how to search for premises that prove or invalidate the different forms of syllogisms. They are algorithms, i.e. they yield a definite outcome after a finite number of steps. In the Prior Analytics, Aristotle introduces the long discussion, where he exposes this set of algorithms, as follows: "We must now state how we may ourselves always have a supply of syllogisms in reference to the problem proposed and *by what road we may reach the principles relative to the problem*: for perhaps we ought not only to investigate the construction of syllogisms, but also *to have the power of making them*".

2.4. Impact, criticisms and modern formalized/computational systems

The most important innovation of the syllogistic, which determined the future directions of logic, was the development of the first *deductive system*. Such formal systems are today used as syntactic devices for establishing semantic properties, like the validity of syllogisms. This is a recurring problem in logic. We start with an (artificial) language and a semantics assigning meaning to expressions in the language, such as the logic formulas. Furthermore, we want to establish a semantic property, for example, the truth for a particular interpretation. A deductive system is then defined in the way that Aristotle taught us, i.e. through a set of axioms, all of which are true formulas, and rules of inference through logical forms (patterns with constant and variable parts), which yield true formulas, when given true formulas. A deduction in such a system can be viewed as a tree labelled with formulas, where the axioms are leaves, the inference rules are interior nodes, and the label of the root is the formula, whose truth is established. A series of important questions then arise for such a system: (i) Are the axioms true and is the truth preserved by the inference rules (soundness)? (ii) Can every true formula be deduced (completeness)?

While Aristotle was never disputed as the one who laid the foundations of deductive reasoning, his syllogistic was occasionally criticized, with the best known arguments those by B. Russell [20]. His criticism was ground on the fact that the Aristotelian system permits syllogisms like:

i. All golden mountains are mountains.
ii. All golden mountains are golden.
iii. Some mountains are golden.

This syllogism insinuates the existence of at least one golden mountain! Russell's observation would be correct only within the frame of mathematical logic, where the terms are substituted by mathematical symbols. However, Aristotle's syllogistic is a theory for logos (speech) and the terms are words that symbolize meanings of two different kinds: intensional and extensional. The *extensional meaning* of a symbol consists of the members of the class that the term *denotes*, whereas the *intensional meaning* refers to the qualities or attributes that the term *connotes*. For example, the intensional meaning of the term "cat" consists of the attributes of being furry, of having four legs and so on, while its extensional meaning consists of all the cats in the universe. Thus, the term connotes the attributes and denotes the cats.

Today, it is evident that most of the criticisms against categorical syllogistic are attributed to an extensional interpretation of its terms, which is dominant in the context of mathematic logic[6].

A deeper understanding of the syllogistic was achieved during the second half of the last century, due to the advent of symbolic formalizations[7] [1, 3, 4], which opened a new dimension for the research that now covers problems, such as the soundness and the completeness of Aristotelian logic. In the light of these new insights, we endorse the statement of Lukasiewicz in [1]: "The syllogistic of Aristotle is a system the exactness of which surpasses even the exactness of a mathematical theory, and this is its everlasting merit. But it is a narrow system and cannot be applied to all kinds of reasoning, for instance to mathematical arguments".

Modern studies suggest that "Aristotelian logic should not and cannot be considered as a simple and incomplete forerunner of predicate logic or even set theory" [11]. Aristotle's syllogistic is a formal system of its own, which can be modelled independently of predicate logic and set theory.

There are also recent computational representations [11, 21] of the Aristotelian logic. Computation is also syntactic in nature and closely related to deduction [19], but this relation between deduction and computation is more thoroughly discussed in Section 4.

3. Languages of mathematical logic

[6] Classical philosophers up to Kant and Leibnitz were fully aware of the difference between these two methods of interpretation. Leibniz was the first, who constructed an intensional interpretation only in terms of his characteristic numbers. The first "modern" term logical system of Łukasiewicz [1] and subsequent ones looked at the terms as standing for concepts ("begriffe"), see also in note no. 8.

[7] The theory of syllogisms is a formal theory that was not originally formalized, i.e. it was defined in natural language.

3.1. Ancient truth-functional logic – the birth of mathematical logic by G. Boole

While for Aristotle and the Peripatetics logic was not part of their philosophy, but rather an Organon (tool) of thought, for the Stoics, the logic was part of the third branch of their philosophy and its scope included the analysis of argument forms, rhetoric, grammar, the theories of concepts, propositions, perception, and thought. They made a sharp distinction between logic and language, with the language being considered as utterance, corporeal, material and sensible, and the words seen as natural signs of natural objects. On the other hand, logic was incorporeal and the logical statements, called *lekta*, though they had meaning they were not corporeal[8]. The lekta included assertibles, arguments, syllogisms, and fallacies.

The *truth-functional logic* of the Stoics, is a logic of propositions rather than a logic of terms; the so-called assertibles[9] are the primary bearers of truth-values and the variables stand for assertibles (not for terms). The most fundamental distinction among assertibles is that between simple and non-simple ones.

Every simple assertible is classified in one of the three affirmative and three negative types, and its type is determined by the form of the sentence through which is expressed. The most important negative assertible is the negation[10], which is formed by prefixing to an assertible the negation particle "not". Negation is truth-functional, i.e. the negation particle, if added to true assertibles, makes them false and if added to false ones makes them true. Every negation is the negation of an assertible; an assertible and its negation form a pair of contradictories. The negation particle can be a prefix of simple or non-simple assertibles. The negation of a simple assertible is itself simple, whereas that of a non-simple assertible is non-simple.

Non-simple assertibles can be composed of more than two simple constituent assertibles in two different ways: its constituent assertibles may be themselves non-simple and the used connectives (conjunctive and disjunctive) are conceived as two-or-more-place functors. To avoid ambiguity, the Stoics determine the position of the connectives in the sentence of a non-simple assertible, relative to the sentences of the constituent assertibles; the first word of the assertible is indicative of the type of non-simple assertible it belongs to. For example, the ambiguous statement "*p* and *q* or *r*" where *p*, *q* and *r* are variables becomes either "Both *p* and either *q* or *r*" or "Either both *p* and *q* or *r*". Apart from the negation, only one type of non-simple assertible is truth-functional, the conjunction. In the remaining cases, the adopted truth-criteria are based on combinations of various relations such as incompatibility, symmetry and asymmetry and partial truth-functionality. Chrysippus used only three types of non-simple assertibles, namely conditionals, conjunctions and exclusive disjunctions. The conditional was defined as the assertible that is formed with the linking connective "if". The conditional declares a relation of consequence: a true conditional, if its antecedent is true, has a true consequent.

The truth-functional logic of the Stoics has many similarities with the modern propositional logic, but it is yet another logic for the validity of arguments (like the Aristotelian logic), and it is not a system of logical theorems like the propositional calculus. The Stoics do not assign a special status to logical principles and laws, as Aristotle did for his syllogistic. We know

[8] The Stoics thus made a clear distinction between the significatum (meaning) and the signifier, a distinction that preoccupied the philosophers years later, including Frege.

[9] Assertibles (the original Greek word is «αξίωμα») resemble the propositions of mathematical logic, but differ in that their truth and falsehood are temporal properties.

[10] The Greek word is «αποφατικόν».

however that they have adopted the *law of bivalence*, which states that every proposition has exactly one truth value, either true or false. The law of bivalence is a property of the semantics of the logic language, as opposed to the Aristotelian law of excluded middle, which is a syntactic property (*p* or not-*p*). The classic mathematical logics comply with the Aristotelian laws of excluded middle and non-contradiction, but not all of the studied semantics comply with the bivalence. Also, some Stoics may have dealt with relations like the commutativity, contraposition[11], and equivalences such as the double negation of an assertible, the disjunction of contradictory assertibles and so on.

An argument is defined as a compound of premises and a conclusion, all of which are assertibles (component assertibles):

i. If it is day, it is light.
ii. But it is day.
iii. Therefore, it is light.

The first mentioned premise (leading) is a non-simple assertible and the second one (co-assumption) may be a simple assertible or a non-simple assertible with fewer constituent assertibles. Validity is judged based on the conditional formed with the conjunction of the premises as antecedent and the conclusion as consequent: if the assertible "If (both p_1 and p_2 and . . . p_n), then *c*" is true, with p_1, p_2, . . ., p_n the premises and *c* the conclusion, then the argument "p_1; p_2; . . .; p_n; therefore *c*" is valid. However, the criterion for the correctness of the conditional was the one by Chrysippus: the contradictory of the conclusion should be "incompatible" with the conjunction of the premises. However, this criterion is not necessarily restricted to our conception of logical consequence that should be necessary and formal[12]. The Stoics also assumed that arguments have a truth value: an argument is true, which corresponds to our notion of soundness, if it is valid and has true premises. On the other hand, an argument is false if it is invalid or has a false premise.

Valid arguments are distinguished between syllogisms and the ones that are not syllogisms (valid in the specific sense). The syllogisms include the indemonstrable arguments and those that can be reduced to indemonstrable arguments. The indemonstrable arguments,

- if *p* then *q*; *p*; therefore *q* (modus ponens)
- if *p* then *q*; not-*q*; therefore not-*p* (modus tollens)
- it is not both *p* and *q*; *p*; therefore not-*q*
- either *p* or *q*; *p*; therefore not-*q*
- either *p* or *q*; not-*p*; therefore *q*

need a proof or demonstration, because their validity of profound. Indemonstrable syllogisms should be reducible through the use of certain ground rules (themata), to the five indemonstrable forms. This is essentially a method of proving the formal validity of arguments by showing how they stand in a certain relation to the indemonstrable syllogisms, i.e. such an argument may be a composite of several indemonstrables, or a conversion of an indemonstrable or a mixture of both. The themata were four rules used to reduce arguments to other arguments and they were regarded as sufficient for the analysis of all non-

[11] The contrapositive of a conditional assertible "if *p* then *q*" is equivalent to the assertible that has its antecedent and consequent inverted and flipped, i.e. the "if not- *q* then not-*p*".
[12] A precise account of logical consequence with respect to a language is given either by constructing a deductive system for this language, or by formalizing its intended semantics.

indemonstrable syllogisms (some theorems were also used as a means to facilitate the analysis).

An accurate, detailed description of the Stoics' truth-functional logic can be found in [22].

The Stoics were the first who introduced logic reasoning over the language by using rules for reasoning with the logical connectives "not", "and", "or" and "if . . . then". Their approach was to avoid the inherent ambiguity in the language, through determining the position of the connectives relative to the sentences that express the used assertibles. Differences in syntax correspond to differences in meaning. This is basically similar to the way of reasoning in today's propositional calculus, where we use instead a well-chosen symbolic representation of the language.

In a symbolic language, well-formed-formulas are treated purely as symbolic expressions having no meanings, which are interpretable as meaningful statements in natural language sentences that involve truth-functional logic connectives. These interpreted statements have an ordinary *thruth-value* (True or False), which can be transferred back to their symbolic well-formed-formulas. Formal systems of propositional logic such as those in [16, 23] have certain axioms, which are well-formed-formulas expressing logical forms that are true under any natural interpretation. These axioms are applied in the steps of formal proofs, in which the meanings of the statements are not involved, like the proofs performed by Aristotle in his syllogistic, and by Euclid in the Elements.

Such a formalized representation is also the key to mechanize computation. Let us consider the propositional variables *p*, *q* and so on, denoting arbitrary declarative natural language sentences, which are either true or false. Syntactically, variables *p*, *q* serve as atomic propositions, i.e. they are the basic building blocks of propositional well-formed-formulas. To compute the truth-value of a compound proposition, when given truth values for the interpreted sentences assigned to the propositional variables in the formulas, it is only necessary to know how the connectives determine truth-values. This has been encoded by natural *truth-tables* for the connectives as follows (T stands for truth and F for falsity):

p	q	p or q	p and q	not-p	if p then q[13]
T	T	T	T	F	T
T	F	T	F	F	F
F	T	T	F	T	T
F	F	F	F	T	T

In 1854, G. Boole's introduced [14] an abstract algebra of propositions, where the logical connectives behave like algebraic operators. He considered logic reasoning by dealing with subsets of some given set. In this type of reasoning, the union of two subsets *X* and *Y*, denoted by $X \cup Y$ arose for a proposition *p* or *q*, from now on denoted as $p \vee q$, where *p* is true for elements in *X* and *q* for elements in *Y*. Then, $p \vee q$ is true for elements in either *X* or *Y* or both, that is, in $X \cup Y$. Likewise, *p* and *q*, from now on denoted as $p \wedge q$ is true for elements in $X \cap Y$ and not-*p*, denoted as $\neg p$, is true for elements in $\bar{X}$ (the complement of *X*). The proposition "if *p* then *q*", from now will be written as $p \rightarrow q$. Boole used 1 for true and 0 for false. If *p* and

[13] Notice that the proposition "if p then q" can be written as not-p or q (recall the correctness criterion by Chrysippus).

q represent propositional variables, they can be either true or false in a propositional logic system and they are therefore assigned one of the values 0 or 1.

For a compound propositional well-formed formula *w* with *n* propositional variables, *w* would take different truth values, for various truth values assigned to the *n* variables. In effect, an arbitrary well-formed-formula *w* in propositional logic defines a *function* (called Boolean function), when we restrict the values of the propositional variables in *w* to be either 0 or 1, since we are only interested in the truth values of *w*. Thus, Boole effectively showed that the computation of truth values for an arbitrary well-formed-formula can be done within an abstract algebraic setting involving two *binary* operations, + and $\cdot$, corresponding to the $\vee$ and $\wedge$ logical connectives and a unary operation $'$ corresponding to $\neg$. Thus, we have an abstract Boolean algebra consisting of an abstract set *B* with binary operations + and $\cdot$ and a unary operation, which satisfy the following axioms:

a. There are elements 0 and 1 in B, such that x + 0 = x and x·1 = x
b. x + y = y + x and x·y = y·x
c. x+(x·y)=x and x·(x+y)=x
d. x·(y+z) = x·y + x·z and x+(y·z) = (x+y)·(x+z)
e. For any x in B there is an element x′, such that x+x′=1 and x·x′=0

An example of a Boolean algebra is the set $B = \{0, 1\}$ with the Boolean + and · operations defined as the usual arithmetic operations except that 1+1=1, 0′=1 and 1′=0. Another example is the set of all subsets of a given set *X*, with the + operation being the union, the · operation being the intersection, and x′ being the complement of x. In this case, 1 is *X* and 0 is the empty set.

It happens that our modern computational devices consist of circuit designs that involve *n* "input" components x(1), x(2), . . ., x(n), which can be in either two states, i.e. "on" corresponding to 1 or "off" corresponding to 0. These components are connected using complex combinations of OR, AND and NOT gate elements, so that there is a single circuit output *f*(x(1), . . ., x(n)), which is either "on", meaning 1, or "off" meaning 0, as required for the specified circuit operation. In effect, every such circuit computes a specified Boolean function *f*. For a Boolean function corresponding to some given well-formed propositional formula, it may be possible to apply a series to algebraic simplifications according to the mentioned Boolean axioms, in order to speed-up the computation of its value.

How is Boole's logic compared with Aristotle's logic? Certainly, Boole's work is the one that begins mathematical logic, but it is not the beginning of logical theory. Aristotle's theory was very well known to and admired by Boole. In [14], Boole explicitly accepted Aristotle's logic as "a collection of scientific truths" and he regarded himself as following in Aristotle's footsteps. The categorical propositions of syllogisms can be written in Boolean logic as follows:

All *X* are *Y*. $X \cdot Y = X$

Some *X* are *Y*. $X \cdot Y \neq 0$

However, as we already mentioned in Section 2.4, this results into an exclusively extensional interpretation of the stated propositions and their intensional semantics is lost. Boole thought [10] that he was supplying a unifying foundation for Aristotle's logic and that he was at the same time expanding the ranges of propositions and of deductions that were formally treatable in logic.

In [24], the author summarizes in only one statement the essence of the contributions by Aristotle and Boole in logic: ". . . using mathematical methods . . . has led to more knowledge about logic in one century than had been obtained from the death of Aristotle up to . . . when Boole's masterpiece was published".

3.2. Predicate logic and the decision problem

Atomic propositions in propositional logic are not considered having internal structure. Let us consider for example the statement "Aristotle writes", denoted by *p*, and the statement "Aristotle walks" denoted by *q*. Here the information that both statements are about Aristotle is already lost, because propositional logic cannot reflect the internal structure of such basic facts. In predicate logic, the "Aristotle writes" is expressed as *S j* and the "Aristotle walks" as *P j*, effectively having two properties for the same person, named by the constant *j*. Predicate logic is a universal logic language capable to express any situation with objects, properties and relations found in our daily life or even in mathematics. Moreover, predicate logic can represent universal quantification (all, every, each, . . .) and existential quantification as well. These features make the language look like the symbolic languages used in mathematics. The predicate logic that is widely used today is a streamlined version of a "language of thought" that was proposed in 1878 by Gottlob Frege in his "Begriffsschrift" [14].

Predicate logic has a precisely defined vocabulary and syntax. We refer to *objects* using *constants*, for example *a*, *b*, *c*, . . . and *variables*, such as *x*, *y*, *z*, . . ., when the object is indefinite. It is also possible to use *function* symbols for complex objects. For being able to define *properties* and *predicates* of objects, let us use capital letters for predicates that may *relate* different numbers of objects. Thus, we can use *unary*, *binary* or *ternary* predicates depending on the number of objects hey relate. The language also incorporates the propositional connectives $\wedge$, $\vee$, $\neg$ and $\rightarrow$ and has a very expressive way to specify quantification. The two common quantifiers from the mathematics,

- $\forall x$, meaning for all *x*
- $\exists x$, meaning there exists an *x*

are tagged by variables for objects.

Let us have a look now to a concrete example showing the use of predicate logic to express natural language statements:

"Every student is younger than some instructor."

We define the following predicates:

S x: *x* is a student

I x: *x* is an instructor

Y x y: *x* is younger than *y*

Now the natural language statement is written as:

$$\forall x\,(Sx \rightarrow (\exists y\,(Iy \wedge Yxy)))$$

In mathematics, we usually use binary predicates like $<$ ("is smaller than"), $\in$ ("is an element of"), but we may also need ternary predicates in order to express facts e.g. in geometry, such as "*x* lies between *y* and *z*". It is also usual to write the predicates in between their arguments, e.g. $7 < 13$. The names of binary relations in mathematics are the only exception to the standard in predicate logic, which is writing the predicate first and then the objects. The binary predicate = ("is equal to") expresses *identity* and it is used to represent uniqueness properties of objects. Thus, the following formula states that there is exactly one object which has the property *P*:

$$\exists x\,(Px \wedge \forall y\,(Py \rightarrow y = x))$$

In natural language, this statement says that there exists a property "*P*" such that each second object which has the property *P* as well must be equal to the first. An important expansion in the predicate logic syntax and semantics is the additions of function symbols, which allow to refer to functions, such as "the sum of *x* and *y*", "the cosine of x" and so on. Mathematical functions create new objects out of old ones and thus the "the sum of *x* and *y*" is a totally new object created out of the objects for the variables *x* and *y*. In mathematics, function symbols are combined with equality to express algebraic equations like for example:

$$x\cdot(z+2) = x\cdot z + x\cdot 2$$

This statement can be viewed as a formula of predicate logic based on the equality predicate and the function symbols for addition and multiplication. However, even this formula still does not have a meaning, because we have not yet connected the used symbols to their intended meaning. Since the term "2" is a constant, it can name a number from either the domain of integers or the domain of natural numbers.

In order to be able to assign a meaning, we have to define how our predicate logic language is interpreted systematically on a semantic structure. A model $\boldsymbol{M}$ is a pair (*D*, *I*) with a domain *D* consisting of individual objects, together with an interpretation *I* that assigns the right kind of predicates on objects in *D*. Thus, if *R* is a binary predicate, then:

$$I(R) \subseteq D \times D$$

For the mentioned mathematical formula in predicate logic, all the terms can be interpreted either to the set of integers or to the set of natural numbers (positive integers and 0).

Now, we have a very expressive language capable to represent statements for our daily life or even for mathematics. In this language, the terms of the Aristotelian categorical statements are interpreted as "sets of individuals", thus representing an extensional semantics according to which the terms necessarily refer to nonempty sets. More precisely, the four categorical propositions are now written as follows:

All *A* are *B*.	$\forall x\,(Ax \rightarrow Bx)$
No *A* are *B*.	$\neg\exists\, x\,(Ax \wedge Bx)$
Some *A* are *B*.	$\exists\, x\,(Ax \wedge Bx)$
Some A are not B.	$\neg\forall x\,(Ax \rightarrow Bx)$

Thus, the language of syllogisms is only a small fragment of predicate logic that imposes a restriction on the form of the predicates that are allowed: they have to be unary predicates.

This special system with only unary predicates is called *monadic predicate logic*. However, this transcription of Aristotelian syntax does not exactly reproduce the Aristotelian theorems. For example, there is no proof system, in which the immediate inference of *subalternation* from the square of opposition, $\forall x\,(Ax \rightarrow Bx) \nvdash \exists x\colon Ax \wedge Bx$, can be proved[14].

David Hilbert and Wilhelm Ackermann introduced in [16] a formal system $\mathfrak{K}$ with axioms and inference rules for proving predicate logic well-formed-formulas, in the same manner as in Aristotle's syllogistic[15]. Let us first focus on the axioms, which are true formulas in any possible interpretation (U, V, W can be any well-formed formulas, U x and V x are formulas with a variable x and α is any constant in some domain):

a. $(U \rightarrow (V \rightarrow U))$
b. $(U \rightarrow (V \rightarrow W)) \rightarrow ((U \rightarrow V) \rightarrow (U \rightarrow W))$
c. $(\neg V \rightarrow \neg U) \rightarrow (U \rightarrow V)$
d. $\forall x\ U\ x \rightarrow U\ \alpha$
e. $\forall x\ (U \rightarrow Vx) \rightarrow (U \rightarrow \forall x\ Vx)$

A proof consists of steps, in which one of the axioms or any of the following inference rules can be applied to the previously shown well-formed-formulas:

i. The modus ponens, which allows a well-formed-formula *V* to be proved if a well-formed-formula *U* and the well-formed-formula $U \rightarrow V$ have been already proved.
ii. The generalization rule, which allows some formula U α to be generalized to the $\forall x\ Ux$.

The modus ponens preserves the truth of earlier steps and thus, every proof starting with a purely logical axiom can only show well-formed-formulas, which are valid, i.e. true *for all possible interpretations* (soundness).

For the completeness of $\mathfrak{K}$, it was essential to prove that any well-formed-formula, which is valid can be proved (Figure 1). In this case, formal proofs of valid well-formed-formulas would be a mechanical manipulation of symbols. In 1929, Kurt Gödel showed in his doctoral thesis that $\mathfrak{K}$ is complete.

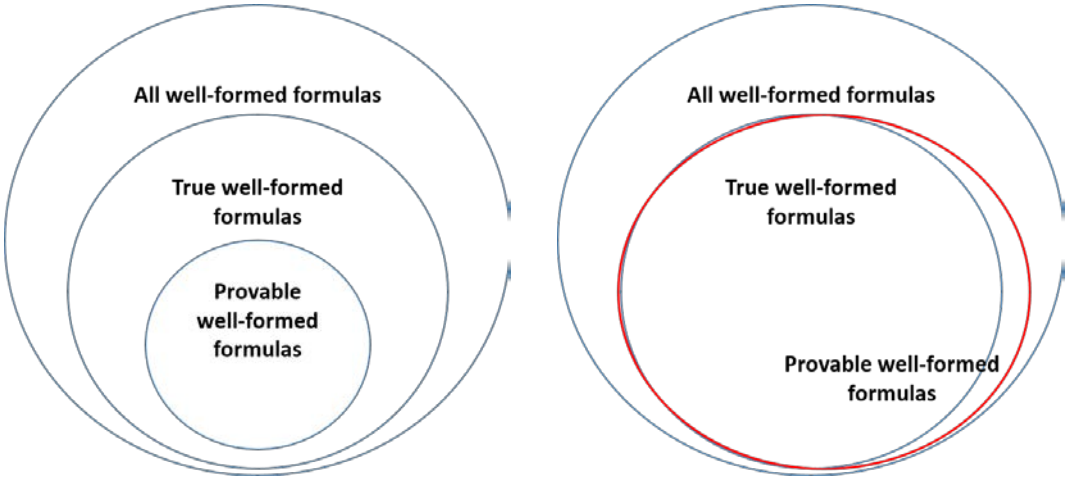
However, Hilbert had an even more ambitious goal: he envisioned a formal system with appropriate axioms, in which *all the theorems of mathematics* would be possible to be proved. To prove well-formed-formulas, which are interpreted as mathematical statements, the formal system would also have to provide axioms of a mathematical type. For the mathematics of natural numbers, $\mathfrak{K}$ is used along with variables that can be assigned values in the domain $\mathbb{N}$. It is also essential to adjoin to $\mathfrak{K}$ well-formed-formulas representing axioms about $\mathbb{N}$, which can be the Peano axioms that employ the constant 0, the successor function,

[14] In [1] and [3], as well as in Smiley's term logic [4], the variables stand for Aristotelian terms like man, animal, soul, etc. Both interpretations are therefore allowed:

- Extensional: the "All A are B" is true if and only if the set representing all the instances of the concept A (extension of A) is a subset of the extension of B, i.e. the more general a concept is, the bigger is its extension.
- Intensional: terms are mapped onto sets of intensions or meanings contained in A and the truth of the "All A are B" is that A has more meaning (more intensions) than B, because it is more specific, i.e. the intensional content grows while proceeding to more specialized concepts, which is contrary to the extensional interpretation.

[15] In [23], Russell and Whitehead have introduced a formal system with five propositional axioms.

addition, multiplication and equality. We denote by $\mathcal{K}(\mathbb{N})$ the extended formal system, for the set of natural numbers $\mathbb{N}$.



Incomplete formal system

Inconsistent formal system: there is false well-formed formula with a valid proof

Figure 1 Inconsistent and incomplete formal systems

Hilbert's belief was that every well-formed-formula in $\mathcal{K}(\mathbb{N})$, which is interpretable in $\mathbb{N}$ as a true statement about $\mathbb{N}$ is formally provable in $\mathcal{K}(\mathbb{N})$. This would make $\mathcal{K}(\mathbb{N})$ complete. If $\mathcal{K}(\mathbb{N})$ is consistent, then for a provable formula U the $\neg$U should not be true and therefore not provable (Figure 1). If so, Gödel showed that $\mathcal{K}(\mathbb{N})$ is not complete through a well-formed-formula U that is true in $\mathbb{N}$, but its representation $\mathcal{K}$(U) is not provable in $\mathcal{K}(\mathbb{N})$. Furthermore, he also showed that the incompleteness of $\mathcal{K}(\mathbb{N})$ cannot be remedied, if for example more axioms would be adjoined.

The established incompleteness of $\mathcal{K}(\mathbb{N})$ did not preclude the possibility to find an effective decision procedure, which could decide for any well-formed-formula interpreted in $\mathbb{N}$ as a mathematical statement, if it is provable in $\mathcal{K}(\mathbb{N})$. This is the so-called decision problem ("Entscheidungsproblem") that was posed by David Hilbert in 1928. Such a decision procedure would tell whether a mathematical statement is formally provable or not, in which case, he should try to prove the denial of this statement. To guarantee an answer to the decision problem, the decision procedure should halt on all possible inputs in a finite number of steps.

4. The Theory of Computation

Gödel's theorems on the incompleteness of $\mathcal{K}(\mathbb{N})$ showed the limitations of formal systems. However, if a solution would exist for the decision problem, it should be possible to decide by some computation, if any well-formed-formula in $\mathcal{K}(\mathbb{N})$ or the negation of it can be formally proved. Computation suggests a purely mechanical transformation procedure to a sequence of symbols, but at that time, there was no definition of the exact transformations that a human computer or perhaps a machine are allowed to apply.

In 1936, Alonzo Church developed a definition of a concept that he called *effectively calculable function* or *effective procedure* [25]. Church's aim was to represent mathematical functions by well-formed-formulas called lambda expressions and to define rules for manipulating such well-formed-formulas in a manner corresponding to the calculations found in informal

mathematics. His formal system called *lambda calculus* would have to include all mathematical functions that can be calculated by obviously executable means. The λ-definable functions turned out to coincide with known classes of recursive functions. This convinced Church conjecture that λ-definability captures the concept of "effective calculability". Thus, Church restricted the manipulations on function well-formed-formulas to "effective calculations" on sequences of symbols representing functions. In this manner, an arbitrary effectively calculable function could be expressed as a well-formed-formula and it was hoped that an effective procedure compliant with the Hilbert's decision problem, would exist for the manipulation of such formulas. At the end, Church proved that the decision problem is unsolvable, i.e. there is no such procedure, which could decide if an arbitrary function is formally provable.

In a parallel and independent research work on the same problem [26], Alan Turing introduced later in 1936 his own theoretical machine model of a procedure, which is general enough for studying the decision problem. With what it is now called Turing Machine, he confirmed Church's finding that the decision problem is unsolvable. Turing then proved that his machines and λ-calculus procedures are equivalent [27], which means that the existence of one of them for any purpose implies also the existence of the other.

These two results produced the so-called *Church – Turing thesis*: each of the two exact definitions for a decision procedure can faithfully represent any effective computation.

The unsolvability of the decision problem ended the attempted formalization of mathematics by Hilbert and was an affirmation for those who insisted in the importance of intuitive mathematics. But what exactly is this that renders a formal deductive system inadequate, so that the problem cannot be solved by the type of computations defined by Church and Turing? The answer lies in a careful distinction between the syntactic aspects of the formal system $\mathcal{H}(\mathbb{N})$ and the semantics, which has been associated with its interpretation. The computations have also syntactic nature, but apart from what exactly are the allowed transformations, we are interested additionally in how they may be executed, i.e. how they are programmed in the chosen model of computation. It is the interpretation process that is not subject to the Church-Turing thesis. Today, it is well understood that the most problematic issue in defining an interpretation that would entail some computable constructions for deductive proofs, is that of specifying the *semantics of truth*[16]. In a systematic treatment of this issue, in [19], it is easily shown that a well-formed-formula in the predicate logic can be neither true nor false, for a given interpretation.

At the same time, with the first general models of computation, a new discipline was born, the Theory of Computation. Through the commonly agreed formalized definitions of a decision procedure or algorithm, the decidability question would also deserve an answer for other problems. Is there an algorithm that produces the correct output for every possible problem instance, in a finite number of steps (i.e. amount of time)? A problem is called *computable*, if there is an algorithm that solves it. Such an algorithm must always terminate and must always produce the correct output, for all possible inputs to the problem. If no such algorithm exists, the problem is called *non-computable*.

[16] In propositional logic, the semantics of truth is encoded in truth tables and the decision problem has a solution.

Alan Turing used his theoretical machine model to develop a general method for proving that a problem is non-computable, without having to enumerate all possible, potentially infinite, candidate solutions. The Turing Machine also became the reference model used to characterize and compare the computational efficiency of algorithms. Today, the Theory of Computation develops an ever growing number of results on the computability and the computational efficiency of solutions, for important problems in science.

Computability and computational solutions also became a central issue in the study of logic systems. Canonical forms suitable for computation have been introduced for propositional and predicate logic formulas and an automated inference rule has been invented that leads to refutation proofs (reductio ad absurdum), the so-called resolution (see also Section 5). These developments have opened new prospects for efficient procedures in automated theorem proving.

5. Computational logic

A typed λ-calculus variant was introduced by Church in [28], to restrict the computational expressiveness of the original λ-calculus, where expressions could even be applied to themselves. Types are objects of a syntactic nature that are assigned to lambda expressions. The typed λ-calculus turned out to evolve into a foundational programming language and it is still playing an important role in the design of type systems for programming languages, where types capture necessary properties for the programs. From 1958 and beyond, Haskell Curry and later William Alvin Howard discovered a syntactic analogy between systems of formal logic and computational calculi. When abstracting from the peculiarities of the different formalisms, we can see a proof as a program and the formula, which is proved, as a type for the program. This sets a rigorous foundation of logic programming: proofs can be represented as programs and in particular as typed lambda expressions. The procedure of checking a proof is like type-checking an expression, which is a computable problem in the type systems of most programming languages.

The Computational Logic, which is also known as "logic in computer science", refers to the use of logic to *perform* or *reason about computation*. It bears a similar relationship to computer science as the mathematical logic bears to mathematics and as the philosophical logic bears to philosophy.

In 1956, the Logic Theorist program by Allen Newell, J.C. Shaw, and Herbert Simon [29, 30] was considered as "the first artificial intelligence program" designed to mimic the problem solving skills of a human being. It was an effort to validate the theoretical work in “Principia Mathematica” by Bertrand Russell and Alfred North Whitehead [23], once it was realized that deductive reasoning could be aided algorithmically. For example, validity in monadic predicate logic, where it is possible to perform syllogistic reasoning with unary predicates, is computable. However, as we saw in Section 4, this is no longer true for predicate logic as a whole.

“Computational Logic” is often associated with a programming paradigm based on formal logic known as “logic programming”. It was introduced by Robert Kowalski in the early 1970s [31], who suggested using of sentences in logical form to express facts and rules about some problem domain. The rules have the form

$$H \leftarrow B1, \ldots, Bn$$

which represents the logical consequence “If B1 and . . . and Bn are true, then so is H”. The programs in the various logic programming languages may have:

- a declarative meaning, executed through a proof procedure or a model generator (not controlled by the programmer)
- a procedural meaning, as goal - reduction procedures, i.e. “In order to solve H, solve B1, and . . . and solve Bn.”

Logic programming is based on a rule of inference known as *resolution*, which leads to a refutation theorem - proving technique by Davis and Putman [32] and John Alan Robinson [33], for propositional and predicate logic formulae. The satisfiability of a propositional and the unsatisfiability of predicate logic formulas can be checked through an iterative application of the resolution rule in an appropriate manner. However, if an unsatisfiability proof is attempted for a satisfiable predicate logic formula, this may result in a non-terminating computation that does not happen for propositional logic formulae.

In a series of research initiatives started with Ron Brachman [34] the idea was the knowledge representation in a subset of predicate logic (undecidable) through an appropriate trade-off between expressivity and computability. The more expressive is the language (i.e. closer to the predicate logic), the more likely is to be prone to infinite computations. Thus, predicate logic could be used as a metric of comparison for all artificial intelligence knowledge representation formalisms.

The Horn Logic [35] is a subset of predicate logic with logical consequences consisting of at most one conclusion, where no negation is allowed in the rules' conditions and all existentially quantified variables have been eliminated through the Skolemization procedure [36]. This language was found to be decidable using *Modus Ponens* as the sole inference procedure, when there are finitely many objects in the universe of discourse and the terms do not contain function symbols. Horn Logic is the basis of logic programming.

Another interesting subset of predicate logic are the Description Logics (DL). The core reasoning problems of DLs are (usually) decidable and there are efficient decision procedures, like the analytic tableau method, with exponential worst case complexity though. DLs are used to describe and reason about the relevant concepts of an application domain, known as terminological knowledge and they provide a formal framework for ontologies and the Semantic Web (in particular for the Web Ontology Language OWL).

6. Conclusion

We reviewed the historical origins of logic, starting with the theory of Aristotle about 2400 years ago, in order to identify the fundamental principles that affected the scientific thought in mathematics and computing. The symbolic languages currently used in the theory of logic and computation are definitely different from the Aristotle’s language of categorical propositions. However, the reasoning based on logical forms that Aristotle first introduced is an approach that we still use in modern formal systems. We described the syllogistic theory by Aristotle with an emphasis in his formal system for deductive reasoning.

The Stoics, in their effort to also prove an argument's validity or not, developed a formalistic inference approach for propositional statements in natural language sentences. These results, together with the symbolic representation of logic languages led the developments towards mechanizing the computation of logic expressions. A key contribution in this was the logic algebra by G. Boole.

In the quest of a suitable predicate logic for mathematical reasoning, G. Frege proposed a more expressive transcription of Aristotle's categorical propositions by means of quantification on individuals, and introduced precise syntax rules for all combinations of quantifiers and negation, as well as conjunction, disjunction and conditionals. New systems of deductive reasoning were developed by D. Hilbert and B. Russell, based on the principles of the first formal system by Aristotle: every proof step should preserve the truth of the previously shown formulas, thus yielding valid results. This is achieved by using a set of axioms (true formulas) together with formation rules stating the kind of symbol sequences that are well-formed, and transformation rules for logical inference. However, Gödel showed that not every well-formed-formula of predicate logic interpreted as mathematical statement is provable.

Hilbert then posed the problem of a decision procedure, to compute whether any mathematical statement is formally provable or not. A rigorous definition of the still vague notion of computation was needed to find the answer. Computation suggests a purely "mechanical" and "effective" transformation procedure applied on a sequence of symbols.

The answer to the decision problem was negative, but the development of the first formal models of computation by A. Turing and Alonzo Church founded a new discipline, the Theory of Computation. We discussed the differences between deduction and computation, and how they are related with each other. The computations do not only specify what exactly the allowed transformations are, but additionally, how they may be executed. Models of computation, like the λ-calculus by Alonzo Church, are also defined similar to Aristotle's formal deductive system, but they are more expressive.

With the latest developments in the area of computational logic, it became possible to develop automated logic inference and new programming paradigms, for specific fragments of the predicate logic. It has been found, for example, that the monadic predicate logic, in which syllogistic reasoning is performed, is computable. In another research direction, a renewed interest has been observed during the last decades, for symbolic formalizations and computational implementations of the Aristotelian logic independently of predicate logic. The main motivation for these new approaches is to reflect also the intensional semantics of the ancient logic.

The theory of syllogisms continues to drive the developments, in a way that Aristotle certainly would not have been imagined.

References

[1] Jan Lukasiewicz, *Aristotle's syllogistic from the standpoint of modern formal logic*, Clarendon Press, Oxford, 1957

[2] Jonathan Lear, *Aristotle and logical theory*, Cambridge University Press, 1980

[3] John Corcoran, “Completeness of an ancient logic”, *Journal of Symbolic Logic*, 37 (4), 1973, 696-702

[4] T. Smiley, “What is a Syllogism?”, *Journal of Philosophical Logic*, 2, 1973, 136-154

[5] John Corcoran, “Aristotle’s natural deduction system”, In: *Ancient Logic and Its Modern Interpretations* – J. Corcoran (ed.), 85-131, 1974

[6] B. H. Slater, “Aristotle’s propositional logic”, *Philosophical Studies*, 36, 1979, 35-49

[7] Allan Back, “Syllogisms with reduplication in Aristotle”, *Notre Dame Journal of Formal Logic*, 23 (4), 1982, 453-458

[8] Robin Smith, “Predication and deduction in Aristotle: aspirations to completeness”, *Topoi*, 10, 1991, 43-52

[9] Alex Orenstein, “Reconciling Aristotle and Frege”, *Notre Dame Journal of Formal Logic*, 40 (3), 1999, 391-413

[10] John Corcoran, “Aristotle’s Prior Analytics and Boole’s laws of thought”, *History and Philosophy of Logic*, 24, 2003, 261-288

[11] Klaus Glashoff, “Aristotelian syntax from a computational – combinatorial point of view”, *Journal of Logic and Computation*, 15 (6), 2005, 949-973

[12] John Corcoran, “Notes on the Founding of Logics and Metalogic: Aristotle, Boole, and Tarski”, In: *Current Topics in Logic and Analytic Philosophy* – C. Martínez et al. (ed.), 145-178, 2007

[13] Leroy Loemker (ed. and trans.), *Gottfried Wilhelm Leibniz: Philosophical Papers and Letters*, 1969

[14] George Boole (1854), *An Investigation of the Laws of Thought*, Prometheus Books, 2003

[15] Gottlob Frege, *Begriffsschrift: eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*, Halle, 1879

[16] David Hilbert and Wilhelm Ackermann (1928), *Principles of Mathematical Logic*, American Mathematical Society, 1950

[17] M. Davis, “Influences of mathematical logic on computer science”, In: *A half-century survey on The Universal Turing Machine*, R. Herken (Ed.), Oxford University Press, Inc., New York, NY, USA, 315-326, 1988

[18] Aaron R. Bradley, Zohar Manna, *The Calculus of Computation*, Springer, 2007

[19] R. E. Davis, *Truth, Deduction, and Computation – Logic and Semantics for Computer Science*, Computer Science Press, 1989

[20] Bertrand Russell, *History of Western Philosophy*, George Allen & Unwin, 1946

[21] Camilo Rocha, José Meseguer, “A Rewriting Decision Procedure for Dijkstra - Scholten’s Syllogistic Logic with Complements”, 2do Congreso Colombiano de Computación (2CCC), Bogotá, 2007.

[22] S. Bobzien, “Logic”, In: *The Cambridge Companion to the Stoics*, B. Inwood (Ed.), Cambridge University Press, 85-123, 2002

[23] Alfred North Whitehead, Bertrand Russell, *Principia mathematica 3*, 2nd ed., Cambridge: Cambridge University Press, 1927

[24] P. Rosenbloom, *Elements of Mathematical Logic*, NY: Dover, 1950

[25] Alonzo Church, "A note on the Entscheidungsproblem", *Journal of Symbolic Logic*, 1, 1936, 40–41

[26] A. M. Turing, “On computable numbers, with an application to the Entscheidungsproblem”, *Proceedings of the London Mathematical Society*, 42, 230–265, 1936

[27] A. M. Turing, "Computability and λ-definability", *Journal of Symbolic Logic*, 2, 1937, 153–163

[28] Alonzo Church, "A formulation of the simple theory of types", *Journal of Symbolic Logic*, 5, 1934, 56–68

[29] Allen Newell, Herbert A. Simon, *The Logic Theory Machine: A Complex Information Processing System*, P-868, The RAND Corporation, Santa Monica, California, June 15, 1956.

Available at: http://shelf1.library.cmu.edu/IMLS/MindModels/logictheorymachine.pdf

[30] Allen Newell, J.C. Shaw, H.C. Simon, "Empirical explorations with the logic theory machine", In: *Computers and Thought*, E. A. Feigenbaum, J. Feldman (Ed.), McGraw Hill, 109–133, 1963

[31] Robert Kowalski, *Predicate Logic as a Programming Language Memo 70*, Department of Artificial Intelligence, Edinburgh University, 1973

Also in Proceedings IFIP Congress, Stockholm, North Holland Publishing Co., 569–574, 1974

[32] Martin Davis, Hilary Putnam, "A Computing Procedure for Quantification Theory", *Journal of the ACM*, 7 (3), 1960, 201–215

[33] J. A. Robinson, "A Machine-Oriented Logic Based on the Resolution Principle", *Journal of the ACM*, 12 (1), 1965, 23–41

[34] Hector Levesque, Ronald Brachman, "A Fundamental Tradeoff in Knowledge Representation and Reasoning", In: *Reading in Knowledge Representation*, R. Brachman and H. J. Levesque (Ed.), Morgan Kaufmann, p. 49, 1985

[35] Alfred Horn, "On sentences which are true of direct unions of algebras", *Journal of Symbolic Logic*, 16 (1), 1951, 14–21

[36] Alex Sakharov, "Skolem Function", From MathWorld -- A Wolfram Web Resource, created by Eric W. Weisstein, http://mathworld.wolfram.com/SkolemFunction.html