RACSAM
Rev. R. Acad. Cien. Serie A. Mat.
VOL. **101** (2), 2007, pp. 133–157
Ciencias de la Computación / Computational Sciences
Artículo panorámico / Survey

What Machines Can and Cannot Do*

Luis M. Laita, Eugenio Roanes-Lozano and Luis de Ledesma Otamendi

Abstract. In this paper, the questions of what machines cannot do and what they can do will be treated by examining the ideas and results of eminent mathematicians.

Regarding the question of what machines cannot do, we will rely on the results obtained by the mathematicians Alan Turing and Kurt Gödel. Turing machines, their purpose of defining an exact definition of computation and the relevance of Church-Turing thesis in the theory of computability will be treated in detail. The undecidability of the "Entscheidungsproblem" (German for "decision problem") is shown to be a consequence of the computations that Turing machines can or cannot do.

Beginning with Peano's arithmetic, a variant of it, "theory S", is presented and discussed. By studying representability and expressibility in S, the notion of recursive functions will be reached.

Gödel arithmetization of logic and of first order theories and his completeness and incompleteness theorems, together with Church's incompleteness theorem, provide both the important possibility of codifying mathematics and the reasons for existence of other undecidable problems

Regarding the question of what machines can do, we mainly rely on the ideas of the Nobel Laureate Herbert A. Simon. Nevertheless, a few facts about modern computing machines and about the so called "expert systems" will be described, because they suggest the existence of important capabilities of machines.

The paper ends with a few considerations about the question: who is more intelligent, the man or the machine?

Lo que las máquinas pueden y no pueden hacer

Resumen. En el presente artículo, la cuestión de lo que las máquinas pueden y no pueden hacer será considerada basando la argumentación en ideas y resultados de eminentes matemáticos e informáticos. En la cuestión ¿qué no pueden hacer las máquinas?, nos basamos en resultados obtenidos por Alan Turing y Kurt Gödel. Las máquinas de Turing, su propósito de mostrar una definición exacta de lo que significa "computación" y/o "algoritmo", conjuntamente con la importancia que tuvo y sigue teniendo en la teoría de la computabilidad la tesis de Church-Turing, serán tratadas detalladamente. Se muestra que la indecidibilidad del problema de la decisión ("Entscheidungsproblem") es una de las consecuencias de lo que las máquinas de Turing pueden o, mejor, no pueden hacer. Se presenta y discute una variante de la aritmética de Peano, denominada "Teoría S", de forma que definiendo en S las nociones de "representabilidad" y "expresabilidad", llegaremos a la noción de "función recursiva". La aritmetización de Gödel de la lógica y de las teorías de primer orden, y sus dos teoremas de incompletud, conjuntamente

*This paper is based on a lecture delivered by the first author at the International Symposium *Mathematics for the XXIth Century*, organized in Madrid, Spain, on May 3–4, 2006, by the Ramón Areces Foundation and the Royal Academy of Sciences of Madrid, together with the Executive Committee of the International Congress of Mathematicians 2006 (ICM2006 Madrid). The second and third authors have improved relevant parts of the lecture.

Presentado por Manuel de León y Manuel López Pellicer.
Recibido: 18 de marzo de 2007. *Aceptado*: 10 deoctubre de 2007.
Palabras clave / Keywords: Artificial Intelligence Surveys, Applications of Computability and Recursion Theory
Mathematics Subject Classifications: 68T02 and 03D80

con el teorema de incompletud de Church, hacen posible, por un lado, la codificación en matemáticas, y, por otro, mostrar la existencia de problemas indecidibles adicionales. Con respecto a la cuestión de ¿qué pueden hacer las máquinas? nos basamos fundamentalmente en las ideas sobre inteligencia artificial del Premio Nobel Herbert A. Simon. Sin embargo, debido a que revelan importantes capacidades de las máquinas, describiremos algunos hechos acerca de algunas máquinas computadoras actuales, y otros hechos acerca de los denominados "sistemas expertos". El artículo termina con algunas consideraciones acerca de la pregunta: ¿quién es más inteligente, el hombre o la máquina?.

1 Introduction

In this paper, the questions of what machines cannot do and what they can do will be treated by examining the ideas and results of eminent mathematicians and computer scientists.

Regarding the question of what machines cannot do, we shall rely on the results obtained by the great, and at the same time eccentric, mathematicians Alan Turing and Kurt Gödel.

Regarding the question of what machines can do, we shall mainly rely on the ideas of the Nobel laureate Herbert Simon. Nevertheless, a few facts about modern computing machines and about the so called "expert systems"shall be described, because they suggest the existence of interesting capabilities of machines.

2 Some of the things that machines cannot do

As advanced above, we shall focus our attention on Turing and Gödel. For an interesting history of machines see [64].

2.1 Alan Turing

2.1.1 Turing machines

Alan Mathison Turing was born in London in 1912. He and his team decodified the German codifying machine "Enigma", contributing decisively in this way to the Allied victory over Germany in the Second World War. He himself designed the decoding machines named "Bomb" and "Colossus". As a reward for his successful contributions to the war effort, Turing was awarded the Royal Medal of the British Empire.

He was homosexual and was accused to maintain sexual relations with a minor. This produced the rejection of his colleagues and the British Government obliged him to medicate in an exchange for not going to prison. This situation, together with the awful secondary effects of the medication imposed to him, made him to commit suicide, in Wilston, in 1954. An excellent biography of Turing, as well as of Gödel and of other logicians, is Mosterin's book [45].

The reason for which he is recognized by the scientific community is the creation of "Turing machines" (Turing used to call them "Logical computing machines"), which were later able to show the existence of an important undecidable problem known as the "Entscheidungsproblem" (German for "decision problem"), that will be treated later on.

In its simplest version, a Turing machine consists of a potentially infinite tape, divided into squares in such a way that each square is either blank, also denoted by a "0", or an "1", and a reading-writing head which is able to read what is written in those squares. Afterwards, the reading-writing head follows only one of the next commands each time:

- erase what is being read and write a "0" or an "1" in that square,
- remain "looking" at the square,
- move one square to the left,
- move one square to the right.

The instructions of the machine are of the form, say, $q_i 1 L q_j$ which means "if the machine is in state q_i and its reading-writing head is reading an 1 in some square of the tape, the reading-writing head should move one square to the left (L) and the machine ends in state q_j", or of the form, say, $q_i 1 0 q_j$ which means "if the machine is in state q_i and its reading-writing head is reading an 1 in some square of the tape, the head erases that 1 and writes 0 in that same square". For such instructions, the machine ends in state q_j (j is not necessarily equal to $i + 1$ and q_j may be the same as q_i, all this depending from how the machine has been programmed). If a command ends in the state symbol q_j, the next command must begin by the same state symbol.

The machine (in its simplest version) is at "the standard position" when in its initial state the head is looking at the first square containing a "1" of a finite row of squares containing an "1" each (no blanks in between), the remainder of the tape being empty.

The problem is that Turing machines are complex, even to perform duties as to "compute" the function $y = 2 \cdot x$ for any given natural number x, that is, the operation "multiply by 2" (something that young boys and girls learn to do mechanically). The Turing machine [8] that computes this function has twelve states with the possibility of having to return to the first state as many times as needed to double the starting number of "1" in the initial state, when the machine is started in a standard position.

This complexity (relative complexity, as will be seen later) does not matter much, because what Turing was looking for was not mainly, although this was one of his aims, to construct a "calculator", as many mathematicians before him, like Leibniz, Pascal and Babbage, among others, had tried to, but to provide an exact definition of "algorithm", and an exact definition of "computation".

The terms "computation" and "algorithm" or "effective calculation" seem to coincide. When thinking on algorithms, we are prone to refer to actions performed by human beings (always with the help of "artifacts" that can be from a pencil to a machine) meanwhile when thinking on computation we are prone to refer to actions performed by machines (always with the help of a human being). When defining what is an algorithm, it is, curiously but not well explained why, in addition to require the process to be "mechanical" and finite, it is required that no ingenuity or insight on the side of the human being is needed to carry out the algorithm, suggesting that humans performing algorithms are as intelligent (or as clumsy as machines).

Algorithms should be verified and validated. Verification consists, basically, in checking the consistency of the algorithm; validation consists of checking that the outputs of the algorithm are correct in a large number of their practical and theoretical applications, when matched with widely accepted empirical, technical, or theoretical results.

Verification and validation refer to three levels of correctness of an algorithm. In the first level, it is checked that the outputs of the algorithm are correct in all cases it has been applied. The bigger the number of correct outputs is found, the bigger the credibility of the algorithm is. The second level is to formally modelize classes of tentative applications of the algorithm and show that the algorithm outputs consistent relevant results in each of those classes by applying it to a relevant representative of each class. The third level consists of using computer verifiers-provers that interact with teams of mathematicians and/or computer scientists, after performing a careful translation of the commands and other items of the algorithm into a symbolic language able to be understood by both the computer and the human team.

When an algorithm goes successfully through the first and second verification levels it is usually considered as verified and validated. It has to be said that going through the third verification level is a sample of the everlasting search for rigor in mathematics; moreover, even in the case the algorithm has successfully passed the requirements in the three verification levels, "absolute" verification is not obtained, but "absolute" certainty does not exists in any field of knowledge.

Buchberger algorithm has accomplished superabundantly the requirements of the first two verification and validation levels. But it has been a nice exercise of rigor to submit it to the third level requirements, a process carried out by José Antonio Alonso, José Luis Ruiz e Isabel Medina, of the University of Sevilla, Spain, using ACL2, being the result positive, as expected.

Professor Alonso, who was the one who informed us about the appropriateness of considering three levels in verification and validation, adds the question: but, who or what verifies/validates the verifier/validator?

To the approach to an exact definition of effective calculation contributed, on one hand, Church and his

disciples Rosser and Kleene, specially the later, who developed the so-called "λ-calculus" (a function of positive integers is λ-calculable if its values are found by using a certain repeated process of substitution, not to be explained here) and, on the other hand, Herbrand and Gödel, who developed the so-called "theory of recursive functions", all of them in the period 1932 to 1936. Turing, just after knowing Church's work on λ-calculus, showed that his (Turing) computable functions coincided with recursive functions (in the domain of positive integers).

Other algorithms as those provided by Post and Markov, and other machines, specially "unlimited register machines" (or, simply, "register machines") [13], have been shown to be equivalent to Turing machines (that is, they compute the same functions, i.e., recursive functions). Thus, it is indifferent to say "recursive functions" or "computable functions". Register machines, conceived by Shepherdson and Sturgis [61] consists of a tape divided into squares, as Turing machines, but infinite only to the right of a square considered to be the first one in the tape. They also consist of a simple set of instructions. Examples of instructions are:

- Z(n), which means: write a 0 in square nth, and
- J(m, n, j), which means: if the number which appears in the mth square is the same as the one in the nth square, jump to the jth instruction; otherwise go to the next instruction to J(m, n, j).

The initial configurations of the tapes of these machines consist of a finite series of natural numbers, each of them written in a square of the tape, beginning in the first square. After the rightmost square, in which a natural number different from 0 appears written, the remainder of the squares of the tape are filled with a 0 each. Register machines processes reflect very well the processes of construction of recursive functions. They are much easier to study and program than Turing machines, very likely because they look much more like real computers.

Recently, "machines" for the so-called molecular based on ADN combinatorics and cellular systems have been proved to be equivalent to Turing machines. The first author of this article witnessed such a proof for cellular computing provided by a student of Professor Paun in Sevilla University. Excellent introductory books for descriptions of these "machines" are [47, 48] (essential parts of the hardware of these machines, obtained in biochemistry laboratories rather than in conventional factories, are totally different from conventional computers hardware). They still are only a little farther than being at the prototype level, but they have been shown to be faster that supercomputers in solving a few problems to which they have been applied. An example is Adleman experiment [2, 3], improved, in regards to the speed of the experiment, by Lipton [41].

Adleman experiment consisted in applying molecular computation to the problem of solving a concrete instance of the Hamiltonian path in a graph with two distinguished nodes (the graph in the example had 7 nodes). To each vertex and to each edge an oligo and an ADN molecule are, respectively, associated. Regarding the capability of information storage in ADN molecules, it is one bit per cubic nanometer (a nanometer, "nm", is equal to 10^{-6} millimeters), while the best conventional supercomputers store one bit per 10^{12} nm^3. A "good" (not yet built) molecular "machine" could perform 10^{20} operations per second, while the best today's supercomputers can perform 10^{12}.

Finally, it is curious to check that abaci, the oldest "machines" compute the same functions than Turing machines (see Chapter 6 of [8]). As all the mentioned machines are equivalent to universal Turing machines (universal Turing machines embody all Turing machines), all of them compute the same class of functions.

An interesting and illustrative testimony about the importance of an appropriate "atmosphere" in the birth of genial ideas, on logic in this case, is Enderton's quotation [46].

> Princeton in the 1930's was an exciting place for logic. There was Church together with his students Rosser and Kleene. There was John von Neumann. Alan Turing who had been thinking about the notion of effective calculability, came as a visiting graduate student in 1936 and stayed to complete his Ph. D. under Church. And Kurt Gödel visited the Institute for advanced Study in 1933 and 1935, before moving there permanently.

Let us pause for a while to describe what recursive functions are and what relation they have with arithmetization of formulae of first order theories (this introduction to the concept of arithmetization will be completed when studying Gödel numbers in 3.1). Our description is based on [43].

Recursive functions are defined inductively. The "initial functions" are three: for the sake of simplicity (these functions, as well as the rules for defining new functions from given functions are defined in any book on the theory of computability), we only mention the "zero function": $Z(x) = 0$ for all x and the "successor function": $S(x) = x + 1$ for all x. The rules for obtaining new functions from given functions are "substitution", "recursion" and "introduction of the μ-operator". The recursive functions obtained from the initial functions and from finite number of applications of only substitution and recursion rules are called "primitive recursive". These, plus those obtained from them by finite numbers of applications of Introduction of the μ-operator rule are called "recursive".

According to Mendelson, the first semi-automated try of formalizing arithmetic was performed by Dedekind [14]. This formalization was improved by Peano, and the resulting set of nine axioms is known as "Peano's arithmetic". Peano's theory has the shortcoming that the notion of "property" used in its ninth axiom (an induction axiom) relays much on intuition. Mendelson improved and made changes in Peano's arithmetic, by building a theory, again with nine axioms, which he named "theory S", a totally formalized axiomatic theory which avoids intuitive notions. This theory accounts for the proofs of all results of elementary number theory. These results refer to properties of "number theoretic relations" and of "number theoretic functions", which are, respectively, relations with arguments natural numbers and functions, the arguments and values of which are natural numbers too.

To each natural number a term is assigned (which is a formal expression, not a number), called "numeral". Numerals are the logical terms 0, $s0$, $ss0$, ... where 0 is the only constant of theory S and s is the function "symbol successor". These terms are represented, respectively, as 0, $\overline{1}$, $\overline{2}$, $\overline{3}$, ...

In this context, a number theoretic relation $R(k_1, k_2, \ldots, k_n)$, where $k_1, k_2, \ldots, k_n$ are natural numbers, is "*expressible* in the theory S" when there exists a well formed formula $A(x_1, x_2, \ldots, x_n)$ of the language of S with n free variables, such that for any list of natural numbers k_1, k_2, ..., k_n the following two conditions hold:

1) if $R(k_1, k_2, \ldots, k_n)$ is true (in the model of natural numbers), then S $\vdash A(\overline{k_1}, \overline{k_2}, \ldots, \overline{k_n})$ (which means that $A(\overline{k_1}, \overline{k_2}, \ldots, \overline{k_n})$ can be proved in S).

2) if $R(k_1, k_2, \ldots, k_n)$ is not true (in the model of natural numbers), then S $\vdash \neg A(\overline{k_1}, \overline{k_2}, \ldots, \overline{k_n})$ (that is, $\neg A(\overline{k_1}, \overline{k_2}, \ldots, \overline{k_n})$ can be proved in S).

A similar, although a little more complex, definition of "representability in the theory S of number theoretic functions", can be given, but in order to simplify our description, it is omitted here.

But what is of utmost importance is that *recursive functions and, consequently, all number theoretical computable (by ideal machines) functions coincide (if the Church-Turing thesis, in the Church version, is accepted, see below) with the functions representable in S.*

The importance of "expressibility" and "representability" is that they relate natural number theoretic relations and functions with formulae of the language of S, that is, recursive functions (with natural numbers as arguments) with formulae of the arithmetic S. For instance, expressibility relates the recursive function theory predicate "$\mathrm{For}(x)$" and "$\mathrm{Pr}(y, x)$" (meaning that y is the Gödel number (to be defined in 2.1.5) of the proof of a formula, the Gödel number of which is x), with (two) formulae of S (this will be used later on to describe intuitively the meaning of the second of Gödel incompleteness theorem).

But what happens with functions and relations, the argument of which are not natural numbers? The answer is that first order theories and, thus, many expressions and processes (as "to be a formula" and "to be a proof") of mathematics can be arithmetizable in the sense that they can be codified by using Gödel's numbers and functions and relations (predicates) of recursive functions theory. This allows to apply recursive functions theory ideas and constructs to study the meta-mathematics of mathematics (meta-mathematics deals with, among other items, general properties of groups of theorems of a part of mathematics: for instance, the tautology theorem in meta-logic deals with properties of mathematical logic theorems in such a

way that it helps proving new logic theorems). Gödel's incompleteness theorems can be considered meta-mathematical theorems.

2.1.2 Church-Turing thesis

Church-Turing thesis (see [68]) is considered by some as "the first axiom of the theory of computability". Nevertheless, this is a little an exaggeration because one should always have in mind that it is not more that a "thesis" (or rather, a well founded hypothesis).

In Turing's version, it says: LCMs (logical computing machines, as Turing named his machines) can do anything that could be described as "rule of thumb" (that is, a useful way of calculating approximately) or "purely mechanical". If the word "calculable" means: anything that could be described as "rule of thumb" or "purely mechanical", then the translation of the previous statement is:

$$\text{calculable} \Longrightarrow \text{Turing computable}$$

In Church's version, the thesis says: "a number theoretical function (that is, a function of natural numbers) is effectively calculable only if it is recursive":

$$\text{effectively calculable} \Longrightarrow \text{recursive}$$

"Rule of thumb" and "purely mechanical" are fuzzy notions; to be "effectively calculable" is usually understood as a mechanical procedure that, in principle, can be carried out by a human being in a finite number of steps, with nothing more than a paper and a pencil, with no insight or ingenuity on the side of the human being. An example is using tables of truth-values for checking whether or not a propositional formula is a tautology (it has been said above "in principle" because complexity of the algorithm may be so great that it is not useful to carry it out). That effective calculations do not require ingenuity or insight is quite dubious either.

For these reasons, the next versions, one by Gandy and the other in [68], seem to be more rigorous.

Gandy's version is: "whatever can be calculated by a machine (working on finite data, in accordance with a finite program of instructions) is Turing-machine-computable.

The thesis in [68] (denoted as "thesis S" by their authors) is: "any process that can be given a mathematical description (or that is scientifically describable or scientifically explicable) can be simulated by a Turing machine".

A nice consequence of the thesis is (see [70]).

> All ordinary computers are equivalent to each other in terms of theoretical computation power, and it is not possible to build a calculation device that is more powerful than a computer.

From what has been said, Church-Turing thesis does not say that any kind of calculus is a computation, as one would be tempted to say when reading only the original Turing statement of the thesis, that may suggest that calculation is a mathematical proof plenty of intuitions and maybe based on dubious bases. An interesting example of this kind of calculation was Boole's inferential method for attaining consequences from a given set of logical hypotheses. He, by using a doubtly well based methodology (see 4.1.1), reached a result that corresponds to a today's accepted theorem which links logical consequence with an ideal membership problem, a sound effective method. It has to be said that many programmers do not care too much about whether or not their procedures are actually computations, except when rigor problems arise.

That computable implies calculable is clear, the converse is the questionable one: an equivalent form to state Church-Turing thesis is "what isn't computable isn't and never shall be calculable".

The thesis is no more than an hypothesis, but it is supported by the fact that all attempts to define effective calculability have ended in realizing that they are equivalent to Turing computability. After Turing's work, many ideal machines and algorithms, mentioned in 2.1.1 above, have been designed for many different reasons: to simplify Turing machines, to try to extend these, to try to find new classes of computable functions that are not Turing computable and to make computations faster and more efficient. All these

machines, have curiously been proved to be equivalent to Turing machines and to compute the same class of functions, recursive functions, thus also denoted as computable functions. The equivalence of the mentioned machines with Turing machines support Church-Turing thesis written as follows: "universal Turing machines can simulate the behavior of any machine".

We strongly recommend the discussion of the "Church-Turing thesis" (specially its misunderstandings) provided in the Stanford Encyclopedia of Philosophy, with an impressive bibliography [68].

At the end of this paper we shall make some comments about the relationship between this thesis and the comparison of human intelligence with artificial intelligence.

An important comment is: the rising of quantum computation makes us guess that a new type of machines that aren't equivalent to Turing machines could be developed. And then, perhaps a new type of Turing-like machine could be shown to be equivalent to quantum computers. Interesting and important studies on this topic appear in [6, 15].

2.1.3 Undecidable problems

Turing and Church, by using different approaches, proved the undecidability of the "Entscheidungsproblem" proposed by Hilbert and Ackerman in 1928 (given a formula of a predicate logic system, to decide by an algorithm whether or not such formula is a theorem in that system).

Undecidable problems are problems such that not only machines or men using algorithms have been unable to solve but, rather, that there exists a proof of the fact that such problems aren't and never shall be able to be solved either by an algorithm or by a machine.

Let's outline, in today's terms, the proof of the undecidability of the "Entscheidungsproblem". Among the possibly Turing computable functions, there is a curious one, the "productivity function p". The "productivity" P of a Turing machine is defined as follows.

Suppose a Turing machine the head of which starts marking an empty square in an empty tape. If after performing all its instructions, the head of the machine ends in standard position marking the first "1" of a row of k "1s", being the remainder of the tape empty, the productivity of the machine is k. Let us agree that the productivity P of a Turing machine is cero in the following cases: if it does not start, if it doesn't end in the standard position, if the final string of "1" does not form an unbroken row and if it doesn't stop.

The productivity function "$p(n)$" is the productivity of the most productive Turing machine of n states. This machine, if it exists, is called "BB", "busy beaver", a mention to the frantic activity of beavers carrying sticks ("1") to fill gaps in walls.

The final result is that BB does not exists, in other words, that the productivity function p isn't Turing computable. If Church-Turing thesis holds, p is neither computable nor calculable. The computation of $p(x) = y$ is an undecidable problem. The problem of the undecidability of the problem of computing p is linked to the "halting problem", which is the problem of designing a systematic and effective procedure for identifying the Turing machines that never halt, once started in their standard position. Suppose that we are in an extremely large room in which all Turing machines of n states fit, and imagine we make them start on empty tapes. Some will stop in situations for which they can be assigned a productivity $P \geq 0$, or a productivity 0; but, what happens with those machines that seem not to stop? What would be needed is an algorithm or a machine which would state, generically, whether or not a Turing machine stops. But if such a machine would exist, it would be able to compute the productivity function.

Thus the decidability of the halting problem is reduced to the question of existence or non existence of the BB machine, a problem whose answer is "no". It means that the halting problem is undecidable, that is, it isn't and never shall be Turing-computable.

It can be asked: does it really matters if the halting problem of a Turing machine is an undecidable one? The answer is that the Entscheidungsproblem, referring to formulae in the predicate calculus with predicates in two or more arguments is reducible to the halting problem, and thus, it is undecidable. A nice proof of the fact that the Entscheidungsproblem is reducible to the halting problem, not using Turing machines but "register machines" appears in [13].

By the Completeness Theorem in Gödel's version (it has another version by Henkin that will be later enunciated and used), also the "validity" problem (to determine by an algorithm whether or not a formula of the predicate logic, for predicates in two or more arguments, is "valid") is undecidable. Gödel's version of the Completeness theorem says: a (first order theory T) formula is a theorem of T if and only if it is valid in T (which means that it is valid in all models of T).

The problem for monadic logic, that is, for predicates in one argument was proved by Löwenheim to be decidable. Löwenhem was born in 1878 in Krefeld, Germany, and died 1957 in Berlin, and he had a very difficult life (almost all his writing in mathematics and logic were lost in a bombing on Berlin). He was, in some sense, able to "codify" logical foundations of mathematics in the domain of real numbers. What he proved is that, for any set of sentences of predicate logic, if there is an interpretation in which they are true in some structure, there is also an interpretation that makes them true in a countable subset of the original structure.

It is also known that, for propositional logic, the problem (in this case of determining tautologies) is decidable (by constructing truth tables).

Thus, predicate logic, which is, together with propositional logic, at the very basis of human formalized thought and of the machines symbolical processes (and thus, of artificial intelligence), does not possess effective, general and finite procedures to determine neither the theoremicity nor the validity of its statements. This, we do not exaggerate, contradicts rooted beliefs of mathematicians and philosophers about the strength of the foundations of formal sciences (and, maybe, about the validity of the belief that symbolic processes of human beings brains are always sound). Recall that a formal science is a science such that its assertions (among them its axioms) can be expressed, symbolically, in a very precise language and that the procedures for making inferences are very precisely stated too.

2.1.4 A digression

As a digression, suppose that a person with little level of power of abstraction, but having some semi-mechanical capabilities (to distinguish just four different symbols like "0" "1", "L" and "R", to erase "0" or "1", write "0" or "1", go left, go right) and to know the actions to be performed when seeing the pairs (0:0, 0:1, 0:L, 0:R, 1:0, 1:1, 1:L, 1:R) and able to follow orderly the arrows in the diagram of the Turing machine, is provided by

(1) such a diagram (on a piece of paper),

(2) a pencil with an eraser incorporated.

The paper contains in its upper side a row 1 1 of two "1". We doubt what would be more difficult for him: to begin understanding an elementary notion of natural number, to know how to sum these numbers and to relate sum to product, or just to follow, mechanically, the instructions of a Turing machine.

2.1.5 The intelligence of machines

Regarding Turing's interests on the problem of the intelligence of machines, he proposed what is known now by the "test of Turing" in the journal "MIND", in 1950. Turing began stating that is first essential to give a precise meaning to the terms "machine" and "thought"and went into the following test-play.

There are three players, a man (A), a woman (B) and another person (C) that proposes questions. C can be a man or a woman. C is in a room different from A and B. The aim of C is to find out, by asking a battery of questions, who of A and B is a woman (or a man)? The answers are written in typewriters (so that A and B cannot be identified by their voices or their writing). C may ask questions related to calculi, which are, usually, better treated by machines, and other questions referring to emotions which are, usually, better answered by human beings.

Turing asks the following question: if we replace A by a machine, does C makes as many mistakes while assessing the gender of A and B as in the case that A is a person (a man)? If the number of mistakes was the same, according to Turing, the machine would be intelligent.

Turing himself recognized that intelligence is a so complex entity, that his test of intelligence of machines was very limited but, that by trying experiments like this one, some time in the future machines could be attributed intelligence.

Note that we have been making a distinction between solving problems by "human beings using algorithms" and "machines solving problems". It has to be said that, at the present time, there is not an essential distinction between these two concepts. Today, most humans solving problems by algorithms use machines, not merely pen and paper. Thus, when we say "problem solved by humans using algorithms", we are almost saying "problem solved by humans using machines". We have said "almost" because machines are little by little becoming independent from humans. This is not only because machines can perform tasks in which humans are less efficient, both cuantitatively, like working with large pieces of information or, qualitatively, like (this refers to most humans) playing chess, but, also, because machines, as we shall see later on this paper, emulate human behavior and simulate human characters as having "insight". This doesn't imply that machines are intelligent yet, because they accomplish all those mentioned tasks because some human(s) being(s) has (have) programmed them.

3 Kurt Gödel

3.1 The arithmetization of logic. Gödel numbers

Kurt Gödel was born in Brno, today in Czech Republic, in 1906. In 1929 he moved to Vienna, in which University stood out both because of his intelligence and his peculiar personality. It was during his sojourn at this university when he discovered his famous incompleteness theorems.

When Hitler overtook power in Germany and Austria, he emigrated to the U.S.A., and was hired by Princeton University. He returned to Vienna because of an illness: his biographers say that during his illness, he spent much time reading about madness, something that might have made him even more eccentric than he was before

He returned definitively to America in 1940. Gödel died in Princeton, in 1978.

He is considered among the most creative mathematicians in history. His contributions were important in the foundations of mathematics and in the theory of computability. Maybe because he and Einstein became scientific partners in Princeton, Gödel dedicated a great amount of time to the study of physics, although this interest for physics had also been remarkable while he was a student in Vienna, much before he went to Princeton.

In this paper we shall focus our attention only, because this has to do with our topic about what machines cannot do, to two Gödel's genial mathematical constructs, which are intimately related between them: the "arithmetization" of formal logic and the incompleteness theorems.

Gödel had the, seemingly just curious but actually crucial, idea that a natural number (a code) could be assigned to formulae, axioms, theorems, proofs and so on, not only in logic, but also in arithmetic and theories which are extensions of arithmetic. This idea, together with his contribution to the foundations and development of the theory of recursive functions led him to state two of the three incompleteness theorems (the other one was due to Church). We deal next, in a quite informal way, with both the arithmetization of logic and formal theories containing arithmetic and the incompleteness theorems [62].

There exists a computable (recursive) function "ß" that assigns a natural number, that is, a code, here denoted "c" to any list of natural numbers. The Gödel number determination is totally algorithmic or able to be performed by a Turing machine or its equivalent machines). Gödel proved that ß has the following properties:

$$\begin{aligned} ß(c,0) &= \text{number of elements of the list} \\ ß(c,i) &= i\text{th element of the list, if } i>0 \end{aligned}$$

Now, given a formula like $\forall x \exists y A(x,y)$, which has ten symbols, if a different natural number is assigned to each symbol, one gets a list of ten natural numbers and, thus, the Gödel number of the list.

As a proof is on its side a list of formulae and a Gödel number corresponds to each of these formulae, we again have a list of natural numbers assigned to the proof and, thus, a Gödel number can be assigned to the whole proof. As a matter of course, theorems, which are the last formula of a proof, have their Gödel number too.

An illustration of the Gödel number of a theorem, which appears in the interesting book of Douglas Hofstadter, "Gödel, Escher, Bach" [21], is the number that corresponds to the logical formula that translates the theorem: "The number of prime numbers is infinite":

$$8445329844508787863070005766619463864455067111 \ .$$

An alternative to the use of the function ß is the following [43]: to each symbol, a certain natural number is assigned. This was advanced above, but without stating what number corresponded to each symbol. For instance:

$$\begin{array}{rcl} G(() & = & 3 \\ G()) & = & 5 \\ G(,) & = & 7 \\ G(\neg) & = & 9 \\ G(\rightarrow) & = & 11 \\ G(x_n) & = & 5+8n, \text{ for } n = 1, 2, 3, \ldots \\ G(a_n) & = & 7+8n, \text{ for } n = 1, 2, 3, \ldots \\ G(f_n^m) & = & 9+8\cdot(2^m\cdot 3^n), \text{ for } n = 1, 2, 3, \ldots \\ G(P_n^m) & = & 11+8\cdot(2^m\cdot 3^n) \text{ for } n = 1, 2, 3, \ldots \end{array}$$

where x_n are variables, a_n are constants, f_n^m is a function symbol and P_n^m is a predicate symbol, for $m, n \geq 1$.

The Gödel number of an expression as $P_2^3(a_2, x_4)$ is

$$2^{11+8\cdot(2^3\cdot 3^3)}\cdot 3^3\cdot 5^{7+8\cdot 2}\cdot 7^7\cdot 11^{5+8\cdot 4}\cdot 13^5$$

Note that it is a product of powers of ordered prime numbers, and each of these powers is the Gödel basic number assigned to the corresponding symbol. This is an arithmetization of logic, of arithmetic and of consistent extensions of arithmetic.

The meaning of the predicates "$\mathrm{For}(x)$" and "$\mathrm{Pr}(y, x)$" have been described in 2.1.1: a formula, formed with these two predicates: $\exists x(\mathrm{For}(x) \wedge \forall y \neg \mathrm{Pr}(y, x))$ will be used in Gödel's second incompleteness theorem. It means that there exists a Gödel number x of a formula such for all y, $\mathrm{Pr}(y, x)$ the predicatė "y is the Gödel number of a proof of the formula the Gödel number of which is x", is false. Intuitively, "there is a formula that cannot be proved". As explained below, a broad definition of "inconsistent theory" is that all formulae of the language of the theory have a proof in the theory, which would mean that contradictory formulae could be also proved, meaning inconsistency.

Thus, that "there is a formula that cannot be proved", means consistency.

Not only For and Pr can be defined in recursion theory, but also a great variety of assertions can be defined too. An example is "to substitute" (function $\mathrm{Sub}(a, b, c)$), which means: "substitute in a formula which Gödel number is a, the variable which Gödel number is b by the term which Gödel number is c. In Cutland's book [13], even program commands are assigned Gödel numbers.

3.2 The incompleteness theorems

The ideas described above, together with Gödel's work on recursiveness and other extra-scientific considerations, led him to the intuition of his two incompleteness theorems. As seen above, Church also worked on recursion theory and on incompleteness, but, Church and Gödel knowledge of each other's work probably took place when Gödel went to Princeton for the first time. Any way, we shall assert the three incompleteness theorems in this order: Church's version first, and the two Gödel's versions afterwards.

These theorems produced both a deep change about the idea of capability of formalization of mathematics in the way Russell and Whitehead guessed, but also had much to do in the problem this paper deals with: what machines (and, consequently what both mathematicians' formalisms and the mechanical behaviors of our brains) can or cannot do.

The completeness theorem has a second version, due to Leon Henkin [62], under whose supervision the first author of this paper worked during an academic year. This version says: a theory is consistent if and only if it has a model.

Let us explain Henkin's version.

Informally, a model of a theory is a universe where the axioms of the theory "have sense" (more formally, "are valid formulae").

For instance, Euclid proposed, as one of the axioms of Geometry, that through a point external to a straight line, only one parallel to the given line can be drawn. Our three dimensional world is a model for Euclidean geometry because this axiom, as well as the others he proposed, have sense in this world.

The theorems in any theory follow from the axioms by application of some explicit inferential method. The theorems also have sense in the same models where the axioms have sense.

Axioms aren't, as students many times told, truths no subject to discussion, but, rather, assertions which gather in a few and condensed statements, prior knowledge. From those assertions, knew knowledge can be extracted (theorems).

Coming back to Euclid's example, he probably was an excellent teacher who organized the dispersed knowledge in Geometry (the tradition says that on Plato's Academy doors, there was a statement saying "none ignorant of geometry can cross this door") in axioms and results rationally proved from the axioms and the previously proven results.

The problem with axioms is that when new visions of the world question them, they have to be revised or even abandoned. Newton "inertia principle" (an axiom) suffered this process of being questioned when relativity theory changed the idea of space as being curved. In this situation, neither Newton's nor Euclid's axioms continued being the only models of the world.

In mathematics there aren't changes in paradigm in regards to theories (sets of axioms), but based on our next argument it seems to us that there are paradigm changes in methodology, being probably the most important one, or seems to us to be, the one that is taking place at the present time because the interaction of computers and mathematics.

The appearance of a recent methodology, not following conventional mathematical proof methodologies, of solving an old problem, "Kepler's problem" (he guessed a solution by 1611, but he couldn't find a proof) may suggest that changes of paradigm (this concept is the core of Kuhn's ideas on scientific revolutions [25]) in mathematics is now or will be soon occur; our account is taken from Keith Devlin's description in [16].

The problem asks what is the more efficient way to pack equal-sized spheres together in a large crate. Kepler suggested that the more efficient way was to pack them as greengrocers do and have been doing when packing oranges by centuries, which is that each sphere (orange) in each higher layer sit in the hollow made by the four spheres beneath it.

As Devlin says, the general problem isn't formulated in terms of the number of spheres that can be packed in the crate, but in terms of the density of the packing, which is the total number of spheres divided by the volume of the crate they are contained in. Let us insist that we are referring to large crates.

Gauss almost approached the solution proving that the mentioned "orange" piled arrangement was the most efficient among all "lattice packings", but for reasons appearing in Devlin's account, this was not the solution to the problem. The real solution came when the Hungarian mathematician Laszlo Toth managed to reduce the problem to a huge calculation in 1953. Using this idea, Hales and Ferguson announced in 1996 that they had found the solution. But the solution required 250 text pages and 3 gigabytes of computer stored data and programs. Hale posted the solution in the Internet, something that allowed that as many mathematicians as would be willing so, could check the proof.

Popper's original idea is that a theory in science can be "proved" (better, "accepted"), if it has the property of being "falsable" [50]. Being "falsable" does not mean that it can be shown to be false but,

rather, that it is built in such a form that any scientist willing so, is able to try to find counter-examples to the theory by using sound scientific experimentation or by finding logical inconsistencies in some of the models of the theory (theories as the happiness of the Greek gods in the Olympus cannot be falsable, "falsable" has a different meaning than "false"). If someone finds a strong counter-example to the theory, the theory should be changed or abandoned (not always happens so, as Kuhn opinions on how the changes of paradigms actually take place).

Hale's proof acceptance of the solution of the Kepler's problem follows a very similar path as Poper's proof acceptance of falsable theories. Being posted in the Internet, as it is, there are in principle, thousands of counter-examples searchers for Hales' proof of the solution of the Kepler's problem. No counter-example has been found.

A similar process followed the resolution of the problem of the four colors, but by that time both the theory and the huge calculations needed for its solving could not be posted in the Internet because it did not exist by that time.

If proofs of the type of Kepler's problem and the four colors problem are accepted, as they belong to a truly non conventional mathematical proving and solving methodology, maybe a change of paradigm is taking place in mathematics, as has happened in astronomy and physics.

Coming back to the completeness theorem, it is very important to emphasize that Henkin's proof of the theorem uses, at a certain point, Zorn's lemma, so that the proof is existential, not constructive. No algorithm or machine can carry out the complete proof.

There are many proofs in mathematics, mostly based on the Axiom of Choice, that aren't constructive. These type of proofs are accepted because their usefulness (otherwise a good deal of mathematical theories or ideas had to be rejected), but the tendency of mathematicians is, usually, that once an existential proof has been found, a constructive proof is searched for.

When discussing the second incompleteness theorem, it shall be seen that this play of existential and constructive proofs leads to a seemingly paradoxical situation.

A digression is: if existential proofs are considered as "calculi", Church-Turing thesis suggests that now or some time in the future, they shall become computations (but recall that there aren't computations for many foundation problems).

Regarding the incompleteness theorems, recall that they are three. The first to be stated, although is posterior to the two others is due to Church, and the second and third, to Gödel. The three theorems shall be stated formally but shall be followed by informal translations because, otherwise, some complex procedures from recursive functions theory would be needed to be explained in detail.

By making a not much orthodox extrapolation, it shall be also suggested that these theorems can say something about the limitations of artificial intelligence.

The first incompleteness theorem says: *if T is a consistent extension of arithmetic, then T is undecidable*.

That a theory is decidable means that the predicate $Thm_T(x)$, where x runs over the Gödel numbers of the theorems of T (in other words, the predicate "to be a theorem" of T) is recursive or computable. Thus, a simplified statement of the first incompleteness theorem is the following one:

> Any theory that both contains arithmetic and is consistent, doesn't have any effective general technique for deciding, given a formula stated in its language, whether such a formula is or isn't a theorem of the theory.

In terms of artificial intelligence one could say:

> Artificial intelligence does not have machines or algorithms to determine if, given a statement expressed in its language, such an assertion is or isn't a consequence of its own presuppositions.

The second incompleteness theorem is: *if a theory T is a recursively axiomatized extension of arithmetic, then it isn't complete*. To be axiomatized, for theories, means that the predicate $Ax_T(x)$, where x runs over the Gödel numbers of the axioms of T (in other words, the predicate "to be an axiom" of T) is computable

(or recursive). To be complete, for a theory, T, means that for each one of all its closed formulae, it can be stated that either the formula or its negation is a theorem in T.

Thus, a simplified statement of the second incompleteness theorem is the following one.

> Any theory that both contains arithmetic and has an effective procedure to recognize its axioms, contains at least a formula such that neither that formula nor its negation can be proved inside the theory.

In terms of artificial intelligence one could say:

> If artificial intelligence has a machine or an algorithm to recognize its own presuppositions, there is at least one expression of its language such that neither that expression nor its negation can be proved.

A consequence of the two first incompleteness theorems is that not all mathematical theories are decidable and/or axiomatized under the meaning these predicates have in the theory of computability (or recursion theory). If Church-Turing thesis is admitted, this means that not all theories can recognize its axioms by using algorithms and that not for all consistent theories can an algorithm be defined for deciding in general, that given a formula written in its own language, such a formula is or isn't a theorem in the theory.

The third incompleteness theorem is: *if arithmetic is consistent, then, inside it, there isn't any algorithm that can prove the formula*

$$\exists x(\mathrm{For}(x) \wedge \forall y \neg \mathrm{Proof}(y, x))$$

(this formula means: "there exists a formula that has not a proof"). This formal statement can be informally stated as follows.

> If a theory which is an extension of arithmetic is consistent, there is no algorithm or machine that can prove that it is consistent.

But recall that the completeness theorem in Henkin's version says that a theory is consistent if and only if it has a model. As arithmetic has models, is, thus, consistent. Does this mean that "if a theory containing arithmetic is consistent, then it is inconsistent?".

This looks like the famous paradoxes in logic, specially the liar's paradox. If someone says "What I am saying is a lie", should it be interpreted as that when he/she is telling the truth he/she is lying and when he/she is lying he/she is telling the truth? The liar paradox, has been misused many times by people in important arguments (in preprints published previously to his recent book [59], Pedro Schwartz already comments the misuse of this paradox in politics). The paradox mixes a language and a meta-language. "What I am saying" belongs to a language: it is by itself a statement or a series of statements in a same level; to assert that "it is a lie", belongs to a meta-language that makes assertions about the statements in the first language.

Similarly, some interpretations of Gödel's second incompleteness theorem are circular because those interpretations identify the language in which "to be consistent" means "proved to be consistent by an existential and non-constructive proof" with the language in which in which "to be consistent" means "to be constructively proved by an algorithm or a machine to be consistent". Maybe because the existence of such a paradox, Hofstadter checked and compared in his book "Gödel, Escher, Bach" [21] circularities in the works of these three "artists": it is difficult to explain circularities in Escher's painting and in Bach's compositions, but they are due, probably, that they were using, respectively, a "Meta-painting" and a "Meta-music" which produce as result, both puzzling and beautiful pieces of arts, as pieces of art are the incompleteness theorems.

In terms of artificial intelligence, one could say regarding the third incompleteness theorem: If artificial intelligence (that must contain arithmetic) is consistent, it hasn't algorithms or machines that prove its consistency.

Has this something to say in regards to the comparison of the intelligence of humans and the intelligence of machines?

On one hand, the first type of consistency that is referred to in the theorem, can be proved by human intelligence by using accepted (under suspicion) mathematical beliefs but not by machines. But humans cannot design a machine that proves consistency of the most basic theories. Is, thus, human intelligence more powerful than artificial intelligence? A joke could be that machines are better (we do not say "more intelligent") than us because they prove that they cannot prove their consistency, while us never would accept that we are inconsistent.

Kant said that the axioms of Euclidean geometry are given a priori to human intuition. But new geometries appeared that later modeled new physics in which the axiom that says that by an external point to a straight line only one parallel line can be drawn does not hold. Well, Riemann proved that if Euclidean geometry is consistent, his was consistent either. Thus, two human constructions that say contrary things can both be consistent. We suppose that a machine that would contradict itself, would stop functioning.

According to Arbib [5], Ernest Nagel and James R. Newman assert that Gödel theorem limits in an unquestionable way the mathematical power of machines. Others, like Hilary Putnam, proposed a contrary opinion [5].

4 Some of what machines can do

4.1 Expert systems: machines can make logical deductions and knowledge extraction

Expert systems or knowledge based systems (to be both hereinafter denoted as "ES") are an interesting mixture of human and machine interaction. Some call then "expert" because the knowledge they contain has been provided by human experts, others ask the system to be able to learn by itself.

Our work on ES has been devoted to the first type of ES, which can be considered as systems based on automated inference. The experts have provided all necessary information. Our approach is a part of symbolic computation [31].

We believe that, so far, expert systems cannot substitute specialists. They can be useful, for instance, in the case of medical diagnosis, for non-specialists like family doctors, that may match their own diagnoses with the expert system's diagnoses, before deciding whether to send the patient to a specialist or not. In some cases, where the information contained in the expert system is remarkably complex and large, the ES becomes useful even to specialists.

Before describing what ES are, let us say that a crucial prior step to extracting consequences from the ES is to *verify* it. Verification consists of several steps, the most important of which is to check the ES is consistent [29].

The ES to be here described are called "Rule Based Expert Systems" which consist of three components.

- A "knowledge base" (denoted as KB), is composed by two different subsets of formulae. Let us call "literal" both any propositional variable and its negation.

  The first subset (hereinafter denoted as R), orderly collects and symbolically represents as logical formulae the available information obtained from experts' consultation, but, also from other sources, as published studies, both in books and journals and in the the Internet. These logical formulae, called "production rules", translate statements as "IF such and such factors, symptoms, etc. occur or do not occur, THEN such action or evaluation has to be performed".

  The second subset (hereinafter denoted as F, is called the set of all "potential facts". It coincides wit the union of the set of all literals in the IF side of the production rules which do not appear in any THEN side of these rules and the set of their contrary literals (for example, the literals p and $\neg p$ are contrary literals). The whole set F is not used in each computation but, rather, what are used are

subsets of F, generically denoted hereinafter by the letter A, which are maximal consistent subsets of F. For instance, in ES dealing with medical diagnoses, a subset A is assigned to each patient.

- An "inference engine" (hereinafter denoted as "IE") which both verifies the consistency of the the KB and extracts consequences automatically from the mentioned symbolic formulation of information.
- An interactive "graphic user's interface" (hereinafter denoted as "GUI") for users not necessarily being familiar with the logical and mathematical details of the system construction.

The implementation, in our case in the computer algebra language CoCoA ("a language for doing Computations in Commutative Algebra" [69, 11, 49]), leads to an automated method for knowledge extraction (for finding diagnoses in the case of medical ES). The method is based on previous works by Kapur and Narendran [24] and Hsiang's [22] (Boolean logic) and Alonso, Briales, Riscos and Chazarain [4, 12] (multivalued modal logic). An algebraic approach and its application to ES appear in [28, 51]. Gröbner bases (GB) of a polynomial ideal and normal forms of polynomials modulo an ideal (both based on Buchberger's algorithm) [9, 10] are used in the implementation. An introduction to GB and computer algebra can be found in [1, 66].

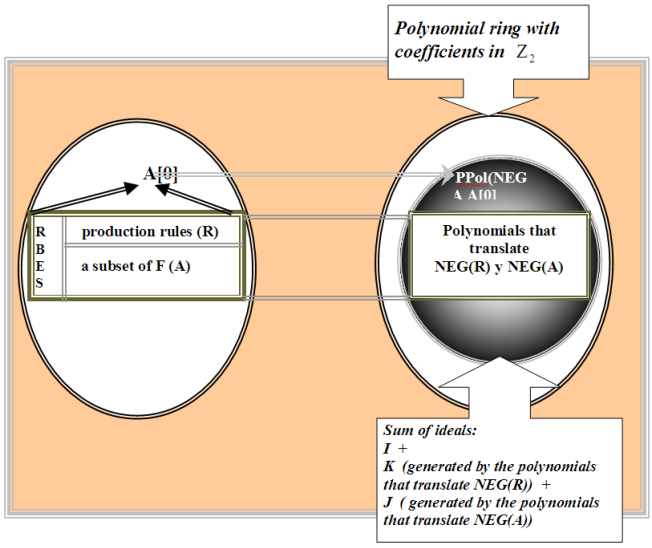

Figure 1. Intuitive graphic description of the theorem.

Let us describe the theorem intuitively using Figure 1. The KB of the ES is represented by the rectangle at the left side. It is composed by production rules and a subset (in the case of an ES for medical diagnoses, this subset characterizes a patient by his/her symptoms) of the set "F" of all factors and symptoms that characterize an illness.

The rectangle is enclosed into a set that represent the infinite set of all well formed formulae that can be written by using, in addition to logical connectives, the propositional variables contained in the formulae in the rectangle.

The question is: does a diagnosis (or treatment) "α" follow from the information (written as logical formulae) contained in the KB? Again, the KB is composed by production rules and facts characterizing a patient.

The process, that is complex, but that can be performed by any user not necessarily knowing mathematics or logic, consists of the following steps:

1. Translate all gathered information about the illness.
2. Translate such information into logical formulae (production rules, and factors that characterize a patient).
3. Translate the negations of these formulae into polynomials. This is done automatically by the program by providing the polynomial translation of the logic connectives (their number depends on the number of truth values of the logic considered). These polynomials are the members of the rectangle in the right side of Figure 1. They (and some other auxiliary polynomials) "generate" an ideal (the grey part of Figure 1) in a polynomial quotient ring with coefficients in a finite field ($2Z$ in case of bi-valued logic, $3Z$ in case of of three-valued logic, etc.).
4. The theorem says: α follows from the information contained in the KB (the rectangle at the left side of Figure 1) of the ES if and only if the polynomial that translates the (negation of) α belongs to the mentioned ideal.
5. But, how to know that a polynomial belongs to an ideal? This is the famous "ideal membership problem", a question that could be solved if obtaining canonical bases of a polynomial ideal was possible. But this topic was solved both by Heisuke Hironaka with his "standard bases" (Hironaka visited Madrid Royal Academy of Sciences recently) and Bruno Buchberger with his "Gröbner bases" in the late sixties. The latter have the advantage that an algorithm for their computation (Buchberger's algorithm) was provided [9]. This algorithm is very useful in many parts of computer algebra and geometry, but, also, in automatic proving in many other fields. The algorithm is pretty complex and requires knowledge of a non trivial background, but the only thing a user not necessarily familiar with mathematics has to do to check an ideal membership is just to type, e.g. in CoCoA, an instruction like `NF(NEG(`α`),ideal)`. Then the answer is "yes" if and only if the output of the computer algebra system is "0".
6. Curiously, but very important, verification (checking for the existence of inconsistencies in the ES), can also be calculated using GB just typing an instruction as `GBasis(ideal)`: the system is inconsistent if and only if the output is "1". The reason is that if "1" belongs to the ideal, then the ideal is the whole ring and, consequently *any formula* (enclosed in the infinite set at the left side of Figure 1) written in the language the ES is expressed, follows from the ES, which means, as referred to above, that the ES is inconsistent. We have constructed a simple program that, in case of inconsistency, incrementally detects which rule produces the inconsistency. We have successfully applied this verification method to the detection of inconsistencies in medical "appropriateness criteria" [30]. For instance in the case of coronary diseases, we were told that he researchers of RAND Corporation, the leading group in studying appropriateness criteria, were verifying systems manually. Ours is, as far as we we know, the first method that detects inconsistencies automatically.

We have also applied Gröbner bases to other fields, as decision taking in railway interlockings [52, 53] and in airport surface movement guide and control systems (SMGCS) [54]; automatic theorem proving and discovery in geometry [55, 56, 57, 58], etc.

4.1.1 A note on George Boole

As a matter of interest, let us say a few words about the background of the theorem in this section. In addition to Stone's and Moisil's XX century work on Boolean rings, we have found that George Boole, in a time (first half part of the XIX century) where ideal machines didn't exist yet (although Babbage had developed a complex calculator), was a pioneer on modern symbolic computation, and a forerunner of the theorem that links tautological consequence in logic with an ideal membership problem referred to in Section 4.1 above.

Boole says in page 55 of *The Mathematical Analysis of logic* [7]: "The treatment of every form of hypothetical Syllogism will consist in forming the equations of the premises, and eliminating the symbol or

symbols which are found in more than one of them. The result will express the conclusion.". Let us see an example (page 56, 5th example):

$$\begin{gathered}\text{If X is true, Y is true: } x \cdot (1-y) = 0,\\ \text{If W is true, Z is true: } w \cdot (1-z) = 0,\\ \text{Either X is true or W is true, } x + w - x \cdot w = 1,\end{gathered}$$

From these equations, *eliminating* w we have: $y - y \cdot z = 1$, which expresses the conclusion, in Boole's words, "Either Y is true, or Z is true, the members being non-exclusive".

Boole calls "elimination" to the following process: given

$$\begin{gathered}ay + b = 0\\ a'y + b' = 0,\end{gathered}$$

multiply the second equation by a and the first by a', and perform the substraction, obtaining

$$ab' - a'b = 0.$$

Note that *the polynomial translation of the negation* (Boole negates an expression by making it equal to 0) *of the conclusion results to be an algebraic combination of the polynomial translation of the negation of the premises*, a fact which shows Boole as a forerunner of the ideas underlying the theorem commented in Section 4.1.

4.2 Herbert Simon

4.2.1 Machines can discover

Herbert A. Simon (under whose supervision the first and third authors of this paper jointly worked for a time), together with Al Newell and Cliff Shaw, begun, some 50 years ago, "plunging into the exhilarating waters of Artificial Intelligence" [63]. They called this new science "Complex Information Processing". The name changed to "Artificial Intelligence" [42], a term introduced by John McCarthy by the same time. We shall focus our attention only on Herbert Simon, who was Professor at Carnegie-Mellon University in both the Departments of Computer Science and Psychology. He was awarded the Nobel Prize in Economics, but his most original work, for which he was most appreciated, was Artificial Intelligence in the questions of: can machines make scientific discovery?, can they make analogies, have intuitions, insight and inspiration?. He was actually looking for a scientific language that could be the language of theories of human thought, like differential equations language was the language of (parts of) physics. He thought that such a language was the subset of Computer Languages with ability to handle symbols of any kind. He and his team published works in his and related topics, [26, 33, 38, 39, 40, 60], a work in which we have collaborated [35, 36, 20] and [32].

In regards to scientific discovery, Simon and his collaborators, presented, among many others, a program called BACON, that discovers experimental laws in physics. Let us describe it in a simplified version.

Let us see how Bacon "discovers" Kepler's third law and how the same program, applied to other astronomy problems and to other fields as physics, happens to discover, by a kind of analogy, laws in those fields.

BACON consists first of a table that introduces objects (planets) and magnitudes (in this case, distances, "D", from the planets to the Sun and the periods "P" or time that the planets take in a complete round around the Sun). BACON also contains a set of production rules that translate strategies for finding regularities involving D and P. The finding of regularities is the aim of most scientific researches. Auxiliary production rules as those defining mathematical operations must also be introduced, among them two simple operations: "ratio" and "product").

Among the production rules in BACON, we only refer to four, named "INCREASING", "DECREASING" "LINEAR" and "CONSTANT", because they are the most intuitive ones.

Johannes Kepler [67] was born in 1571. His master, Maestlin, used to teach both geocentric and heliocentric systems, although Kepler was a convinced heliocentrist. He worked during long periods of time with Tycho Brahe, a great observer of the heavens who had collected thousands of valuable data. It seems that Kepler used Brahe's information to assert his first two laws, but for the third one he used his own observations and these were only twelve! The friendship, as well as the cooperation between Brahe and Kepler deteriorated. Brahe died poisoned: by Kepler? Kepler died in Ratisbone in 1630, after having published an immense treatise based on the heliocentric theory, with a great amount of data about the planetary system.

Kepler's third law asserts $\frac{D^3}{P^2} = \text{constant}$. "$D$" represents the distance from the planet to the Sun and "P" represents the time the planet takes to do one round to the Sun.

The four production rules above mentioned are the following (they have been taken from [34], pages 76 onwards):

- INCREASING

  IF you want to find laws,

  And you have recorded a set of values for the term X (example: *Distance*),

  And you have recorded a set of values for the term Y (example: *Period*),

  And the absolute values of X increases as the absolute values of Y increases and these values are not linearly related

  THEN consider the ratio of X and Y.

- DECREASING

  IF you want to find laws,

  And you have recorded a set of values for the term X (example: *Distance*),

  And you have recorded a set of values for the term Y (example: *Period*),

  And the absolute values of X increase as the absolute values of Y decrease and these values are not linearly related,

  THEN consider the product of X and Y.

- LINEAR

  IF you want to find laws,

  And you have recorded a set of values for the term X (example: distance),

  And you have recorded a set of values for the term Y (example: period),

  And the values of X and Y are linearly related with slope M and intercept B,

  THEN infer that a linear relation exists between X and Y with slope M and intercept B.

- CONSTANT

  IF you want to find laws,

  And the dependent term D (final relation between X and Y) has value V in all data clusters,

  THEN infer that D has always value V.

Planet	Distance (D)	Period (P)	Term 1 (D/P)	Term 2 (D^2/P)	Term 3 (D^3/P^2)
A	1,0	1,0	1,0	1,0	1,0
B	4,0	8,0	0,5	2,0	1,0
C	9,0	27,0	0,333	3,0	1,0

Table 1. Objects, magnitudes and terms for three imaginary planets.

In Table 1, three imaginary planets A, B and C are considered.

Distance "D" to the Sun of each one of these planets appears in the second column, observe that "D" increases.

Period "P" of each one of these planets appears in the third column. Observe that "D" also increases.

As D and P increase together, production rule "INCREASING" says "perform ratio" $\frac{D}{P}$, which is Term 1 of the fourth column.

As D increases and $\frac{D}{P}$ decreases, production rule "DECREASING" says "perform product" $\frac{D^2}{P}$, which is Term 2 of the fifth column.

As $\frac{D}{P}$ decreases and $\frac{D^2}{P}$ increases, production rule "DECREASING" says "perform product" $\frac{D^3}{P^2}$, which is Term 3 of the sixth column.

As $\frac{D^3}{P^2}$ is constant for A, B and C, production rule "CONSTANT" says "make $\frac{D^3}{P^2}$ equal to constant".

But $\frac{D^3}{P^2} = \text{constant}$ is, precisely, Kepler's third law.

4.3 Machines make analogies and discover regularities

Production rules INCREASING, DECREASING and CONSTANT, as well as all others in the BACON program are high level statements that are expressed as symbolic production rules in some program and are so introduced into the computer. The program follows a process of discovery by applying the appropriate instruction in each step. Both the fact that Kepler only based his third law on twelve observations and the simplicity of the program might suggest that he, in fact, might have followed the instructions of BACON without having them written down as a "program".

But a most interesting fact is that, once BACON was written down and run on Kepler's third law, Simon, Langley and the other designers of BACON, realized the following.

1. By introducing distance D and time T, some laws of the uniformly accelerated movement, as Earth gravitational acceleration, $9.8 = \frac{D}{T^2}$, is obtained.

2. By introducing the distance D of a satellite of Jupiter and its period P, Borelli's law was found.

3. By introducing the length L of an electric wire, the resistance r and the intensity I and potential difference v of a current, by using the production rule "LINEAR", an Ohm's-like law was obtained: $IL = -rI + v$.

4. By introducing the volume V and pressure P of a gas, Boyle's law $P \cdot V = \text{constant}$ was obtained.

5. But, more importantly, qualitative laws, i.e., new perspectives, were discovered: a step further than merely quantitative laws.

Note that in Table 1 the data were adjusted knowing Kepler's third law. If we worked with real data, the last column would be formed by values only close to 1.0. Nevertheless, BACON system is also able to find the right rules in such cases. This is the case above for Borelli's and Boyles' laws.

This made Simon and his collaborators think that machines may perform activities such as pattern discovery (and learning by analogy), processes which are very human.

Nevertheless, these patterns and analogies discovered by machines weren't possible if a human wouldn't have introduced the right data. The question is that we, humans, as well as some animals, learn patterns and analogy maybe by an innate process but necessarily improved by learning from others (i.e., programmed by others).

The following are works on machine systems that simulate human behavior:

4.4 Machines have intuition, inspiration, insight

In the already mentioned article, "Explaining the Ineffable" [63], Simon presented a curious problem named "the mutilated checkboard". Recent works, with substantial answers to it, have been recently published [18, 19, 20].

The aim of these articles is to show the interaction between Artificial Intelligence and Cognitive Science Theory. These type of studies has resulted in new conceptions about the possibilities of a computer simulation of human intelligent attitudes. Furthermore, these simulations (the already existing and the ones to be developed in the future) are pieces of a theory of Cognitive Psychology.

Simon was involved on showing that there are computer programs that, in many senses, may reflect human behaviour such as intuition, inspiration and insight [63]. Let us focus our attention only on insight.

To consider and define insight in such a way that it may be said that machines behave in a insightful way, Simon and the mentioned authors intended both to give an acceptable characterization of insight and some clues about how insight is attained. This is better understood by considering the experiment of the mutilated checkboard problem [63].

A group of persons, having no prior knowledge of the problem, is provided with both a chess table ($8 \times 8 = 64$ squares), two of whose diagonally opposite corners have been removed and dominoes. The experiment consists of asking the people in the group to totally cover the mutilated checkboard with dominoes, with the condition that each of the dominoes can only cover two adjacent squares of the board. The experiment is supervised by a trained interviewer.

Through experiments, it was seen that the resolution of the problem started being irregular, by tries at random to cover the checkboard, and, many times, depending on the hints given by the interviewer.

After such a blind try, some people abandoned and many of them needed a rest. But in most cases, Simon and Kaplan checked that some people had suddenly found some regularity, suggesting that the solution of the problem was near.

The guessed solution to the problem is that it is not possible to cover the mutilated checkboard with dominoes because each of these covers just two squares, one black and the other white, but in the mutilated checkboard, the number of squares black and white is not the same.

Thus, the experiment ends when one or more persons of the group find the regularity (it is actually an invariant) provided by the properties of the dominoes and the properties of the mutilated checkboard. This finding, usually sudden, is one of the key elements in the insight definition.

The nice fact is that this process of "having insight" can be programed, as shown by Hernando, de Ledesma and Laita [19]. This suggests that machines (computer programs) may emulate a human power as insight.

As already said, we have also constructed an "expert system "BOOLE 2"", that in some ways simulates the intuition, inspiration and insight that George Boole had when developing his algebraic logic [32].

4.4.1 A brief addendum

As mentioned in page 136, today's computation both molecular based on ADN and cellular with membranes [47, 48], open new approaches to the idea of computation and of complexity of problems.

Something different might occur with quantum computation, which uses "qubits" instead of today's computers "bits". Some elementary quantum machines, with a few qubits have already been constructed. If powerful quantum machines are constructed in the future, a 250 qubits machine could make 10^{75} simultaneous computations, something that might change our idea of what machines can do. Quantum mechanics

asserts the crucial influence of the observer on the experiments: would this mean that the user cannot interact with the computer while the program is running? Probably a new logic of quantum computation is needed, a logic that would embody relativists concepts. A quantum Church-Turing thesis would also be necessary.

5 Last comment: are machines more intelligent than humans?

The first conclusion of this study about what machines can and cannot do, is that one cannot answer this question in a clear and definite way [27].

Machines have more calculating power, they play chess much better than most humans and, above all, they emulate or simulate what are considered typical human abilities, as to make scientific discoveries, simulate insight and so on.

But, in all cases, machines do all of this because they have been programmed by a human, similarly, somewhat, as senior researchers teach doctoral students. Moreover, the "discoveries" and "insight" of machines, as those mentioned in this paper, are very limited types of discoveries and insight. The reader is recommended to read the classic [44].

Some human characters seem to be difficult to simulate by machines, for instance to have or feel hope for desirable things.

We all have hopes and immediately design a more or less informal "algorithm" to get these hopes. When talking of "hopes" we are referring, not only to just every day hopes, but deep hopes as, for instance, those that scientists have. Let us refer briefly, as an illustration, to the hopes the great physicist Stephen Hawking [17] has about finding a unified theory that would avoid the contradictions (or rather, the incompatibilities) between the approaches to an explanation of the world, cosmic and atomic, of relativistic and quantum mechanics. He and others scientist have already developed calculi, as the theories of chords and of p-branes, approaching a unified theory of gravitation.

By establishing a (probably week) parallelism with what has been said in this paper, from the algorithm designed to reach the goal of some hope and by the Church-Turing Thesis, we would get a computation, that is, a machine that would give an effective and finite procedure to achieve the goal of a particular hope. But the experience shows that both our small hopes, as well as the great ones, rarely end being a "computation", both because the reality breaks the process of making real our little hopes or because new paradigms related with the conception of the world make unrealistic the existence of machines that efficiently and forever would show us how to accomplish the great hopes. Could a machine be designed that would emulates the process of having hopes, trying to obtain the objects of these hopes and end with an output that says: "hope partially or not accomplished, search for a new one"?; surely not at the present time.

But we do not know what, in this regard, will happen in future, first because of the particular characteristics of quantum computation which embodies a great deal of uncertainty and dependence from the human processing the computation and, second (and more importantly), because new strategic and economic reasons rather than absolute ones, may change our present view of machines limitations.

In addition to hopes, there are of course plenty of tasks performed by humans, so far unable to be performed by machines, like the mathematization of a problem expressed in natural language. But recall, for instance, that some mathematical problems, as Kepler's experiment mentioned above, need man-machine cooperation and that, in many instances, user interaction with the machine (recall what was said regarding verification and validation of algorithms), is actually viewed as more useful, thus, embodying a change in the mathematical methodology paradigm.

Anyway, the most realistic answers to the question: who are more intelligent, machines or humans?, are the following two ones. First the one given by Simon: *when facing a task*, who would get it better? and, second, that this question is today (we do not know for how much time ahead), senseless: instead of asking, at the present time, who is more intelligent, try guessing: *how the pair man-machine will evolve?*. Humans have always evolved by using artifacts, but today have an admirable artifact: the computer. The evolution man-machine is no more as predictable as it was with the artifacts of the past.

Acknowledgement. We acknowledge the comments of professors José Antonio Alonso and Mario Pérez-Jiménez (Universidad de Sevilla). Their comments have helped us to improve the paper.

We would also like to thank the support given by Professor Manuel López Pellicer: he both invited us to publish the article in RACSAM and patiently accepted the changes we made in successive versions of the paper.

This work was partially supported by the research projects *MTM2004–03175* (Ministerio de Educación y Ciencia, Spain) and *UCM2005–910563* (Comunidad de Madrid – Universidad Complutense de Madrid, research group *ACEIA*).

References

[1] Adams, W. W. and Loustaunau, P., (1994). *An Introduction to Gröbner Bases*, Graduate Studies in Mathematics, American Mathematical Society, Providence, RI.

[2] Adleman, L., (1994). Computation of Solutions to Combinatorial problems, *Science*, **268**, 1021–1024.

[3] Adleman, L., (1996). On Constructing a Molecular Computer, in *DNA based Computers*, Lipton R. J. and Baum, E. B. eds., American Mathematical Society,

[4] Alonso, J. A., Briales, E. and Riscos, A., (1990). Preuve Automatique dans le Calcul Propositionnel et des Logiques Trivalentes, in *Proceedings of the Congress on Computational Geometry and Topology and Computation* (Universidad de Sevilla, Seville, Spain), 15–24.

[5] Arbib, M., (1976). *Cerebros, Máquinas y Matemáticas*, Alianza editorial, Madrid.

[6] Bernstein, E. and Vazirani, U., (1997). Quantum Complexity Theory, *SIAM J. of Computing*, **26**, 5 1411–1473.

[7] Boole, G., (1847). *The Mathematical Analysis of Logic, Being an Essay towards a Calculus of Deductive Reasoning*, Cambridge and London. Reprinted in *George Boole, Studies in logic and probability*, R. Rhees ed., Watts and Co., London, 1952, 49–124.

[8] Boolos, G. S. and Jeffrey, R. C., (1982). *Computability and Logic*, Cambridge University Press, Cambridge,

[9] Buchberger, B., (1965). *An Algorithm for Finding a Basis for the Residue Class Ring of a Zero-Dimensional Polynomial Ideal*, Ph. D. Thesis (in German), Math. Institute - University of Innsbruck, Insbruck, Austria,

[10] Buchberger, B., (1965). Applications of Gröbner Bases in Non-Linear Computational Geometry, in *Mathematical Aspects of Scientific Software*, J. R. Rice ed., Springer-Verlag, IMA Vol. **14**, New York, NY, 60–88.

[11] Capani, A. and Niesi, G., (1996). *CoCoA User's Manual, v. 3.0b.*, Dept. of Mathematics - University of Genova, Genova, Italy.

[12] Chazarain, J., Riscos, A., Alonso, J. A. and Briales, E., (1991). Multivalued Logic and Gröbner Bases with Applications to Modal Logic, *J. of Symbolic Computation*, **11**, 181–194.

[13] Cutland, N. J., (1980). *Computability: an Introduction to Recursive Function Theory*, Cambridge University Press, Cambridge.

[14] Dedekind, R., (1901). *Essays on the Theory of Numbers*, Dover Publications, 1963. Reprint of Open Court Pub. Co.

[15] Deutsch, D., (1985). Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer, *Proceedings of the Royal Society Ser. A*, **A400**, 97–117.

[16] Devlin, K., (1998). Kepler's Sphere Packing Problem Solved, in *MAA Online* (The Mathematical Association of America). URL: `http://www.maa.org/devlin_9_98.html`

[17] Hawking, S., (2002). *El mundo en una cáscara de nuez*, Editorial Planeta, Colección Crítica, Barcelona.

[18] Hernando, A., de Ledesma, L. and Laita, L. M., (2005). El problema del tablero mutilado, una resolución a través de perspicacia (insight), *Bol. Soc. "Puig Adam" de Profesores de Matemáticas*, **70**, 72–79.

[19] Hernando, A., de Ledesma, L. and Laita, L. M., (2003). Towards a Theory of Insight, in *Proceedings of the 12th IASTED Conference on Applied Simulation and Modelling*, Ed. Hamza, 181–186.

[20] Hernando A., de Ledesma, L., Laita, L. M., A System simulating representation Change Phenomena while Problem Solving, *Mathematics and Computers in Simulation*, (accepted for publication).

[21] Hofstadter, D. R., (1979). *Gödel, Escher, Bach: an Eternal Golden Braid*, Basic Books.

[22] Hsiang, J., (1985). Refutational Theorem Proving using Term-Rewriting Systems. *Artificial Intelligence*, **25**, 255–300.

[23] Jha, A. et al., (2003). La Informática inteligente, newspaper *El Mundo*, December 22nd , p. 28.

[24] Kapur, D. and Narendran, P. (1984). *An Equational Approach to Theorem Proving in First–Order Predicate Calculus*. 84CRD296, General Electric Corporate Research and Development Report, Schenectady, NY, March 1984, rev December 1984. Also in: Proceedings of IJCAI-1985; 1446–1456

[25] Kuhn, T., (1970). *The structure of Scientific Revolutions*, Chicago University Press, –with a postscript. First published in 1962.

[26] Kulkarni, D. and Simon, H. A., (1988). The Processes of Scientific Discovery, the Strategy of Experimentation, *Cognitive Science*, **12**, 139–175.

[27] Laita, L. M., (2006). *Algunas consideraciones acerca de la pregunta; ¿llegarán a ser algún día las máquinas más inteligentes que los seres humanos?*. Lección inaugural de curso 2006–07 de la Universidad San Pablo CEU (Fundación de la Universidad San Pablo CEU, Madrid).

[28] Laita, L. M., Roanes-Lozano, E., de Ledesma, L. and Alonso, J. A., (1999). A Computer Algebra Approach to Verification and Deduction in Many-Valued Knowledge Systems, *Soft Computing*, **3**, 1, 7–19.

[29] Laita, L. M. and de Ledesma, L., (1997). Knowledge Based Systems Verification, in *Encyclopedia of Computer Science and Technology*, A. Kent and J. G. Williams eds., Vol. **36**, Marcel Dekker, New York, 253–280.

[30] Laita, L. M., Roanes-Lozano, E., Maojo, V., de Ledesma, L. and Laita, L., (2000). An Expert System for Managing Medical Appropriateness Criteria Based on Computer Algebra Techniques, *Computers and Mathematics with Applications*, **51**, 5, 473–481.

[31] Laita, L. M., Roanes-Lozano, E. and Alonso, J. A. (eds.), (2004). Symbolic Computation in Logic and Artificial Intelligence (special issue). *RACSAM*, Revista de la Real Academia de Ciencias, Series "A" of Mathematics **98**, 1–2.

[32] Laita, L. M., de Ledesma, L. and Roanes-Lozano, E., (2005). The genesis of Boole's Logic; its History and a Computer Exploration, *Memorias de la Real Academia de Ciencias, Serie de Ciencias Exactas*, Vol. **XXXIII**, Madrid.

[33] Langley, P. W., Zytkow, J. M., Simon, H. A. and Bradshaw, G. L., (1986). The Search for Regularity: four Aspects of Scientific Discovery, in *Machine Learning: An artificial intelligence approach*, Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. eds., Vol. **2**, Morgan Kaufmann, San Mateo, CA, 425–469.

[34] Langley, P. W., Zytkow, J. M., Simon, H. A. and Bradshaw, G. L., (1987). *Scientific Discovery, Computational Explorations of the Creative Processes*, MIT Press, Cambridge, MA.

[35] de Ledesma, L., Laita, L. M., Aurora Pérez, A. and Borrajo, D., (1993). Descubrimiento científico e inteligencia artificial, in *Segundo curso de conferencias sobre inteligencia artificial: teoría y práctica*, organized by Darío Maravall–Casesnoves, Real Academia de Ciencias (ed.), Madrid, 115–132.

[36] de Ledesma, L., Pérez, A., Borrajo, D. and Laita, L. M., (1995). Theory-driven historical discovery: Boole's abstract formalization of Logic, in: *Working Notes of the AAAI Spring Symposium Series, Systematic Methods of Scientific Discovery*, Stanford, 60–65.

[37] de Ledesma, L., Pérez, A., Borrajo, D. and Laita, L. M., (1997). A Computational Approach to George Boole's Discovery of Mathematical Logic, *Artificial Intelligence*, **91**, Simon, H. A., Valdés–Pérez, R. and Sleeman, D. H. guest eds., 281–307.

[38] Lenat, D. B., (1977). Automated Theory Formation in Mathematics, in: *Proceedings of the 5th IJCAI*.

[39] Lenat, D. B., (1982). The Nature of Heuristics, *Artificial Intelligence*, **19**, 189–249.

[40] Lenat, D. B., (1983). Eurisko: A Program that Learns new Heuristics and Domain Concepts, *Artificial Intelligence*, **21**, 61–98.

[41] Lipton, R. J., (1996). Speeding up Computations via molecular Biology, in *DNA based Computers*, Lipton R. J. and Baum, E. B. eds., American Mathematical Society.

[42] Luger, G. F. and Stubblefield, W. A., (1998). *Artificial Intelligence, Structures and Strategies for Complex Problem Solving*, Addison Wesley Longman, Inc., Harlow, England.

[43] Mendelson, E., (1964). *Introduction to Mathematical Logic*, D. Van Nostrand, Princeton, New Jersey.

[44] Minsky, M., (1963). Can Machines Think?, in *Computers and thought*, Fiegenbaum and Feldman eds., McGraw-Hill.

[45] Mosterín. J., (2000). *Los Lógicos*, Espasa Calpe S. A., Madrid.

[46] O'Connor, J. J. and Robertson, E. F., *MacTutor History of Mathematics*, November 2004.
URL: `http://www-history.mcs.st-andrews.ac.uk/history/index.html`

[47] Pérez-Jiménez, M. and Riscos-Núñez, A., (2004). *Modelos de Computación Molecular, Celular y Cuántica*, Fénix Editorial, Sevilla.

[48] Pérez-Jiménez, M., Romero-Jiménez, A. and Sancho-Caparrini, F. eds., (2004). *Recent Results in Natural Computing*, Fénix Editorial, Sevilla.

[49] Perkinson, D., (2000). *CoCoA 4.0 Online Help*, electronic file acompanying CoCoA v.4.0,

[50] Popper, K., (1959). *Logic of Scientific Discovery*, Hutchinson, London, Translated from: *Logic der Forschung*, Julius Springer Verlag, Vienna, 1935.

[51] Roanes-Lozano, E., Laita, L. M. and Roanes Macías, E., (1998). A Polynomial Model for Multivalued Logic, with a Touch of Algebraic Geometry and Computer Algebra, *Mathematics and Computers in Simulation*, **45**, 1, 175–184

[52] Roanes-Lozano, E., Laita, L. M. (1998). An Applicable Topology independent Model for Railway Interlocking Systems, *Mathematics and Computers in Simulation*, **45**, 1, 83–99.

[53] Roanes-Lozano. E. and Laita, L. M., (2002). Railway Interlocking Systems and Gröbner Bases, *Mathematics and Computers in Simulation*, **58**, 203–214.

[54] Roanes Lozano, E., Muga, R., Laita, L. M. and Roanes Macías, E., (2005). A terminal area topology-independent GB-based conflict detection system for A-SMGCS, *RACSAM* (Revista de la Real Academia de Ciencias, Serie A, Matemáticas), **8**, 1–2 214–229.

[55] Roanes-Macías, E. and Roanes-Lozano, E., (1994). *Nuevas Tecnologías en Geometría*, Editorial Complutense, Madrid.

[56] Roanes-Lozano, E. and Roanes-Macías, E., (1996). Automatic Theorem Proving in Elementary Geometry with DERIVE 3, *The Intl. DERIVE J.*, **3**, 2, 67–82.

[57] Roanes-Macías, E. and Roanes-Lozano, E., (2001). Automatic determination of geometric loci. 3D-Extension of Simson-Steiner Theorem, in: *Artificial Intelligence and Symbolic Computation. Procs. Intl. Conf. AISC 2000*, Campbell, J. A. and Roanes-Lozano, E. eds., Springer LNCS 1930, Berlin-Heidelberg, 157–173.

[58] Roanes-Macías, E. and Roanes-Lozano, E., (2004). A Completion of Hypotheses Method for 3D-Geometry. 3D-Extensions of Ceva and Menelaus Theorems, in *Proceedings of* 20^{th} *European Workshop on Computational Geometry*, Díaz-Bánez, J. M., Márquez, A. and Portillo, J. R. eds., Universidad de Sevilla, Seville, 85–88.

[59] Schwartz, P., (2007). *En busca de Montesquieu. La democracia en peligro*, Encuentro, Madrid.

[60] Shrager J. and Langley, P., (1990). *Computational Models of Scientific Discovery and Theory Formation*, Morgan Kaufmann.

[61] Shepherdson, J. C. and Sturgis, H. E., (1963). Computability of recursive Functions, *Journal Assoc. for Computing Machinery*, **10**, 217–255.

[62] Shoenfield, J. R., (1967). *Matemátical Logic*, Addison Wesley, Reading, Massachussets.

[63] Simon, H. A., (1996). Explaining the Ineffable: AI on the topics of Intuition, Insight and Inspiration, in: *Proceedings of the 14th IJCAI*, 839–846.

[64] Trillas, E., (1998). La Inteligencia Artificial, in: *Máquinas y Personas*, J. M. Sánchez Ron (series ed.). Temas de Debate, Madrid.

[65] Turing, A. M., (1948). Intelligent Machinery, report for the National Physical Laboratory, in *Machine Intelligence*, **5**, Meltzer, B. and Michie, D. eds., Edinburgh University Press, New York (1969), 3–23.
Also in: URL: `http://www.alanturing.net/intelligent_machinery/`

[66] Winkler, F., (1996). *Polynomial Algorithms in Computer Algebra*, Springer-Verlag, Vienna,

[67] —-, Johannes Kepler, *Enciclopedia Gran Larousse Universal*, Vol. **23**, Ed. Plaza y Jané, Barcelona, 1979, 7314–7315.

[68] —-, The Church–Turing Thesis, *Stanford Enciclopedia of Philosophy*.
URL: `http://plato.stanford.edu/entries/church--turing`. First published January 1997, substantive revision 2002.

[69] URL: `http://cocoa.dima.unige.it`

[70] URL: `http://www.bookrags.com/Church`

Luis M. Laita
Emeritus Professor of Computer Sciences and Artificial Intelligence
Artificial Intelligence Dept.,
Computer Science School,
Univ. Politécnica de Madrid
Corresponding Academician of Madrid Royal Academy of Sciences.

Eugenio Roanes-Lozano
Associate Professor of Algebra,
Algebra Dept.,
School of Education,

Univ. Complutense de Madrid

Luis de Ledesma Otamendi
Professor of Computer Sciences and Artificial Intelligence,
Artificial Intelligence Dept.,
Computer Science School,
Univ. Politécnica de Madrid