Gödel and Computability [1]

Luís Moniz Pereira

Centro de Inteligência Artificial – CENTRIA
Universidade Nova de Lisboa, Portugal

Abstract

We discuss the influence of Gödel and his results on the surfacing of the rigorous notion of computability afforded by Turing. We also address the debate about the limits of Artificial Intelligence spurned by Roger Penrose, on the basis of Gödel's theorems, and the views of Gödel himself. We conclude by touching upon the use of logic as a tool with which to approach the description of mind.

Hilbert's Questions

The research programme that David Hilbert presented to the international congress of mathematics in Bologna in 1928 was essentially an extension of the work he had initiated in the 1890s. He was not intent on answering the question that Frege and Russell had posed, about what mathematics *really was*. In that regard, he showed himself to be less philosophical and less ambitious. On the other hand he was much more penetrating, as he raised profound and hard questions *about* the systems of the kind that Russell had produced. In fact, Hilbert begot the question about what were, in principle, the limitations of a scheme like *Principia Mathematica*. Was there any way to find out what could, and could not, be demonstrated from within such theory?

Hilbert's approach was deemed *formalist*, as he treated mathematics as a game, a matter of form. The allowed steps in a demonstration should be looked at in the same way as the possible moves in the game of chess, with the axioms being the initial state of the game. In this analogy, 'playing chess' stood for 'executing mathematics', but the *statements* about chess (such as 'two knights cannot achieve checkmate') stood for statements *about* the scope of mathematics. It was with the latter statements that Hilbert's programme was concerned.

In that congress of 1928, Hilbert made his questions very precise. First, was mathematics *complete*? In the sense of whether all mathematical statements (such as "all integers are the sum of four square numbers") could be demonstrated, or falsified. Second, he asked whether mathematics was *consistent*, which amounts to asking whether any contradictory statements, like '2+2=5' could be demonstrated by a correct application of valid steps of the rules for derivation. And third he asked was mathematics *decidable*? By this he meant to ascertain if there was a well-defined method which, in principle, if applied to any assertion, could be guaranteed to produce a correct decision about the truth of that assertion.

---

[1] Translation of an invited article for a special issue of the "Bulletin of the Portuguese Mathematical Society" dedicated to Kurt Gödel, to appear end of 2006 on the occasion of the 100th anniversary of his birth.

In 1928, none of these questions had an answer. Hilbert hypothesized that the answer was 'yes' to every one of them. In his opinion, "there were no unsolvable problems". But it didn't take long before the young Austrian mathematician, Kurt Gödel, presented results which struck a severe blow to Hilbert's programme.

Gödel's Answers

Kurt Gödel was able to show that every formalization of arithmetic must be *incomplete*: that there are assertions that cannot be demonstrated nor rebutted. Starting from Peano's axioms for the integer numbers, he extended them to a simple type theory, in such a way that the obtained system could represent sets of integers, sets of sets of integers, and so forth. However, his argument could be applied to any formal system sufficiently rich to include number theory, and the details of the axioms were not crucial.

Next he showed that all operations in a 'proof', the 'chess-like' deduction rules, could be represented in arithmetic. This means that such rules would be constituted only by operations like counting and comparing, in order to verify if an expression (itself represented arithmetically) had been correctly replaced by another – just like the validity check of a chess move is only a matter of counting and comparing. In truth, Gödel showed that the formulas of his system could be codified as integers, to represent assertions *about* integers. This is his main idea. Subsequently, Gödel showed how to codify demonstrations as integers, in such a way that he had a whole theory of arithmetic, codified *from within* arithmetic. He was using the fact that, if mathematics was to be regarded as a pure symbol game, then one could employ numerical symbols instead of any others. He managed to show that the property of 'being a demonstration' or being 'provable' was as much arithmetical as that of 'being square' or 'being prime'.

The result of this codification process was that it became possible to write arithmetical assertions that referred to *themselves*, just like when a person says "I'm lying". In fact, Gödel constructed a specific assertion that had exactly that property, as it effectively said "This assertion is not provable." It followed that it could not be shown *true*, as that would lead to a contradiction. Neither could it be shown *false*, by the same reason. It was thus an assertion that could neither be proved nor refuted by logical deduction from the axioms. Gödel had just showed that arithmetic was *incomplete*, in Hilbert's technical sense.

But there is more to be said about the matter, because there was something special about that particular assertion of Gödel, more precisely the fact that, considering that it is not provable, it is in some sense *true*. But to assert that it is *true* requires an observer that can look at the system from the outside. It cannot be shown *from within* the axiomatic system.

Another important issue is that this argument assumes that arithmetic is *consistent*. In fact, if arithmetic were inconsistent then *any* assertion would be provable, since in elementary logic everything follows from contradiction. More precisely, Gödel had just shown that arithmetic, once formalized, had to be either *inconsistent* or *incomplete*[2]. He could also show that arithmetic could not be proved consistent from within its own

[2] In truth, the fact that a Gödel assertion cannot be refuted formally depends on the omega-consistency of the system, not just on its consistency. Rosser, later, by means of a slightly different kind of assertion, showed that one can do without the omega-consistency requirement.

axiomatic system. To prove consistency, it would suffice to show that there was some proposition (say, '2+2=5') that could not be proven true.

But Gödel managed to show that such an existential statement had the same characteristic of the assertion that claimed its own non-provability. In this way, he «discharged» both the first and second questions posed by Hilbert. Arithmetic *could not* be proved consistent, and for sure it *wasn't* consistent *and* complete. This was a spectacular turn of events in research on the matter, and quite disturbing for those that wished for an absolutely perfect and unassailable mathematics.

Gödel, computability, and Turing

The third of Hilbert's questions, that of *decidability*, still remained open, although now it had to be formulated in terms of *provability*, instead of *truth*. Gödel's results did not eliminate the possibility that there could be some way of distinguishing the provable from the non-provable assertions. This meant that the peculiar self-referencing assertions of Gödel might in some way be separated from the rest. Could there be a well-defined *method*, i.e. a *mechanizable* procedure, that could be applied to any mathematical statement, and that could decide if that statement was, or was not, derivable in a given formal system?

From 1929 to 1930, Gödel had already solved most of the fundamental problems raised by Hilbert's school. One of the issues that remained was that of finding a precise concept that would characterize the intuitive notion of computability. But it was not clear at the time that this problem would admit a definitive answer. Gödel was probably surprised by Turing's solution, which was more elegant and conclusive than what he had expected. Gödel fully understands, at the beginning of the '30s, that the concept of formal system is intimately tied up with that of mechanizable procedure, and he considers Alan Turing's 1936 work (on computable numbers) as an important complement of his own work on the limits of formalization.

In 1934, Gödel gave lectures at the Institute of Advanced Studies in Princeton where he recommended the Turing analysis of mechanizable procedures (published in 1936) as an essential advance that could raise his incompleteness theorems to a more finished form. In consequence of Turing's work, those theorems can be seen to "apply to *any* consistent formal system that contains part of the finitary number theory".

It is well known that Gödel and Alonzo Church, probably in the beginning of 1934, discussed the problem of finding a precise definition to the intuitive concept of computability, a matter undoubtedly of great interest to Gödel. In accordance with Church, Gödel specifically raised the issue of the relation between that concept and Church's definition of *recursivity*, but he was not convinced that both ideas could be satisfactorily made equivalent, "except heuristically". Over the years, Gödel regularly credited Turing's 1936 article as the definitive work that captures the intuitive concept of computability, he being the only author to present persuasive arguments about the adequacy of the precise concept he himself defined.

Regarding the concept of mechanizable procedure, Gödel's incompleteness theorems also naturally begged for an exact definition (as Turing would come to produce) by

which one could say that they applied to every formal system, i.e. every system on which proofs could be verified by means of an automatic procedure. In reality, Hilbert's programme included the *Entscheidungsproblem* (literally, 'decision problem'), which aimed to determine if there was a procedure to decide if, in elementary logic, any proposition was derivable or not by Frege's rules (for elementary logic, also known as first-order logic). This requires a precise concept of automatic procedure, in case the answer is negative (as is the case).
To this end, in 1934, Gödel introduces the concept of general recursive functions, which was later shown to capture the intuitive concept of mechanizable computability. Gödel suggested the concept and Kleene worked on it. The genesis of the concept of general recursive functions was implicit in Gödel's proof about the incompleteness of arithmetic. When Gödel showed that the concept of proof using "chess-like" rules was an 'arithmetical' concept, he was in fact saying that a proof could be realized by a 'well-defined' method. This idea, once formalized and somewhat extended, gave rise to the definition of 'recursive function'. It was later verified that these were exactly equivalent to the computable functions.

After some time, Gödel came to recognize that the conception of 'Turing machine' offers the most satisfactory definition of 'mechanic procedure'. According to Gödel himself, Turing's 1936 work on computable numbers is the first to present a convincing analysis of such a procedure, showing the correct perspective by which one can clearly understand the intuitive concept.

On the rigorous definition of the concept of computability performed by Turing, Gödel says: "This definition was by no means superfluous. Although before the definition the term 'mechanically calculable' did not have a clear meaning, though not analytic [i.e. synthetic], so the issue about the adequacy of Turing's definition would not make perfect sense, instead, with Turing's definition the term's clarity undoubtedly receives a positive answer".

Once the rigorous concept, as defined by Turing, is accepted as the correct one, a simple step suffices to see that, not only Gödel's incompleteness theorems apply to formal systems in general, but also to show that the *Entscheidungsproblem* is insoluble. The proof of this insolubility by Turing himself showed that a class of problems that cannot be solved by his "A-machines" (from "**A**utomatic machines", today known as Turing machines) can be expressed by propositions in elementary logic.

In his Ph.D. thesis, Turing is interested in finding a way out from the power of Gödel's incompleteness theorem. The fundamental idea was that of adding to the initial system successive axioms. Each 'true but not demonstrable' assertion is added as a new axiom. However, in this way, arithmetic acquires the nature of a Hydra, because, once the new axiom is added, a new assertion of that type will be produced that takes it now into consideration. It is then not enough to add a *finite* number of axioms, but it is necessary to add an infinite number, which was clearly against Hilbert's finitary dream. If it were possible to produce a finite generator of such axioms, then the initial theory would also be finite and, as such, subject to Gödel's theorem.

Another issue is that there exist an infinite number of possible sequences by which one can add such axioms, leading to different and more complete theories. Turing described his different extensions to the axioms of arithmetic as 'ordinal logics'. In these, the rule

to generate the new axioms is given by a 'mechanical process' which can be applied to 'ordinal formulas', but the determination whether a formula is 'ordinal' was *not* mechanizable. Turing compared the identification of an 'ordinal' formula to the work of intuition, and considered his results disappointingly negative, for although there existed 'complete logics', they suffered from the defect that one could not possibly count how many 'intuitive' steps were needed to prove a theorem[3].

This work, however, had a pleasantly persistent side effect, namely the introduction of the concept of 'oracle Turing machine', precisely so it could be allowed to ask and obtain from the exterior the answer to an insoluble problem from within it (as that of the identification of an 'ordinal formula'). It introduced the notion of relative computability, or relative insolvability, which opened a new domain in mathematical logic, and in computer science.

The connection, made by S.A. Cook, in 1971, between Turing machines and the propositional calculus would give rise to the study of central questions about computational complexity. In 1936, Gödel also noted that theorems with long proofs in a given system could very well obtain quite shorter proofs if one considered extensions to the system. This idea has been applied in Computer Science to demonstrate the inefficiency of some well known decision procedures, to show that they require an exponential (or worse) amount of time to be executed. This is a particular instance of one of Gödel's favourite general ideas: problems raised in a particular system can be handled with better results in a richer system.

Mens ex-machina

[3] I would like to make a reference here to my own ongoing work. The basic idea is to use an implemented logic programming language, EVOLP (Alferes et al. 2002), which has the capacity of *self-updating*, in order to obtain this self-evolution predicted by Turing.

The model theory can be defined with the revised stable models semantics (Pereira e Pinto, 2005), with the introduction in each program of the following causal scheme, involving the usual *not* construct of logic programming, representing non-provability:

GA ← gödel_assertion(GA), *not* GA

This scheme, representing its the *ground* instances, states that if GA is a Gödel assertion, and it is not demonstrable, then GA is true, where we take as a given the codification of the predicate gödel_assertion(GA), which tests, once codified, if the literal GA can be constructed using the Gödelization method. The conclusion is obtained by *reductio ad absurdum*, i.e. if *not* GA was true, then GA would be true; since this is a contradiction and we use a two-valued logic, then the premise *not* GA is false and the conclusion GA is true.

The proof system is derived from this intuition, and uses the following EVOLP rule, where the literal GA can itself be a full-blown rule:

assert(GA) ← gödel_assertion(GA), *not* GA

This means that each GA that complies with the premises will be part of the next state of the self-evolving program, and so forth. Such a role takes the place of the intuitive step necessary for evolution, conceptually realizing it by *reductio ad absurdum* followed by an *update*, which is allowed expression within the EVOLP language itself.

It is worth mentioning that, contrary to Turing, Gödel was not interested in the development of computers. Their mechanics is so connected with the operations and concepts of logic that, nowadays, it is quite commonplace for logicians to be involved, in some way or other, in the study of computers and computer science. However, Gödel's famous incompleteness theorem demonstrates and establishes the rigidness of mathematics and the limitations of formal systems and, according to some, of computer programs. It relates to the known issue of whether mind surpasses machine. Thus, the growing interest given to computers and Artificial Intelligence (AI) has lead to a general increase in interest about Gödel's own work. But, so Gödel himself recognizes, his theorem does not settle the issue of knowing if the mind surpasses the machine. Actually, Gödel's work in this direction seems to favour (instead of countering) the *mechanist* position (and even *finitism*) as an approach to the automation of formal systems.

Gödel contrasts *insight* with *proof*. A proof can be explicit and conclusive for it has the support of axioms and of rules of inference. In contrast, insights can be communicated only via "pointing" at things. Any philosophy expressed by an exact theory can be seen a special case of the application of Gödel's conceptual realism. According to him, its objective should be to give us a clear perspective of all the basic metaphysical concepts. More precisely, Gödel claims that this task consists in determining, through intuition, the primitive metaphysical concepts *C* and in making correspond to them a set of axioms *A* (so that only *C* satisfies *A*, and the elements in *A* are implied by the original intuition of *C*). He further admits that, from time to time, it would be possible to add new axioms.

Gödel also advocates an 'optimistic rationalism'. His justification appeals to (1) "The fact that those parts of mathematics which have been systematically and completely developed … show an astonishing degree of beauty and perfection." As such (2) It is not the case "that human reason is irredeemably irrational by asking itself questions to which it cannot answer, and at the same time emphatically asserting that only reason can answer them." It follows that (3) There are no "undecidable questions of number theory for the human mind." So (4) "The human mind surpasses all machines."

However, the inference from (1) to (2) seems to be obtained from accidental successes in very limited fields to justify an anticipation of universal success. Besides, both (2) and (3) concern only a specific and delimited part of mind and reason which refer just to mathematical issues.

Gödel understands that his incompleteness theorem by itself does not imply that the human mind surpasses all machines. An additional premise is necessary. Gödel presents three suggestions to that end: (a) It is sufficient to accept his 'optimistic realism' (b) By appealing "to the fact that *the mind, and the way it's used, is not static, but finds itself in constant development,*" Gödel suggests that "there is no reason to claim that" the number of mental states "cannot converge to infinity in the course of its development." (c) He believes that there is a mind separate from matter, and that such will be demonstrated "scientifically (maybe by the fact that there are insufficient nerve cells to account for all the observable mental operations)."

There is a known ambiguity between the notion of *mechanism* confined to the mechanizable (in the precise sense of computable or recursive) and the notion of

materialist *mechanism*. Gödel enounces two propositions: (i) The brain operates basically like a digital computer. (ii) The laws of physics, in their observable consequences, have a finite limit of precision. He is of the opinion that (i) is very likely, and that (ii) is practically certain. Perhaps the interpretation Gödel assigns to (ii) is what makes it compatible with the existence of non-mechanical physical laws, and in the same breath he links it to (i) in the sense that, as much as we can observe of the brain's behaviour, it functions like a digital computer.

Is mathematical insight algorithmic?

Roger Penrose (1994) claims that it is *not*, and supports much of his argument, as J. R. Lucas before him (revisited in 1996), on Gödel's incompleteness theorem: It is *insight* that allows us to *see* that a Gödel assertion, undecidable in a given formal system, is accordingly true. How could this intuition be the result of an algorithm? Penrose insists that his argument would have been "certainly considered by Gödel himself in the 1930s and was never properly refuted since then …"

However, in his Gibbs lecture delivered to the *American Mathematical Society* in 1951, Gödel openly contradicts Penrose:

> "On the other hand, on the basis of what has been proven so far, it remains possible that a theorem proving machine, indeed equivalent to mathematical insight, can exist (and even be empirically discovered), although that cannot be *proven*, nor even proven that it only obtains correct theorems of the finitary number theory."

In reality, during the 1930s, Gödel was especially careful in avoiding controversial statements, limiting himself to what could be proven. However, his Gibbs lecture was a veritable surprise. Gödel insistently argued that his theorem had important philosophical implications. In spite of that, and as the above citation makes it clear, he never stated that mathematical insight could be shown to be non-algorithmic.

It is likely that Gödel would agree with Penrose's judgment that mathematical insight could not be the product of an algorithm. In fact, Gödel apparently believed that the human mind could not even be the product of natural evolution. However, Gödel never claimed that such conclusions were consequence of his famous theorem.

Due to the current prevailing trend to restrict discussion about the limits of rationality, in contraposition to insight, to the well-defined and surprising advances in computer science technology and programming, the perspective of considering reason in terms of mechanical capabilities has received much attention in the last decades. Such is recognised as being core to the study of AI, which is clearly relevant to Gödel's wish of separating mind and machine.

In this stance, AI would be primarily interested in what is feasible from the viewpoint of computability, whose formal concern involves only a very limited part of mathematics and logic. However, the study of the limitations of AI cannot be reduced to this restriction of its scope. In this regard, it is essential to distinguish between *algorithms for problem-solving*, and *algorithms simpliciter*, as sets of rules to follow in a systematic and automatic way, which are eventually self-modifiable, and *without* necessarily having a specific and well-defined problem to solve.

Logic consciousness

If one asks “How can we introduce the unconscious in computers?” some colleagues will answer that computers are totally unconscious. In truth, what we don’t know is how to introduce consciousness in algorithms, because we use computers as unconscious appendices to our own consciousness. The question is as such premature, seeing that we can only refer to the human conception of unconscious after we introduce consciousness into the machine. Indeed, we understand much better the computational unconscious than our own unconscious.

These fertile questions point to the complexity of our thought processes, including those of creativity and intuition, that in great measure we do not understand, and pose a much richer challenge to AI, which can help us by providing an indispensable symbiotic mirror.

The translation into a computational construction, of some functional model, as alluded above, of an introspective and thus self-referent consciousness, would be permitted using whatever methodologies and programming paradigms currently at our disposal. In the light of this realization, one might be inclined to ask why the use a logic paradigm, via logic programming, which is precisely the one we prefer (e.g. Lopes and Pereira (2006), plus ongoing work). There are several arguments which can be raised against its use. Hence we will try to reproduce in this section the most relevant, rebut them, and present our own in its favour.

The first argument to be raised in these discussions is that regular human reasoning does not use logic, there existing complex, non-symbolic processes in the brain that supposedly cannot be emulated by symbolic processing. Following this line of thought, many models have been produced based on artificial neural networks, on emergent properties of purely reactive systems, and many others, in an attempt to escape the tyranny of GOFAI (‘Good Old Fashioned AI’). There is a catch, however, to these models: Their implementation by its proponents ends up, with no particular qualms, being on a computer, which cannot help but use symbolic processing to simulate these other paradigms.

The relationship of this argument to logic is ensured by the philosophical perspective of functionalism: logic itself can be implemented on top of a symbol processing system, independently of the particular physical substrate supporting it. Once a process is described in logic, we can use its description to synthesize an artefact with those very same properties. So long it is a computational model, any attempt to escape logic will not prove itself to be inherently more powerful.

On the other hand, there is an obvious human capacity for understanding logical reasoning, a capacity developed during the course of brain evolution. Its most powerful expression today is science itself, and the knowledge amassed from numerous disciplines, each of which with their own logic nuances dedicated to reasoning in their domain. From nation state laws to quantum physics, logic, in its general sense, has become the pillar on which human knowledge is built and improved, the ultimate reward for our mastery of language.

Humans can use language without learning grammar. However, if we are to understand linguistics, knowing the logic of grammar, syntax and semantics is vital. Humans do use grammar without any explicit knowledge of it, but that does not mean it cannot be described. Similarly, when talking about the movement of electrons we surely do not mean that a particular electron knows the laws it follows, but we are certainly using symbolic language to describe the process, and it is even possible to use the description to implement a model and a simulation which exhibits precisely the same behaviour. Similarly, even if human consciousness does not operate directly on logic, that does not mean we won't be forced to use logic to provide a rigorous *description* of that process. And, if we employ logic programming for the purpose, such a description can function as an executable specification.

Once obtained a sufficiently rigorous description of the system of consciousness, we are supposedly in possession of all *our* current (temporary) knowledge of that system, reduced to connections between minimal black boxes, inside which we know not yet how to find other essential mechanisms. Presently, no one has managed to adequately divide the black box of our consciousness about consciousness in the brain, but perhaps we can provide for it a sufficiently rigorous description so that we can model a functional system its equivalent. If a division of that epistemic cerebral black box into a diversity of others is achieved later on, we are sure to be able, in this perspective, to describe new computational models equivalent to the inherent functional model.

In its struggle for that rigorous description, the field of Artificial Intelligence has made viable the proposition of turning logic into a programming language (Pereira 2002). Logic can presently be used as a specification language which is not only executable, but on top of which we can demonstrate properties and make proofs of correction that validate the very descriptions we produce with it. Facing up to the challenge, AI developed logic beyond the confines of monotonic cumulativity, far removed from the artificial paradises of the well-defined, and well into the real world purview of incomplete, contradictory, arguable, reviseable, distributed and updatable, demanding, among other, the study and development of non-monotonic logics, and their computer implementation.

Over the years, enormous amount of work has been carried out on individual topics, such as logic programming language semantics, belief revision, preferences, evolving programs with updates, and many other issues that are crucial for a computational architecture of the mind. We are in the presence of a state-of-the-art from whence we can begin addressing the more general issues with the tools already at hand, unifying such efforts into powerful implementations exhibiting promising new computational properties. Computational logic has shown itself capable to evolve to meet the demands of the difficult descriptions it is trying to address.

The use of the logic paradigm also allows us to present the discussion of our system at a sufficiently high level of abstraction and generality to allow for productive interdisciplinary discussions both about its specification and its derived properties.

As previously mentioned, the language of logic is universally used both by the natural sciences and the humanities, and more generally is at the core of any source of human derived common knowledge, so that it provides us with a common ground on which to reason about our theories. Since the field of cognitive science is essentially a joint effort

on the part of several distinct knowledge fields, we believe such language and vocabulary unification efforts are not just useful, but mandatory.

Consulted oeuvres and references

J. J. Alferes, A. Brogi, J. A. Leite, L. M. Pereira (2002). Evolving Logic Programs. in: S. Flesca et al. (eds.), Procs. of the 8th European Conf. on Logics in Artificial Intelligence (JELIA'02), 50-61, Springer, LNAI 2424, 2002.

J. L. Casti, W. DePauli (2000). *Gödel – A life of Logic*. Basic Books 2000.

M. Davis (1990). Is mathematical insight algorithmic? *Behavioral and Brain Sciences*, 13 (4), 659- 60, 1990.

M. Davis (1993). How subtle is Gödel's theorem. More on Roger Penrose. *Behavioral and Brain Sciences,* 16, 611-612, 1993.

M. Davis (2000). *The Universal Computer: The Road from Leibniz to Turing*. W.W. Norton & Co. 2000.

D. C. Dennett (2005). *Sweet Dreams – philosophical obstacles to a science of consciousness.* The MIT Press 2005.

C. Glymour (1992). *Thinking things through*. The MIT Press 1992.

A. Hodges (1983). *Alan Turing – the enigma*. Simon and Schuster 1983.

G. LaForte, P. J. Hayes, K M. Ford (1998). Why Gödel's theorem cannot refute computationalism. *Artificial Intelligence* 104, 265-286, 1998.

G. Lopes, L. M. Pereira (2006). Prospective Programming with ACORDA. in: Empirically Successful Computerized Reasoning (ESCoR'06) workshop, at The 3rd International Joint Conference on Automated Reasoning (IJCAR'06), Seattle, USA, 2006.

J. R. Lucas (1996). Minds, Machines, and Gödel: A Retrospect. in: P. Millican and A. Clark (eds.), *Machines and Thought (vol. 1),* 103-124, Oxford University Press 1996.

D. McDermott (2001). *Mind and Mechanism*. The MIT Press 2001.

M. L. Minsky (1967). *Computation: Finite and Infinite Machines*. Prentice-Hall 1967.

E. Nagel, J. R. Newman (2001). *Gödel's Proof*. New York University Press 2001.

E. Nagel, J. R. Newman, K. Gödel, J.-Y. Girard (1989). *Le Théorème de Gödel*. Éditions du Seuil 1989.

R. Penrose (1994). *Shadows of the Mind: a search for the missing science of consciousness.* Oxford University Press 1994.

L. M. Pereira (2002). Philosophical Incidence of Logical Programming.

in: *Handbook of the Logic of Argument and Inference,* D. Gabbay et al. (eds.), 425-448, Studies in Logic and Practical Reasoning series, vol.1, Elsevier Science 2002.

L. M. Pereira, A. M. Pinto (2005). Revised Stable Models – a semantics for logic programs. in: C. Bento et al. (eds.), Progress in AI, Procs. 12th Portuguese Intl. Conf. on Artificial Intelligence (EPIA'05), 29-42, Springer, LNAI 3808, 2005.

A. Turing, J.-Y. Girard (1995). *La Machine de Turing*. Éditions du Seuil 1995.

H. Wang (1973). *From Mathematics to Philosophy*. Routledge 1973.

H. Wang (1987). *Reflections on Kurt Gödel*. The MIT Press 1987.

J. C. Webb (1980). *Mechanism, Mentalism and Meta-mathematics*. Reidel 1980.