Turing is Among Us

Luís Moniz Pereira

Centro de Inteligência Artificial – CENTRIA
Departamento de Informática, Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa, 2829-516 Caparica, Portugal

Phone: (+351) 21 2948536 Email: lmp@fct.unl.pt

17 April 2012

Abstract

Turing's present-day and all-time relevance arises from the timelessness of the issues he tackled, and the innovative light he shed upon them.

Turing first defined the algorithmic limits of computability, when determined via *effective* mechanism, and showed the generality of his definition by proving its equivalence to other general, but less algorithmic, non-mechanical, more abstract formulations of computability.

In truth, his originality much impressed Gödel, for the simplicity of the mechanism invoked—what we nowadays call a Turing Machine (or program)—and for the proof of existence of a Universal Turing Machine (what we call digital computer)—which can demonstrably mimic any other Turing Machine, that is, execute any program. Indeed, Turing Machines simply rely on having a finite-state automaton (like a vending machine), and an unbound paper tape made of discrete squares (like a paper roll), with at most one rewritable symbol on each square.

Turing also first implicitly introduced the perspective of 'functionalism'—though he did not use the word, only Putnam later would inspired on Turing's work—by showing that what counts is the realizability of functions, independently of the hardware which embodies them.

And that realizability is afforded by the very simplicity of his mechanism of election, what he then called A-machines (but now bear his name), which rely solely on the manipulation symbols—as discrete as the fingers of one hand—wherein both data and instructions are represented with symbols, both being subject to manipulation. The twain, data as well as instructions, are stored in memory, where instructions double as data and as rules for acting—the stored program idea.

No one to this day has invented a computational mechanical process with such general properties, which cannot be theoretically approximated with arbitrary precision by some Turing Machine.

In these days of discrete-time quantization, computational biological processes, and proof of ever expanding universe—the automata and the tape—the Turing Machine reigns supreme. Moreover, universal functionalism—another Turing essence—is what enables the inevitable bringing together of the ghosts in the several embodied machines (silicon-based, biological, extra-terrestrial or otherwise) to promote their symbiotic epistemic co-evolution, since they partake of the same theoretic functionalism.

Turing is truly and forever among us.

Keywords: Alan Turing, Turing Machines, Computability, Functionalism, Learning.

Alan Turing and Computation

Alan Turing dared to ask whether a machine could think. His contributions to understanding and answering this and other questions defy conventional classification. At the start of this twenty-first century, the 1936 concept of the Turing machine appears not only in mathematics and computer science, but in cognitive science and theoretical biology. His paper 'Computing machinery and intelligence' (Turing 1950), describing the so-called Turing test, is a cornerstone of the theory of Artificial Intelligence (Hodges 1997). In between, Turing played a vital role in the outcome of the Second World War, and produced single-handedly a far-sighted plan for the construction and use of an electronic computer (Hodges 1983). He thought and lived a generation ahead of his time, and yet the features of his thought, which burst the boundaries of the 1940s, are very much alive among us today.

Gödel's work left outstanding Hilbert's question of *decidability,* the ***Entscheidungsproblem*****,** namely the question of whether there exists a definite method which, at least in principle, can be applied to a given proposition to decide whether that proposition is provable. This question had survived Gödel's analysis because its settlement required a precise and convincing definition of *method.* This Turing achieved, working by himself for a year until April 1936; his idea, now known as the **Turing machine,** was to be published at the very end of 1936, in the paper *On Computable Numbers, with an Application to the Entscheidungsproblem* (Turing 1936). It is characteristic of Turing that he refreshed Hilbert's question by casting it in terms not of proofs, but of computing numbers. The reformulation staked a clear claim to have found an idea central to mathematics. As his title said, the *Entscheidungsproblem* was only an application of a new idea, that of computability (Hodges 1997).

What we nowadays call a 'Turing machine' is a finite state automaton supplied with a 'tape' (the analogue of paper) running through it, and divided into sections (called 'squares') each capable of bearing a 'symbol'. At any moment there is just one square [...] which is 'in the machine'. We may call this square the 'scanned square'. The symbol on the scanned square may be called the 'scanned symbol'. The 'scanned symbol' is the only one of which the machine is, so to speak, 'directly aware' [...]. Turing then specifies precisely the repertoire of actions available to the imagined machine. An action is totally determined by the 'configuration' the machine is in, and the symbol it is currently scanning. It is this complete determination that makes it 'a machine'. The action is limited to the following: at each step it (1) either erases the symbol or prints a specified symbol, (2) moves one square either to left or right (3) changes to a new configuration. Slightly different versions of Turing's idea are given in different text books, and the precise technical form he originally gave is not important; the essence is that the action is completely given by what Turing calls a 'table of behaviour' for the machine, dictating what it will do for every configuration and every symbol scanned: this determination is what makes it 'mechanical.' Each 'table of behaviour' is a different Turing machine. The actions are highly restricted in form, but Turing's thesis is that they form a set of atomic elements out of which all mathematical operations can be composed. In fact, in a style very unusual for a mathematical paper, argument is given in very general terms,

justifying the Turing machine actions as sufficient to encompass the most general possible method of computing (Casti & DePauli 2000, Leavitt 2011, Minsky 1967, Pereira 1970):

"I shall also suppose that the number of symbols which may be printed is finite. If we were to allow an infinity of symbols, then there would be symbols differing to an arbitrarily small extent. The effect of this restriction of the number of symbols is not very serious. It is always possible to use sequences of symbols in the place of single symbols" (Turing 1936).

Turing thus claims that a finite repertoire of symbols actually allows a countable infinity of symbols, but not an infinity of immediately recognisable symbols. Note also that the tape has to be of unlimited length, although at any time the number of symbols on it is finite. The computable numbers are then defined as those infinite decimals which can be printed by a Turing machine, starting with a blank tape.

Turing thus approached the question of computable functions in the opposite direction from the usual, that is, from the point of view of the numbers produced as a result, not from the point of view of which functions can be constructed from a set of primitive ones. Turing began with the informal idea of a computer—which in 1935 meant not a calculating machine but a human being equipped with pencil, paper, and time. He then substituted unambiguous components in, until nothing but a formal definition of 'computable' remained. To this end Turing introduced two fundamental assumptions: discreteness of time and discreteness of state of mind. A Turing machine thus embodies the relationship between an unbound array of symbols in space, and a sequence of events in time, regulated by a finite number of mental states. The title 'On Computable Numbers' (rather than 'On Computable Functions') signalled a fundamental shift. Before Turing things were done to numbers. After Turing, numbers began doing things. By showing that a machine could be encoded as a number, and a number decoded as a machine, 'On Computable Numbers' led to numbers (now called "software", that were "computable", in a way that was entirely new (Dyson 2012). He sketched a proof that π is a computable number, along with every real number defined by the ordinary methods of equations and limits in mathematical work.

Armed with his new definition of computable, it is then easy to show that uncomputable numbers exist. The crucial point is that the table of behaviour of any Turing machine is finite. Hence, all the possible tables of behaviour can be listed in an alphabetical order: this shows that the computable numbers are countable. Since the real numbers are uncountable, it follows that almost all of them are uncomputable (Hodges 1983).

Inspection shows that the problem comes in identifying those Turing machines which fail to produce infinitely many digits. This is not a computable operation: that is, there is no Turing machine which can inspect the table of any other machine and decide whether it will produce infinitely many digits or not. This can be seen more directly: if there were such a machine, it could be applied to itself, and this idea can be used to get a contradiction. Nowadays this is known as

the fact that the *halting problem* cannot be decided by a Turing machine. From this discovery of a problem that cannot be decided by a machine, it is not a difficult step to employ the formal calculus of mathematical logic, and to answer Hilbert's *Entscheidungsproblem* in the negative (*ibidem)*.

Alonzo Church, the pre-eminent American logician at Princeton, announced the same conclusion or thesis regarding the *Entscheidungsproblem.* Church's thesis was the claim that effective calculability could be identified with the operations of Church's very elegant and surprising formalism, that of the lambda-calculus (from which Lisp arose). Turing wrote an appendix relating his result to Church's (Turing 1936, 1995), under whom he studied in Princeton for a spell, from 1936 onwards, already after having produced his paper's draft. In March 1937, Alonzo Church reviewed 'On Computable Numbers' for the *Journal of Symbolic* Logic, and coined the term *Turing machine*. "Computability by a Turing machine," wrote Church, "has the advantage of making the identification with the effectiveness in the ordinary (not explicitly defined) [intuitive] sense evident immediately."

Church's thesis is now sometimes called the Church-Turing thesis, but the Turing thesis is different, bringing the physical world into the picture with a claim of what can be done. 'On Computable Numbers' not only solved a major outstanding question posed by Hilbert, and opened the new mathematical field of computability, and offered a new analysis of mental activity, but also had a practical implication: it laid out the principle of the computer through the concept of the Universal Turing machine (M. Davis 1958). Indeed, the several and diverse attempts to formally and rigorously characterize the *a priori* 'intuitive' notion of effective computable function, by quite different approaches, ended up all proven equivalent, thereby sustaining the consensus that intuition has been definitely captured. Among these approaches we have: Church's λ-definability, Turing's computability, Post's canonical systems, Smullyan's formal elementary systems, Herbränd-Gödel-Kleene's general recursiveness (Davis 1958, Minsky 1967, Pereira 1970).

The idea of the Universal machine is easily indicated. Once the specification of any Turing machine is given as a table of behaviour, tracing the operation of that machine becomes a mechanical matter of looking up entries in the table, and of mimicking them. Because it is mechanical, a Turing machine can do it: that is, a single Turing machine may be designed to have the property that, when supplied with the table of behaviour of another Turing machine, it will do whatever that other Turing machine would have done. Turing called such a machine the Universal machine (Dyson 2012, Hodges 1983, Leavitt 2011, Minsky 1967, Pereira 1970).

The Universal machine affords Turing the claim to have invented the principle of the computer—and not just in an abstract way. One cannot study Turing machines without seeing them as computer programs that are executable, stored in symbols, along with the data on the memory tape—the 'modifiable stored program'. The universal machine being the computer on which any programs may run. Symbols describe both a program's instructions and the actions they trigger on the universal machine. Moreover, programs themselves can be

manipulated on the tape, even self-modifiable—a possibility that AI only recently began to explore (Alferes et al. 2002, Pereira & Lopes 2009, Pereira & Han 2009).

The Turing machine construction had shown how to make all formal proofs 'mechanical'; and in the 1936 paper such mechanical operations were to be taken as trivial, putting instead under the microscope the non-mechanical steps which remained. In this period that he abandoned the idea that moments of intuition corresponded to uncomputable operations. Instead, he decided, the scope of the computable encompassed far more than could be captured by explicit instruction notes, and quite enough to include all that human brains did, however creative or original.

Machines of sufficient complexity would have the capacity for evolving into behaviour that had never been explicitly programmed. Turing's claim is that the only features of the brain relevant to thinking or intelligence are those which fall within the discrete-state-machine level of description (Hodges 1997). The particular physical embodiment is irrelevant—what would later become known as 'machine functionalism'. (In Logic, the Herbränd base and the Herbränd universe likewise afford a similar representational abstraction.)

Alan Mathison Turing, born 23 June 1912

The post-war Turing claims that Turing machines can mimic the effect of *any* activity of the mind, not only a mind engaged upon a 'definite method'. Turing is

clear that discrete state machines include machines with learning or self-organizing ability, and makes a point of the fact these still fall within the scope of the computable. Turing draws attention to the apparent conflict with the definition of Turing machines having fixed tables of behaviour, but sketches a proof that self-modifying machines are still in fact defined by an unchanged instruction set. Turing advocates two different approaches—in modern parlance top-down and bottom-up—which in fact derive from his 1936 descriptions of the machine model. Explicit instruction notes become explicit programming; implicit states of mind become the states of machines attained by learning and self-organizing experience (Hodges 1983, McDermott 2001).

Gödel, Computability, and Turing

The third of Hilbert's questions to the international congress of mathematics in Bologna in 1928, that of *decidability*, became formulated in terms of *provability*, instead of *truth*, as 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 *mechanical* 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? (Pereira 2007, Wang 1973, Webb 1980).

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 mechanical 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 mechanical 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". Over the years, Gödel regularly credited Turing's 1936 article as the definitive work that captures the intuitive concept of computability, and the only author to present persuasive arguments about the adequacy of the precise concept he himself defined (Wang 1987).

Regarding the concept of mechanical 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) (*ibidem*).

To this end, in 1934, Gödel introduces the concept of general recursive functions, which was later shown to capture the intuitive concept of mechanical 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 (Casti & DePauli 2000, Wang 1973, Webb 1980).

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. "With this concept one has for the first time succeeded in giving an absolute definition... not depending on the formalism chosen," he admitted in 1946 (Dyson 2012).

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" (Wang 1987).

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 (Hodges 1983).

In his Ph.D. thesis, completed in May of 1938 and published as "Systems of Logic Based on Ordinals" in 1939, Turing attempted to find a way out from the power of Gödel's incompleteness theorem. The fundamental idea was that of adding to the initial system successive axioms, so that it would incrementally become more complete, making non-deterministic steps once in a while by consulting "a kind of oracle, as it were, that cannot be a machine." 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 such a type will be produced to be taken 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. Turing showed, however, that undecidable statements, resistant to the assistance of an external oracle, could still be constructed, and the *Entscheidungsproblem* would remain unsolved (Webb 1980).

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* mechanical. 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 (Hodges 1983).

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 (*ibidem*).

Mens ex-Machina

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 (Pereira 2007, Wang 1987).

There is a known ambiguity between the notion of *mechanism* confined to the mechanical (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, like a Universal Turing machine (*ibidem*).

Is Mathematical Insight Algorithmic?

Roger Penrose (1994) claims that it is *not*, and supports much of his argument, as J. R. Lucas before him (Lucas 1969), 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 …" Davis (1990).

However, in his Gibbs lecture, delivered to the *American Mathematical Society* in 1951, Gödel openly contradicts Penrose (Casti & DePauli 2000):

> "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. Indeed, 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 (Wang 1987).

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, and rooted in Turing's achievement, 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, like Turing ventured, and *without* necessarily having a specific and well-defined problem to solve (Davis 1990, Pereira 2007).

Prolegomena to Artificial Neurology

Potential bridging points between neurology and computer science deserve being explored, with special emphasis on the hardware/software distinction. First, for some definitions.

That which is essentially the result of an intention is artificial, even if itself has intentionality – its intentions will be artificial. The intention behind which there is no other is natural. Nevertheless, there's nothing more artificial than the definition of 'natural' (Pereira 1979, 1988).

For example, if the universe was intended by an all-powerful being who thereby realized its intentions, then all of it is artificial, including of course our own intentions. Another example: a computer built by a human being, and which manifests intentions, is wholly artificial, regardless of whether the human being, in turn, is artificial.

Granted, the usage of words changes. I can even foresee that the word 'natural', by virtue of the development of Artificial Intelligence, may come to be identically applied to many a computer, the originally artificial intentions of which will no longer be foreseeable by its human constructors. Then the creature forfeits independence from its creator.

This section refers to an artificial world where there exist brains. I shall always thus omit the word 'artificial', and hence will write 'neuron' rather than 'artificial neuron', 'intention' rather than 'artificial intention', etc.

The artificial world alluded to, wherein there exist brains, was constructed by anonymous nondescript intentional beings, as an experimental laboratory for possibly confirming their conceptions about their own world. In particular, the brains therein are used as epistemological devices where the constructors of the world test, *in vitro*, some of their conceptions about their own cognitive processes, metaphysics, etc.

Such brains function according to principles previously tested by the constructors of the world on their computers, although the physiological and physical substrate of ones (the computers) and the others (the brains) differ substantially, the twain differing yet again from the material substrate supporting the mental activity of the world constructors.

This state of affairs does not prevent ones be taken as models of the others. To the contrary, it is on that very principle of software independence of hardware, widely confirmed by their Computer Science, that the constructors of the world base their constructions. Consequently, when modelling their cognitive processes, they explore the potentialities of the model to improve, through a cognitive feedback loop, their own mental abilities.

Recourse to the nomenclature, paradigms, and techniques of computer science for brain modelling was not new to them. In fact, computer science had been, in the process, enriched with the nomenclature, paradigms, and techniques of the brain sciences.

Historically, that cross-fertilization began when it was acknowledged that only with the help of an instrument with the accumulated and organized complexity of a computing system (computer, peripherals, and programs) was it possible to deal in a rigorous fashion with the complexity of cerebral processes and structures of the constructors of the world and, subsequently, with those of the artificial brains they wished to build.

In particular, computer implemented models become well-defined, eminently observable in their formulation and their dynamics, and can be transformed incrementally in an expedite way. On the other hand, the observable dynamics of the models liberates neurology from its excessive emphasis on the pathology of lesions, and allows it to adopt a methodology more in line with the study of normal brain functioning. But pathology itself gains a new impetus with the possibility of simulating, in the model, a platter of specific and provoked lesions.

The Software/Hardware Distinction

Let us recall, next, the admittedly most important principle originator of the introduction of the computer as a laboratory for brain models (Pereira 1979, Pereira 1988).

The principle of the distinction between software and hardware—of function and form, after all, in its simplest version, which in some way is present in any machine—appears finally clear-cut with the advent of the digital computer and its conceptual precursor, the Universal Turing machine.

The diversity of technologies employed to achieve the same function, ever since the first computers, indeed confirms it. One same program is executable in physically different machines, precisely because at the level of discourse of the program the details of its execution, below an ascertainable level of analysis, are irrelevant, just as long the an identical level of discourse result is produced. In a crude analogy, one may say the ink colour and the handwriting are irrelevant to the message being conveyed.

This distinction, let it be said, is susceptible of levels. That which is hardware is not necessarily things physical, but rather that which, at a certain level of analysis, is considered fixed, given, and whose analysis or non analysability may irrelevant for some purpose—e.g. RISC instructions, or glia cells' DNA.

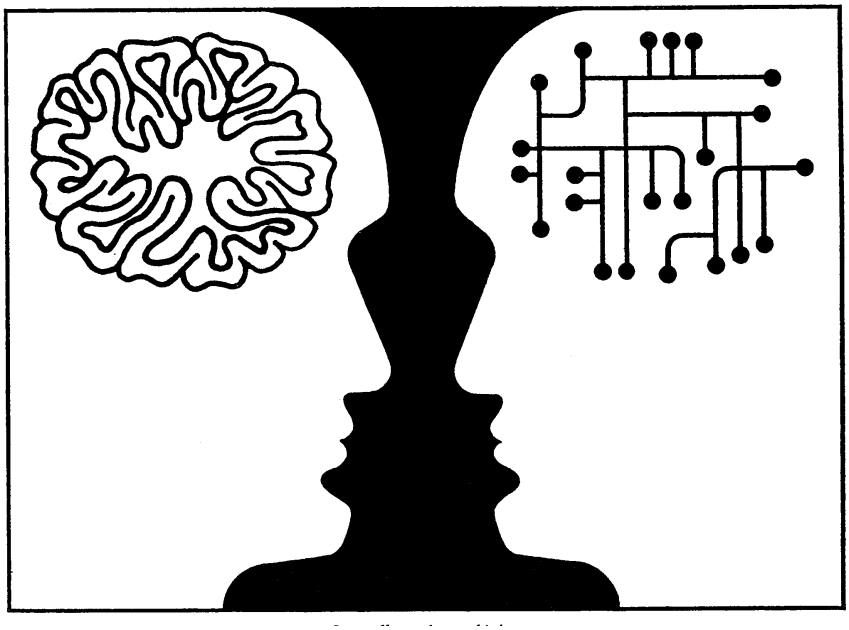
Historically, in the first computers, that level coincided with the level of the physical parts of the machine—hence the confusion. Subsequently, that level moved in opposite directions, and its relativity became clear. On the one hand, the concept of abstract machine was introduced, that is, a given and non analysable fixed collection of instructions, mathematically defined, capable of supporting a set of software functions, independently of the physical processes and details that enact the implementation of the abstract machine on a physical

one. On the other hand, the fixed physical components of one generation of computers have given place, in the next generation of computers, to partially programmable components, whose functions are software determinable (microprogramming). An old crude analogy would in this case be an IBM changeable ball typewriter. More to the point, some previously software defined functions, such as floating point arithmetic, became fully hardware available.

One main consequence, for artificial neurology, of this ever-present principle—whose precursors were the first axiomatic schools of thought—consists in the better focusing on the level of neurological analysis more appropriate for answering its questions.

Another main consequence has been, of course, the rising popularity, among the neurologists, of the computer as an instrument of simulation, given the advantage of being able to choose the level of abstraction of the simulation, including at the level of the neuron. In this task they helped themselves of the theory of black boxes of abstraction, successively and successfully developed by the computer scientists.

Many neurologists, in fact, had no difficulty in adjusting to the idea of neuron simulation because they recognized that their own conception of the neuron was already a rather abstract model of it, in contradistinction with the case of computer scientists, who can intimately know the computer's 'circuits', given that these are built according to specifications.



The Epistemic Mirror

The software/hardware distinction is rich in consequences for artificial neurology. Namely it explains why the correspondence between function and its physical support is not compulsory. The physical hardware is not specific of any high level software function. Instead, it enables the execution of a variety of functions, in a distributed and non-localized way, exception being made for the

hardware specific to the interface with peripheral organs, and to external information coding/decoding, as is the case of the nervous system (Churchland 2002).

As the cerebral processes gain in abstraction and level of integration of several sensory sources, the neurons which support them become less specific and independent of the origin of the information. How else would integration be possible if it were not so?

It is a fact that beyond the interface with the exterior there exist specific hardware that is also specialized. But such specificity could conceivably be realized in other ways, *pace* organic chemistry, so that no specific software actually requires a specific hardware.

Another consequence of the hardware/software distinction concerns the notion of appropriate explanation level. A program can be understood, in its function or dysfunction, in terms of itself, of its own level of discourse. Of course, its dysfunction can originate in a dysfunction of the underlying hardware, but in that case it manifests itself in a bizarre behaviour, not comprehensible at the program's level of discourse, and not specific to that program.

Complementarily, its function can be described resorting to the hardware level, but such description does not constitute an appropriate level of explanation because, in being too detailed, it cannot be generalised. The following analogy, adapted from Putnam, is an example of what I mean.

Imagine a rigid panel, with a circular hole of 10 cm in diameter, and a square hole of 10 cm side. One wants to explain why a rigid cube, of 9 cm side, goes through one of the holes but not the other. The adequate level of explanation resorts to the geometrical concepts and principles involved. A possible but in appropriate level of an explanation would consider the quantum-physical properties of the materials present, say glass for the panel and aluminium for the cube, and explain the impossibility of passage of the cube through one the circles in terms of mechanical resistance, for any approaching trajectory.

Such an explanation, because overly specific, it is generalised only with difficulty to other constituent materials, say iron and granite. There is a level of analysis below which the explanation loses in generality and becomes unnecessary, as long as all the relevant properties are guaranteed to be fixed at a that level—in the example given, the form invariance of the elements in presence, relative to any trajectory followed and/or their physical composition.

Finally, in a computer, the software prevails, in general, over the hardware. Though the hardware supports and causes the execution of the software, the initiative belongs, more times than not, to the software. It is the software that chooses and provokes the coming into activity of the appropriate hardware at each step—Turing's stored program.

Such activity consists in consulting the instructions stored in memory, and in executing the software instructions in the hardware, with the result that instruction-selected hardware is provoked into activity, closing the circle. In this

way, the teleology of the software is kept in charge, notwithstanding the underlying causality of the physical hardware.

Once a Turing machine reads the first scanned symbol on its tape, then the instructions therein assume control. Similarly, once a computer is bootstrapped, it starts executing the instructions in its memory. Outside interrupts and updates by oracles to its memory may occur, only for the same pattern to be followed relentlessly. After all, we want the computer to do what its software tells it to do.

Logic and Consciousness

If one asks “How can we introduce the unconscious in computers?” some 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 as such is 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 (Pereira 2007).

In the early 1980's, William Reinhardt was interested in how much a Turing machine could know about itself. He conjectured that in epistemic arithmetic—Peano's enriched with a knowledge operator—a Turing machine can prove, with mathematical certainty, the sentence "I know I am a Turing machine." Timothy Carlson gave the first proof of Reinhardt's conjecture in the mid 1990's (Buechner 2007).

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, not just epistemological but a symbiotic mirror too.

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. 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’), rooted in Turing's symbol computationalism—itself arisen to answer in the negative Hilbert's *Entscheidungsproblem* on the decidability of logic augmented with arithmetic. 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. And neural networks can implement a Universal Turing machine, if not logic directly.

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. Likewise, even if human consciousness does not *operate* directly on logic, that does not mean we won't be forced to use logic, amongst ourselves, 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 too.

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.

Functionalism

The thesis of multiple realizability says that a mental state can be 'realized' or 'implemented' by different physical states. Beings with different physical

constitutions can thus be in the same mental state, and can hence symbiotically cooperate epistemic-wise.

According to classical functionalism, multiple realizability implies that psychology is autonomous: in other words, biological facts about the brain are irrelevant to it (Boden 2008). As one computationalist put it: "whether the physical descriptions of the events subsumed by the psychological generalizations have anything in common is, in an obvious sense, entirely irrelevant to the truth of the generalizations, or to their interestingness, or to their degree of confirmation, or, indeed, to any of their epistemological important properties" (Fodor 1974). This doctrine is still used as an argument to counter the objection that metal-and-silicon computers are (physically) very different from neuroprotein, and also as a way of avoiding neuroscientific questions to which, as yet, answers cannot be given.

The secret of behaviour revealed in structure

The information-processing systems defined by connectionism are broadly inspired by the brain. However, most existing systems are in fact hugely different from the brain. In general, the component units are computationally far too simple in comparison with real neurons. Moreover, the mathematics that defines the learning rules is usually highly unrealistic. The popular method of back propagation, for example, depends on units' being able to transmit information

in two directions—which real neurons cannot do. The theoretical "autonomy" of psychology/computation remains, in the sense that one may—and sometimes, one must—consider what computer scientists would term the virtual machine of the mind-brain, without worrying about the details of its biological implementation (Boden 2008).

The broad position of "functionalism" can be articulated in many different varieties. The first formulation of a functionalist theory of mind was put forth by Hilary Putnam (Putnam 1960). This formulation, which is now called machine-state functionalism, or just machine functionalism, was inspired by the analogies which Putnam and others noted between the mind and the theoretical "machines" or computers capable of computing any given algorithm which were developed by Alan Turing—and now called Universal Turing machines (Wikipedia 2012). In this light, machines are physical models of abstract processes (Minsky 1967).

In a seminal paper (Turing, 1950), A.M. Turing proposed that the question “Can machines think?” can be replaced by the question “Is it theoretically possible for a finite state digital computer, provided with a large but finite table of instructions, or program, to provide responses to questions that would fool an unknowing interrogator into thinking it is a human being?”

Nowadays, in deference to its author, this question is most often expressed as “Is it theoretically possible for a finite state digital computer (appropriately programmed) to pass the Turing Test?” In arguing that this question is a legitimate replacement for the original (and speculating that its answer is “yes”), Turing identifies thoughts with states of a system defined solely by their roles in producing further internal states and verbal outputs, a view that has much in common with contemporary functionalist theories. Indeed, Turing's work was explicitly invoked by many theorists during the beginning stages of 20th century functionalism, and was the avowed inspiration for a class of theories, the “machine state” theories most firmly associated with Hilary Putnam (1960), that had an important role in the early development of the doctrine (Levin 2010).

Functionalism is fundamentally a broadly metaphysical thesis as opposed to a narrowly ontological one. That is, functionalism is not so much concerned with *what there is* than with what it is that characterizes a certain type of mental state, e.g. pain, as the type of state that it is. Previous attempts to answer the mind-body problem have all tried to resolve it by answering *both* questions: dualism says there are two substances and that mental states are characterized by their immateriality; behaviourism claimed that there was one substance and that mental states were behavioral disposition; physicalism asserted the existence of just one type of substance and characterized the mental states as physical states (as in "pain = C-fiber firings") (Wikipedia 2012).

On this understanding, type-physicalism can be seen as incompatible with functionalism, since it claims that what characterizes mental states (e.g. pain) is that they are physical in nature, while functionalism says that what characterizes pain is its functional/causal role and its relationship with yelling "ouch", etc. However, any weaker sort of physicalism which makes the simple ontological claim that everything that exists is made up of inorganic matter is perfectly

compatible with functionalism. Moreover, most functionalists who are physicalists require that the properties that are quantified over in functional definitions be physical properties. Hence, they *are* physicalists, even though the general thesis of functionalism itself does not commit them to being so (*ibidem*).

In 1984, Putnam elaborated an ingenious argument as backbone of his assault on computationalism, and introduced *Rational Interpretation* (and other related notions). It employs not only demonstrative reasoning, to which the Gödel theorems apply; it also employs non-demonstrative reasoning. Putnam found a way to apply Gödel's theorems to it and, as it is presented in (Putnam 1988), it purports to show that all epistemic methods employed in human inquiry are, if formalized, susceptible to Gödel's theorems. In his rendering, weak formalizable methods of showing that a computer program for human mentality is consistent can be employed (with less than mathematical certainty) by human beings that are not Gödel susceptible, but can't be employed by agents who are Gödel susceptible (Buechner 2007).

Putnam's ingenious argument fails though. Indeed, what mathematicians and philosophers have failed to appreciate is that the Gödel theorems show that no one—whether Gödel susceptible or not—can prove the consistency of Peano arithmetic with mathematical certainty without constructing an infinite proof tree, which puts a limitation on the finitary humans. This is the real reason why anti-functionalists who wish to employ the Gödel theorems are doomed to failure. Unless human beings can construct infinite proof trees, we are limited by Gödel's theorems, even if we are not computing machines to which they directly apply. This simple point has resisted appreciation by the anti-functionalists.

Putnam's argument claims that all methods of inquiry into the world employed by a computing machine are Gödel susceptible. From this it easily follows that the machine cannot know the truth of any of its Gödel sentences in any of the general epistemic modalities, defined by Putnam, and under which it conducts inquiry into the world.

But this limitative result also applies to human beings, even f we are not Gödel susceptible. If a finitary computing machine running a program is unable to determine its correctness in some epistemic modality because it is Gödel susceptible, then no finitary human being can determine its own correctness in that epistemic modality either, even if humans are not Gödel susceptible.

Unless we can construct infinite proof trees, none of our finitary methods of inquiry into the word can show, in their characteristic epistemic modality, the truth of the Gödel sentences arising in the formalization of those methods of inquiry. And if we do not employ a formalizable method of inquiry, no non-formalizable method of inquiry we use can prove the truth of a Gödel sentence of a formalizable method of inquiry in the epistemic modality of that formalization (*ibidem*).

Despite Putnam's rejection of functionalism, it has continued to flourish and been developed into numerous versions by thinkers as diverse as David Marr, Daniel Dennett, Jerry Fodor, and David Lewis (Fodor 1974, Dennett 2005).

Functionalism helped lay the foundations for modern cognitive science, and is the dominant theory of mind in philosophy today.

In the last part of the 20th century, functionalism stood as the dominant theory of mental states. Like behaviorism, functionalism takes mental states out of the realm of the "private" or subjective, and gives them status as entities open to scientific investigation.

But, in contrast to behaviorism, functionalism's characterization of mental states in terms of their roles in the production of behaviour grants them the causal efficacy that common sense takes them to have. And in permitting mental states to be multiply realized, functionalism offers an account of mental states that is compatible with materialism, without limiting the class of minds to creatures with brains like ours (Levin 2010).

Emergence and Functionalism

Biological evolution is characterized by a set of highly braided processes, which produce a kind of extraordinarily complex combinatorial innovation. A generic term frequently used to describe this vast category of spontaneous, and weakly predictable, order generating processes, is 'emergence'.

This term became a kind of signal to refer the paradigms of research sensitive to systemic factors. Complex dynamic systems can spontaneously assume patterns of ordered behaviours which are not previously imaginable from the properties of their composing elements nor from their interaction patterns. There is unpredictability in self-organizing phenomena—preferably called 'evolutionary', as Turing did (Turing 1950)—with considerably diverse and variable levels of complexity.

'Complexity' refers to the study of the emergence of collective properties in systems with many interdependent components. These components can be atoms or macromolecules in a physical or biological context, and people, machines or organizations in a socioeconomic context.

What does emerge? The answer is not something defined physically but rather something like a shape, pattern, or function—as in functionalism. The concept of emergence is applicable to phenomena in which the relational properties predominate over the properties of composing elements in the determination of the ensemble's characteristics. Emergence processes appear due to configurations and topologies, not to properties of elements (Deacon 2003). This functionalism is, almost by definition, anti substance essence, anti vital principle, anti monopoly *qualia*.

Evolutionary Psychology and Logic

Logic, we sustain, provides the overall conceptual cupola that, as a generic module, fluidly articulates together the specific emerged modules identified by evolutionary psychology (Pereira 2012). In that respect, it is mirrored by the

general computability of Turing's Universal machines, which can execute any program, compute any computable function.

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 the description to synthesize an artefact with those very same properties.

However, the canonical definition of objective scientific knowledge avidly sought by the logical positivists is not a philosophical problem nor can it be attained, as they hoped, simply by logical and semantic analysis. It is an empirical question too, that can be answered only by a continuing probe of the possible functionality of the thought process itself and its physical basis.

In some cases, the cognitive tools and instruments of rationality will be found hardware independent. Even then, the appropriateness of their use in specific real circumstances and goals will need to be empirically determined. There is no universal one-size-fits-all epistemological recipe, but agreement can be had on the relative success of any given tool kit.

In any case, partial understanding may also be sought by building intelligent machines, functionalism coming to the rescue when positing that the material substrate is often not of the essence, that it suffices to realize equivalent functionality albeit over different hardware. Moreover, distinct functioning roads to the same behaviour may be had, thereby accruing to our understanding of what general intelligence means, toward their symbiotic entwining, the most recent step in evolutionary epistemology. Functionalism can only make that more adroit.

The most fruitful procedures will almost certainly include the use of Artificial Intelligence, theory and technique, aided in time by the still embryonic field of artificial emotion, to simulate complex mental operations, already foreseen in (Turing 1950). This modelling system will be joined to an already swiftly maturing neurobiology of the brain, including the high-resolution scanning of computational networks active in various forms of thought.

How does natural selection anticipate our future needs? By creating a cognitive machine, called brain, that can create models of the world, and even of itself, plus process hypotheticals, much like a Universal Turing Machine can mimic any other Turing machine, and just like any given computer can in principle run any program. This plasticity provides for its universal versatility (Davis 2000).

It is useful to consider a duality I designate “Turing versus Eve”. The mathematician Alan Turing represents the computer in the essence of its complete flexibility. The Universal Turing Machine is the one which can imitate every computer every program: it is mimetism *par excellence.* Such mimetism makes us think about the meme and our own mental flexibility, so vital in complementing genetic reproduction, due to the different reproduction timings. In the latter, genetic reproduction, the difference spans across generations, and that is not enough when adaptation must be agile. It is from that need that stems

the cerebral mechanism of reproduction—those memes which jump from brain to brain. In genetic reproduction, mitochondria are the fixed genetic structures by excellence, arising from the feminine side, which are replicated without mating of genes. They metaphorically stand for the multitude of specific modules we inherit in virtue of our species' past.

Turing's approach to machine intelligence, in his 1948 NPL report 'Intelligent Machinery' (Turing 1948), was as unencumbered as his approach to computable numbers ten years before. He continued once again from where Gödel had left off. Does the incompleteness of formal systems limit the abilities of computers to duplicate the intelligence and creativity of the human mind? Turing summarized the essence of this convoluted argument in 1947, saying that "in other words then, if a machine is expected to be infallible, it cannot also be intelligent." Instead of trying to build infallible machines, we should be developing fallible machines able to learn from their mistakes (Dyson 2012).

He suggested incorporating a random-number generator to create a "learning machine," granting the computer the ability to take a guess and either reinforce or discard the consequent results. If guesses were applied to modifications in the computer's own instructions, a machine could learn to teach itself. "What we want is a machine that can learn from experience," he wrote. "The possibility of letting the machine alter its own instructions provides the mechanism for this."

Turing drew a parallel between intelligence and "the genetical or evolutionary search by which a combination of genes is looked for, the criterion being survival value. The remarkable success of this search confirms to some extent the idea that intellectual activity consists mainly of different kinds of search." Evolutionary computation would lead to truly intelligent machines. "Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's?" he asked. The path to artificial intelligence, suggested Turing, is to construct a machine with the curiosity of a child, and let intelligence evolve, with the help of outside oracles.

The Internet search engine is a finite-state deterministic machine, except at those junctures where people, individually and collectively, make a non-deterministic choice as to which results are selected as meaningful and given a click. These clicks are then immediately incorporated into the state of the deterministic machine, which grows ever so incrementally more knowledgeable with each click. This is what Turing defined as an oracle machine (Dyson 2012).

Computational Logic and AI

Turing’s work on computability leads to a deep question: "Does computation with discrete symbols give a complete account of our conception of the physical world? In other words, is the world as we see it computable?"

In AI, an agent is any entity, embedded in a real or artificial world, that can observe the changing world and perform actions on the world to maintain itself in a harmonious relationship with the world. Computational Logic (CL), as used in AI, is the agent’s language of thought. Sentences expressed in this language

represent the agent's beliefs about the world as it is and its goals for the way it would like it to be. The agent uses its goals and beliefs to control its behaviour. It can also use CL to guide its public communications with other agents.

The most influential and widely cited argument against logic comes from psychological experiments about reasoning with natural language sentences in conditional form. The most popular interpretation of these experiments is that people do not have a natural general-purpose ability to reason logically—like a Universal Turing machine— but have developed instead, through the mechanisms of Darwinian evolution, only specialised algorithms for solving typical problems that arise in their environment. One of the problems with the experiments is that they fail to appreciate that the natural language form of a conditional is only an approximation to the logical form of its intended meaning. Another problem is that the interpretation of these experiments is based upon an inadequate understanding of the relationship between knowledge and reasoning. In contrast, the CL understanding of human thinking can be expressed loosely as "thinking = specialised knowledge + general-purpose reasoning" (Kowalski 2011), very much in line with Turing's viewpoint on AI.

Accordingly, 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 cumulativeness, far removed from the artificial paradises of the well-defined, and decidedly into the real world purview of the incomplete, contradictory, arguable, revisable, learnable, distributed and updatable, thus demanding consequently, among others, the study and development of non-monotonic logics, and their computer implementation.

Over the years, enormous amount of work has been carried out on these individual topics, such as logic programming language semantics, belief revision, preferences, evolving programs with updates, inductive learning, and many other issues that are crucial for a computational architecture of the mind. We are today 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. CL has shown itself capable to evolve to meet the demands of the difficult descriptions it is trying to address.

The use of the CL 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.

Alan Turing's philosophy might appear the ultimate in reductionism, in its atomizing of mental process, its scorn for the non-material. Yet, seen in the above light, it depends upon a synthesis of vision running against the grain of an intellectual world split into many verbal or mathematical or technical specialisms. His many interests: from the computable, to morphogenesis, to quantum mechanics, to the cognitive mind, show a deep ambition for a an overarching natural philosophy synthesis.

Symbiotic Epistemology

When considering scientific knowledge, if the computer processing of the human genome is what lead us to Bio-informatics then, by analogy, we may state that the "cognome" will be the basis of a future 'Cognotechnology', applicable in any science. This way, the future of AI is connected to the characteristic of it being an epistemological instrument, not only for an autonomous agent, but a symbiotic one which will help humans in performing science itself. And I'm not just talking about data mining, pattern recognition, ontology building, although in those fields we can approach more structured aspects of epistemology. I'm thinking about that which every scientist does, which is to abduce, invent and prophesy theories, put them to the test, create experiments, draw conclusions to support additional observations, discuss those observations and his conjectures with other scientists.

There is an ongoing meta-argumentation about what is good reasoning, what are the conclusions we can draw from a discussion (i.e. a semantics), which is inherent to all scientific activity. The computer will be used more and more as a research aide, not just to automate but also propose experiences and hypotheses and, in the end, by making our own conceptions on epistemology application repeatable and externalized, it will make them more objective too (Pereira 2012).

To understand—and thence to perfect—human intelligence, one must express how it works, and the computer is the experimental apparatus to test our own self-understanding, modelling it in detail to the extent we can. Logic, in the wide sense of logical, is the natural shared vehicle for so doing in a precise scientific way. And the computer, our privileged computational machine par excellence, is no doubt our artificial shared vehicle for objectively proving the worthiness of that understanding. Computational Logic spans both, and symbiotically benefits both.

Veritably, the capacity for cognition is what allows us to anticipate the future, to pre-adapt and imagine scenarios of possible evolutions—of the world and of ourselves as cognitive agents—to make choices, to use preferences about some hypothetical worlds and their futures, and meta-preferences, i.e. preferences on which preferences to employ and how to make them evolve. The activity of prospecting the future is vital and characteristic of our species and its capacity to

understand the real world and ourselves, living in society, where distributed cognition is the normal and regular way to do science.

Prospective consciousness allows us to pre adapt to what will happen. For that, a capacity to simulate, to imagine "what would happen if", i.e. is hypothetical thinking, becomes necessary. Such thinking is indispensable in science; for it gives us the rules to predict and explain what will or can happen, without which technology would not be possible.

The automaton, symbiosis of man and machine

Lately, we've been working towards automating this capacity, by implementing programs which can imagine their futures, making informed choices about them, and then modifying even themselves—just as Turing permitted and predicted would happen—to enact those choices, the inklings of free will. We call it prospective computing (Alferes et al. 2002, Pereira & Lopes 2009, Pereira & Han 2009).

Epistemology will eventually have the ability to be shared, be it with robots, aliens or any other entity who must needs perform cognition to go on existing and program its future. Creating situated computers and robots means carrying out our own cognitive evolution by new means. With the virtue of engendering symbiotic, co-evolving, and self-accelerating loops.

Computerized robots reify our scientific theories, making them objective, repeatable, and a part of a commonly constructed extended reality, built upon multi-disciplinary unified science.

Artificial Intelligence and the Cognitive Sciences, by building such entities, provide a huge and stimulating step towards furthering the unity of science, through the very effort of that construction.

In these days of discrete-time quantization, computational biological processes, and proof of ever expanding universe—the automata and the tape—the Turing Machine reigns supreme. Universal functionalism—Turing's essence—is what enables the inevitable bringing together of the ghosts in the several embodied machines (silicon-based, biological, extra-terrestrial or otherwise) to promote their symbiotic epistemic co-evolution, since they partake of the same theoretic functionalism.

Hence, the computational functionalist general stance, first spurred by Alan Turing, is most helpful. Turing is truly and forever among us!

Acknowledgements

Plates: Alan Mathison Turing (King's College, Cambridge, and The Royal Society). The Epistemic Mirror (from Michael Arbib's "The Metaphorical Brain", Wiley-Interscience 1972). The two Japanese XVIII and XIX century automata, in the order shown (Japan Foundation; Japan Foundation and Yamaguchi Koichi).

Consulted œuvres and references

[My works are available at http://centria.di.fct.unl.pt/~lmp/publications/Biblio.html]

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

M. A. Boden (2008). Information and Cognitive Science. *Philosophy of Information*. P. Adriaans & J. van Bentham (eds.), Amsterdam: North-Holland, Elsevier.

J. Buechner (2007). *Gödel, Putnam, and Functionalism: a New Reading of Representation and Reality*. Cambridge: The MIT Press.

J. L. Casti, W. DePauli (2000). *Gödel – A life of Logic*. Cambridge: Perseus.

P. S. Churchland (2002). *Brain-Wise: Studies in Neurophilosophy*. Cambridge: The MIT Press.

M. Davis (1958). *Computability and Unsolvability*. New York: McGraw-Hill.

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

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

G. Dyson (2012). *Turing's Cathedral: The Origins of the Digital Universe*. London: Allen Lane.

T. W. Deacon (2003). The Hierarchic Logic of Emergence: Untangling the Interdependence of Evolution and Self-Organization. *Evolution and Learning: The Baldwin Effect Reconsidered*. H.W. Weber & D.J. Depew (eds.), Cambridge: The MIT Press.

D. C. Dennett (2005). *Sweet Dreams: Philosophical Obstacles to a Science of Consciousness*. Cambridge: The MIT Press.

G. Dyson (2012). *Turing's Cathedral: The Origins of the Digital Universe*. London: Allen Lane.

J. A. Fodor (1974). Special Sciences, or the Disunity of Science as a Working Hypothesis. *Synthese,* 28: 77-115.

A. Hodges (1983). *Alan Turing – the Enigma*. New York: Simon and Schuster.

A. Hodges (1997). *Alan Turing: one of The Great Philosophers*. London: Phoenix.

R. Kowalski (2011). *Computational Logic and Human Thinking: How to be Artificially Intelligent*. Cambridge: Cambridge University Press.

D. Leavitt (2006). *The Man Who Knew Too Much: Alan Turing and the Invention of the Computer*. London: Phoenix.

J. Levin (2010). Functionalism. *The Stanford Encyclopaedia of Philosophy*, E.N. Zalta (ed.), http://plato.stanford.edu/archives/sum2010/entries/functionalism/.

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

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

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

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

L. M. Pereira (1970). Introdução aos Autómatos Infinitos e Teoria da Computabilidade, *Técnica*, vol. 395, pp. 265-273 & vol. 396, pp. 321-328. Lisboa: Instituto Superior Técnico.

L. M. Pereira (1979). Prolegómeno a uma Neurologia Artificial, *Análise Psicológica*, vol. 11(4), pp. 519-522.

L. M. Pereira (1988). Inteligência Artificial: Mito e Ciência. *Revista Colóquio/Ciências*, vol. 3, pp. 1-13, Lisboa: Fundação Calouste Gulbenkian.

L. M. Pereira (2002). Philosophical Incidence of Logical Programming. *Handbook of the Logic of Argument and Inference,* D. Gabbay et al. (eds.), pp. 425-448, Studies in Logic and Practical Reasoning series, vol.1, Amsterdam: Elsevier Science.

L. M. Pereira (2007). Gödel and Computability. *Progress in Artificial Intelligence*, J. M. Neves et al. (eds.), pp. 63-72, LNAI 4874, Berlin: Springer-Verlag.

L. M. Pereira, G. Lopes (2009). Prospective Logic Agents. *International Journal of Reasoning-based Intelligent Systems (IJRIS)*, 1(3/4):200-208.

L. M. Pereira, T. A. Han (2009). Evolution Prospection in Decision Making. Intelligent Decision Technologies (IDT), 3(3):157-171.

L. M. Pereira (2012). Evolutionary Psychology and the Unity of Sciences - Towards an Evolutionary Epistemology. *Special Sciences and the Unity of Science*, O. Pombo, J. M. Torres, J. Symons, S. Rahman (eds.), Series on Logic, Epistemology, and the Unity of Science, Vol. 24, pp. 163-175, Dordrecht: Springer.

H. Putnam (1960). Minds and Machines. Reprinted in Putnam (1975).

H. Putnam (1975). *Mind, Language, and Reality*. Cambridge: Cambridge University Press.

H. Putnam (1988). *Reality and Representation*. Cambridge: The MIT Press.

A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. Lond. Math. Soc.* ser. 2, 42 (1936) pp. 230-65, correction ibid. 43 (1937) pp. 544-6.

A. M. Turing (1948). Intelligent Machinery. Report to the National Physics Laboratory. Alan Turing papers, King's College Archives, Cambridge, UK.

A. M. Turing (1950). Computing Machinery and Intelligence. *Mind* 59:433-460.

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

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

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

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

Wikipedia (2012). http://en.wikipedia.org/wiki/Functionalism_(philosophy_of_mind)