Turing is Among Us

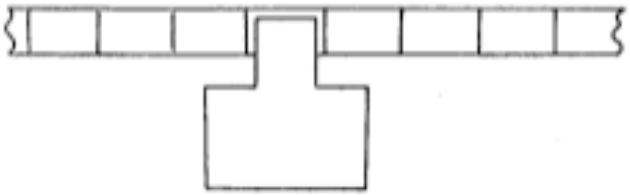
Luís Moniz Pereira

Turing's Relevance - 1

- 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 an *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.

Turing's Relevance - 2

- His originality lies too in the simplicity of the mechanism invoked —a Turing Machine (today a program)— and the proof of existence of a Universal Turing Machine (today a digital computer), which can immitate any other Turing Machine, i.e. execute any program.
- Indeed, Turing Machines simply rely on a finite-state automaton (like a vending machine), and an unbound paper tape of discrete squares (like a paper roll), with at most one rewritable symbol on each square.



Turing's Relevance - 3

- Turing also first *implicitly* introduced the perspective of 'functionalism', by showing what counts is the realizability of functions, independently of the hardware which embodies them.
- This realizability lies in the very simplicity of his mechanism of choice, in relying solely on the manipulation of symbols—discrete as the fingers of one hand—where both *data* and *instructions* are represented with symbols, *both* being subject to manipulation.
- Data and instructions are stored in memory, and instructions double as data and as rules for acting—the stored program idea.

Turing's Relevance - 4

- No one has invented a computational mechanical process with such general properties, which cannot be approximated with arbitrary precision by a Turing Machine.
- In these days of discrete-time quantization, computational biological processes, and proof of ever expanding universe —automata and tape—the Turing Machine reigns supreme.
- Universal functionalism enables the bringing together of the ghosts in the several embodied machines—silicon, biological, extra-terrestrial or otherwise—to promote their symbiotic epistemic co-evolution, for they partake of the same theoretic functionalism.
- Turing is truly and forever among us.

Turing's Relevance - 5

- Turing dared ask if a machine could think. His contributions to understanding and answering this and other questions defy conventional classification. At the start of the XXI century, the 1936 concept of Turing Machine appears not only in mathematics and computer science, but in cognitive science and theoretical biology.
- 'Computing machinery and intelligence' (1950), which defined the Turing test, is a cornerstone of the theory of AI.
- Turing played a vital role in the outcome of WWII, and produced single-handedly a far-sighted plan for construction and use of an electronic computer. His thoughts, then a generation ahead of his time, are still very much alive today.

Alan Turing and Computation - 1

- Gödel left outstanding Hilbert's question of *decidability,* the ***Entscheidungsproblem***, namely if there exists a definite method, applicable to a proposition, to decide if it is provable.
- The question requires a precise definition of *method*. This Turing achieved in 1936 with the **Turing machine**, in his *'On Computable Numbers, with an Application to the Entscheidungsproblem*'.
- Turing recast the question not in terms of proofs, but of computing numbers, a clear claim to have found an idea central to mathematics. As his title states, the *Entscheidungsproblem* was only an application of a new idea, that of computability.

Alan Turing and Computation - 2

- A 'Turing machine' is a finite state automaton supplied with a 'tape' (the analogue of paper) running through it, and divided into sections (the 'squares') each capable of bearing a 'symbol'.
- At any moment there is just one 'scanned square in the machine'. The 'scanned symbol' on it is the only one the machine is 'directly aware' of.
- Turing specifies precisely the repertoire of actions available to the imagined machine. An action is totally determined by the 'configuration' it is in, and the symbol it is currently scanning. It is this complete determination that makes it 'a machine'.

Alan Turing and Computation - 3

- Turing claims that a finite repertoire of symbols actually allows a countable infinity of symbols, but not an infinity of immediately recognisable symbols.
- Note that the tape needs 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.

Alan Turing and Computation - 4

- Turing introduced two fundamental assumptions: discreteness of time and of state of mind. A Turing machine 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.
- 'On Computable Numbers' (rather than 'On Computable Functions') signalled a fundamental shift. Before, things were done to numbers. Afterwards, 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.

Alan Turing and Computation - 5

- With his definition of computable it can be shown that non computable numbers exist. Crucially, the table of behaviour of a Turing machine is finite. Thus, all the possible tables of behaviour can be put in alphabetical order, showing that the computable numbers are countable. Since the reals are uncountable, almost all are uncomputable.
- The problem is identifying those Turing machines which fail to produce infinitely many digits. This is not a computable operation: no Turing machine exists that can inspect the table of any other machine to decide if it will produce infinitely many digits. If one existed, it could be applied to itself, and this can be used to get a contradiction. The *halting problem* cannot be decided by a Turing machine.

Alan Turing and Computation - 6

- From this discovery of a problem undecidable by a machine, one can employ the calculus of mathematical logic and answer the *Entscheidungsproblem* in the negative.
- Alonzo Church announced the same conclusion regarding the *Entscheidungsproblem*. Church's thesis was the claim that effective calculability could be identified with the operations of his very elegant and surprising formalism, that of the λ-calculus—from which Lisp arose.
- Turing equated his result to Church's. "Computability by a Turing machine," wrote Church, "has the advantage of making the identification with the effectiveness in the ordinary [intuitive] sense evident immediately."

Alan Turing and Computation - 7

- Church's thesis is sometimes called the Church-Turing thesis, but the Turing thesis is distinct, bringing the physical world into the picture with a claim of what can be done.
- Attempts, by different approaches, to formalize the *a priori* 'intuitive' notion of computable function proved equivalent, forming a consensus that the intuition was captured. Among others: Church λ-definability, Turing computability, Post canonical systems, Smullyan formal elementary systems, Herbränd-Gödel-Kleene general recursiveness.
- Turing opened the field of computability, offered an analysis of mental activity, and a practical implication: the idea of the computer via the concept of Universal Turing Machine.

The Universal Machine - 1

- 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.

The Universal Machine - 2

- The Universal machine gives Turing claim to have invented the principle of the computer, and not just abstractly.
- One cannot study his machines without seeing them as executable computer programs, stored in symbols along with the data on the tape: the 'modifiable stored program'.
- The Universal machine is the computer where any programs may run. Symbols describe both a program's instructions and the actions they trigger on the Universal machine.
- Also, programs can be manipulated on the tape, even self-modifiable—a possibility AI only recently began to explore.

Machine Functionalism

- Turing machines made all formal proofs 'mechanical'. In1936 such mechanical operations were to be taken as trivial, putting under scrutiny the non-mechanical steps.
- Later Turing abandoned the idea that moments of intuition were uncomputable operations, deciding the computable encompassed far more than could be captured by explicit instruction notes, and enough to include all that human brains did, however creative or original.
- Sufficiently complex machines will have the ability for evolving behaviour never explicitly programmed for. The brain features relevant to thinking are those of the discrete-state-machine description level. Physical embodiment is irrelevant—what is known as 'machine functionalism'.

Gödel, Computability, and Turing - 1

- From 1929 to 1930, Gödel had already solved most of the fundamental problems raised by Hilbert's school.
  One issue remaining was that of finding a precise concept to characterize the intuitive notion of computability.
- Gödel was surprised by Turing's solution, more elegant and conclusive than he had expected.
- Gödel fully understands, beginning of the '30s, that the concept of formal system is intimately tied up with that of mechanical procedure.
- He considers Alan Turing's work on computable numbers an important complement of his own work on the limits of formalization.

Gödel, Computability, and Turing - 2

- 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 defined.
- 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.

Gödel, Computability, and Turing - 3

- Turing attempted a way out from Gödel's incompleteness theorem. The idea is that of adding to the initial system successive axioms, incrementally making it more complete, doing non-deterministic steps once in a while by consulting "a kind of oracle, that cannot be a machine."
- Each "true but not demonstrable" assertion is added as a new axiom. Once a new axiom is added, a new assertion of such a type will be produced to be taken in consideration.
- Turing showed, however, that undecidable statements resistant to the assistance of an external oracle could still be constructed, and the *Entscheidungsproblem* would remain unsolved.

Gödel, Computability, and Turing - 4

- This work, however, had a pleasantly persistent side effect: 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 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.

Is Mathematical Insight Algorithmic? - 1

- 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, in his Gibbs lecture, in 1951, Gödel openly contradicts Penrose:

"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."

Is Mathematical Insight Algorithmic? - 2

- Restricting discussion on the limits of rationality—in contrast to insight—computer science that sees reason in mechanical terms has received the most attention.
  It is at the core of AI, and relevant to Turing's achievement of separating mind and machine.
- AI would thus be mainly interested in the computability viewpoint, involving only a limited part of mathematics and logic.
- But AI's limits cannot be reduced to this scope. It is essential to distinguish *algorithms for problem-solving* and *algorithms simpliciter:* sets of rules to follow in a systematic automatic way, self-modifiable like Turing ventured, *without* necessarily having a specific well-defined problem to solve.

The Software/Hardware Distinction - 1

- The software/hardware distinction—of form and function, present in any machine—appears clear-cut in the digital computer. Diversity of technologies in computers employed to achieve one same function confirms it.
- A program is executable in physically different machines because at the program level of discourse the details of its execution, below a certain level of analysis, are irrelevant. In crude analogy, ink colour and handwriting are irrelevant to the message being conveyed.
- But 'hardware' is not necessarily things physical, but rather that which, at the level of analysis, is considered fixed, given, and whose analysability is irrelevant for some purpose.

The Software/Hardware Distinction - 2

- In a computer, the software prevails 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.
- This way, the teleology of the software is kept in charge, notwithstanding the underlying causality of the physical hardware.

Logic and Consciousness - 1

- "How to introduce consciousness in computers?"
  In the 80's, William Reinhardt questioned how much a Turing machine could know about itself. He conjectured that in arithmetic plus a knowledge operator, a Turing machine can prove "I know I am a Turing machine." Timothy Carlson proved the conjecture in the 90's.
- 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.

Logic and Consciousness - 2

- There is a catch to these models: Their implementation by proponents ends up, with no particular qualms, being on a computer, which cannot help but use symbolic processing to simulate their paradigms.
- The relationship of this argument to logic is ensured by functionalism: Logic can be implemented on top of a symbol processing system, independently of the particular physical substrate. And neural networks can implement a Universal Turing machine, if not logic directly.
- 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.

Functionalism - 1

- The thesis of multiple realizability says 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.
- The first functionalist theory of mind was put forth by Hilary Putnam in1960, and inspired by the analogies he noted between the mind and the theoretical "machines" developed by Alan Turing, now called Universal Turing machines.
- In this light, machines are physical models of abstract processes.

Functionalism - 2

- In 1984 Putnam changed his mind, elaborating an assault computationalism. Putnam's argument fails though.
- What mathematicians and philosophers have failed to appreciate is that the Gödel theorems show that no one —Gödel susceptible or not—can prove the consistency of Peano arithmetic without constructing an infinite proof tree, thus putting a limitation on finitary humans.
- Anti-functionalists employing Gödel's theorems are doomed to failure. Unless we 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 point has resisted appreciation by anti-functionalists.

Functionalism - 3

- 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.
- Functionalism helped lay the foundations for modern cognitive science, and is the dominant theory of mind in philosophy today.
- In permitting mental states to be multiply realized, functionalism offers an account of mental states compatible with materialism, but without limiting the class of minds to creatures with brains like ours.

Evolutionary Psychology and Logic

- Logic provides the overall conceptual cupola, a generic module that fluidly articulates together the specific emerged modules identified by evolutionary psychology.
- It is mirrored by the general computability of Turing's Universal machines, which can execute any program, compute any computable function.
- 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, like a Universal Turing Machine can mimic any Turing machine, and a computer run any programs. This plasticity provides its universal versatility.

Evolutionary Computation - 1

- Turing's approach to machine intelligence, in his 1948 'Intelligent Machinery', was as unhampered as his take on computable numbers in 1936.
- "Does incompleteness of formal systems limit the abilities of computers to duplicate the intelligence and creativity of the human mind?" Turing summarized his position by saying "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. "*The possibility of letting the machine alter its own instructions provides the mechanism for this.*"

Evolutionary Computation - 2

- 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 intelligent machines, with the help of outside oracles.
- The path to artificial intelligence, suggested Turing, is to construct a machine with the curiosity of a child, and let intelligence evolve.

Evolutionary Computation - 3

- Turing's work poses a deep question: "Does computation with discrete symbols give a complete account of our conception of the physical world? Is the world as we see it computable?"
- The Internet search engine is a finite-state deterministic machine, except at those junctures where people, individually and collectively, make a non-deterministic click choice as to which results are selected as meaningful.
- 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.

Natural Philosophy

- Alan Turing's philosophy might appear the ultimate in reductionism, in its atomizing of mental process, its scorn for the non-material.
- Yet 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 - 1

- Epistemology will eventually have the ability to be shared, be it with robots, aliens or any other entity who must 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.

Symbiotic Epistemology - 2

- AI and Cognitive Science, 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, of computational biological processes, and of ever expanding universe—the automata and the tape—the Turing Machine reigns supreme.
- Universal functionalism—Turing's essence—is what enables the bringing together of the ghosts in the several embodied machines (silicon, 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 !

THANKS !