

The mode of computing

Luis A. Pineda

Universidad Nacional Autónoma de México, México

ARTICLE INFO

Action editor: Stephen Fiore

Keywords:

Mode of computing
Natural computing
Representation
Interpretation
Consciousness

ABSTRACT

The Turing Machine is the paradigmatic case of computing machines, but there are others such as analogical, connectionist, quantum and diverse forms of unconventional computing, each based on a particular intuition of the phenomenon of computing. This variety can be captured in terms of system levels, re-interpreting and generalizing Newell's hierarchy, which includes the knowledge level at the top and the symbol level immediately below it. In this re-interpretation the knowledge level consists of human knowledge and the symbol level is generalized into a new level that here is called *The Mode of Computing*. Mental processes performed by natural brains are often thought of informally as computing processes and that the brain is alike to computing machinery. However, if natural computing does exist it should be characterized on its own. A proposal to such an effect is that natural computing appeared when interpretations were first made by biological entities, so natural computing and interpreting are two aspects of the same phenomenon, or that consciousness and experience are the manifestations of computing/interpreting. By analogy with computing machinery, there must be a system level at the top of the neural circuitry and directly below the knowledge level that is named here *The mode of Natural Computing*. If it turns out that such putative object does not exist the proposition that the mind is a computing process should be dropped; but characterizing it would come with solving the hard problem of consciousness.

1. Cognition and computation

Cognition is the study of mental processes, such as perception, thought, attention and memory. It has been with us at least since the Greeks, certainly since Aristotle – e.g., (Shields, 2016) – but in current times it is understood as the study of the mind in terms of information processes or, more specifically, as computational processes. This use was anticipated by Charles Babbage's Analytic Computing Engine (ACE) in the XIX century, that could automate the computation of mathematical functions, and the manipulation of symbols more generally, but it was properly introduced with the Turing Machine (TM) in 1936 (Turing, 1936) when an abstract and general notion of computing was presented. The TM allowed theorists to model mental processes, such as playing chess, a paradigmatic form of thinking, that could only be performed by people before computers were available. The potential scope of computing machinery for simulating the mind was stated explicitly by Turing with the presentation of the Imitation Game, better known as the Turing Test, and with his expectation that digital computers will eventually compete with humans in all purely intellectual fields (Turing, 1950). Turing also held that a computing engine able to pass the imitation game should be ascribed intentionality and consciousness, in the same way that we ascribe such properties to other people, despite that we only have access to our own internal subjective experience. This position gave rise to the more general claim

that the mind is a computational process carried out by a TM. This is the central tenet of current cognitive studies. This view holds implicitly that the TM, which is a human invention, captures a putative phenomenon which can be designated as *natural computing*, that was never observed before the invention of computers.

From a mathematical perspective, TMs compute functions — see Boolos and Jeffrey (1989). Computing is simply the process of obtaining the object in the codomain – i.e., the value – assigned to a given object in the domain – i.e., the argument – for all the objects in the function's domain. This process is achieved by means of an algorithm that is executed by the TM. If the algorithm terminates for all of its arguments is said to be effective, and the functions that have an effective algorithm are said to be computable. Functions that do not have an effective algorithm, or have no algorithm at all, are said to be uncomputable. TMs are enumerable and can be placed in a list including all possible algorithms. The hypothesis that all computable functions are computed by one or another algorithm in such a list is called Church's Thesis or Church-Turing Thesis. Church's Thesis also states that the TM is the more general computing engine that can ever be conceived, and that every alternative general enough computing device computes the set of functions that are computed by the TM exactly. The thesis postulates or defines the very notion or concept of computability, and

E-mail address: lpineda@unam.mx.

<https://doi.org/10.1016/j.cogsys.2023.101204>

Received 10 September 2021; Received in revised form 24 November 2023; Accepted 15 December 2023

Available online 3 January 2024

1389-0417/© 2024 Elsevier B.V. All rights reserved.

is the fundamental tenet of computer science. However, the inverse problem of finding out the algorithm that computes a given function, for all computable functions, is open — it would be required to know in advance whether the function is computable or not by means of a computing engine more powerful or qualitatively different in an essential way from the TM, which cannot exist by hypothesis, and the thesis cannot be proven. Conversely, the thesis would be refuted if such a machine were discovered.

Adopting the current view of cognition formally – as opposed to metaphorically – involves postulating that the mind is a computing process, that Church Thesis is true, and that natural computing is powered by the TM. Consequently, that the mind is produced by a set of effective algorithms and cannot compute uncomputable functions. However, it might turn out that there is natural computing but powered by a machine that is not equivalent to the TM in an essential way; hence, Church Thesis would be refuted or at least its scope would be limited; and it is also possible that the mind is not caused by a computing engine and there is not natural computing after all.

2. Computation, representation and interpretation

Computing machines manipulate representations but do not make interpretations. This is shown very clearly in computing engines built with mechanical devices. Babbage's Analytical Engine, for instance, used mechanical gears to express and perform computations (Rojas, 2021). The input argument and the output value of the computed function were represented by the gears positions when the computation was started and ended, respectively, in the same way that mechanical clocks display the time. The computing process was performed by the mechanical machinery that ran continuously until it came to a halt, but the machine did not have the functional nor the structural ingredients to interpret such representations – did not understand or had knowledge in any conceivable sense – and the interpretation was performed by people, in the same way that the positions of the clock hands are interpreted as the time by humans, but clocks do not understand the concept of time.

Representation are material objects that use a physical medium with patterns marked on it that express information. Intuitively, natural representations appeared when the phenomenon of communication first emerged, most probably very early in the history of life, perhaps as movements or gestures to express basic intentions such as where to go to find food or shelter, or run away from a predator. The representations are *produced* and *interpreted* by intentional agents. The former is a process that “places” a mental content or intent on a physical medium, and the latter is the reverse process that inputs the representation and places its content in the interpreter's mind. Hence, the production and interpretation of representations indicates clearly the agent has a mind already.

Producing and interpreting representations are subsumed within the more general processes through which intentional agents perform actions with intent and interpret the material world. Such actions and interpretations depend on the particular brains, that support the mental endowment of natural species and individuals, such as perception and motor behavior and, in the case of humans, thought. Hence, the mental world is constituted by interpretations, that are subjective, as opposed to the objective material reality. Consequently, representations, that express mental content, represent interpretations.

A representation is *declarative* if the result of interpreting it places the concept of an individual object, property or relation in the interpreter's mind. For instance, the speech act “this is a ball” accompanied by a pointing gesture, is a declarative representation. There are also representations that express instructions or commands that the interpreter should perform directly. In this case, interpreting the representation consists of performing an action, such as collecting food or running away from a predator. This kind of representation is *procedural*. Performing actions is accompanied by an experience or feeling,

and interpreting the representation and experiencing performing the actions, are two aspects of the same act. Representations can have a dual declarative and procedural aspect, as the interpretation can produce mental content, that the interpreter may be conscious of, and involve performing actions, that the interpreter may experience or feel.

The paradigmatic form of human natural representation is the spoken language. The medium is the air wave, and the sound patterns codify the phones, words and sentences. Speech acts, either spoken, gestural or multimodal, express the knowledge, beliefs, desires, intentions, feelings and affections of the agent. Natural representations evolved into conventional representations that were not necessarily linguistic, such as diagrams, paintings and sculptures, with deep and subtle meanings, that allowed the transmission of mental contents beyond the immediate face to face interaction. Then came the invention of manuscript symbols and the appearance of written language, that uses the paper as the physical medium and the ink to mark the patterns, and allowed communication between agents at different spatial and time locations. The written language served also as the first form of artificial memory, that allowed the registering of commercial, biographical or historical records, and the appearance of literary art. It also made possible, with the invention of numerical notations and algorithms, the ability of making complex calculations beyond what is possible to calculate mentally. The next great chapter of the history of representations was the invention of the printing press. Gutenberg automated the production of textual representations using movable printing types with standard forms for letters, digits and punctuation marks, that allowed the impression of different texts through a manual but regimented process, using a finite number of types, making possible the massive production of texts, with the consequent cultural explosion.

The Turing Machine can be thought of as the next main cultural evolutionary event in the history of representations. The invention was made in the tradition of Babbage's Analytical Engine, which was already a universal computer, but Turing introduced the printed text as the representational format, instead of Babbage's gears. While Gutenberg's machine automated the process of printing texts, the TM automated their transformation through mechanical well-defined procedures. The TM uses a tape divided into cells as its representational medium, each holding an instance of a type symbol of a finite alphabet, and the tape as a whole holds a textual representation. The machine can be in one of a finite number of possible states, and has a scanning device placed on a specific cell at the current state. It also has a control system that executes the instructions included in the so-called transition table. Every instruction specifies the action that is performed and the state that the machine moves to for a given current state and symbol being scanned. The actions that the machine can do are only to write a symbol on the cell being scanned, or to shift the scanner one cell to the right or to the left. The machine halts when the action and next state for the current state and symbol being scanned is not specified in the transition table. The instructions specify the algorithm, and all computable functions are computed by this very simple machine. The MT is an ideal model which assumes that there are not memory limitations and computations are performed infinitely fast, so the result of a particular computation appears on the tape immediately once the machine is started.

The interpretation of textual representations on the machine's tape is performed by people according to the interpretation of its parts, down to the words, and their mode of grammatical combination, along the lines of Frege's principle of compositionality, but not by the TM itself that lacks the functional and structural ingredients to make interpretations. The interpretation is also made in relation to a well-defined set of interpretation conventions, as stressed by Boolos and Jeffrey (Boolos & Jeffrey, 1989), involving the notation (e.g., whether “111” is interpreted as three, as seven or as one hundred and eleven, in monadic, binary or decimal notation, respectively), the order and direction in which symbols and expressions should be read, and perhaps other pragmatic considerations, that the interpreter knows, possibly

implicitly, and such knowledge is deployed in the interpretation process. In particular, the interpretation conventions include the so-called standard configuration, which states the format of expressions in relation to the position of the scanner when the process is started and ended, such that the representations at the corresponding states of the TM can be interpreted by people directly. The strings on the tape at intermediate states of the process need not be interpreted and may not have an interpretation, but such intermediate strings are contingent and the interpretations conventions are meant to guarantee that the representations can be interpreted at the initial and final states. Hence, the objective but arbitrary manipulation of symbols independently of an interpretation and a set of interpretation conventions does not constitute a computation. A computing engine always involves the device or natural phenomenon that maps or transforms the input into the output, and the agent that makes the interpretation or the semantic attribution. Put it simply and directly, there is no computation without representation and interpretation.

Textual representations were later “internalized” into digital computers with the development of the von Neumann architecture and the introduction of the Random Access Memory (RAM). In these machines each basic memory cell or byte contains an instance of a type – e.g., a symbol of the ASCII alphabet – representing a letter, a digit or a punctuation mark, and the content of a physical or logical sequence of bytes is a textual representation. Algorithms are expressed through computer programs as strings of texts, that are compiled and placed into the computer memory as binary codes, which are also textual representation using only two types of digits. Images of the diverse modalities of perception and action are codified as strings of digits too. Modern digital computers are implementations of the TM that manipulate representations, but the interpretations are made by people, and computing machines know, experience and feel as much as carving tools, pens and the printing press.

Written text, specially in the printed form, is perhaps the paradigmatic form of representation, and its adoption as the representational format by Turing contributed greatly to the success of the TM and to the impact of computing theory and technology. However, this choice prevents the direct use of non-textual representations, natural or conventional, such as hand and body gestures, diagrams, pictures, paintings, sculptures, music pieces, etc., that need to be translated into a textual format, using digits and numerical notations, to be used by computers. This limitation impacts also in representing the natural world that is presented to the agent through signals and forces, giving rise to images of the different modalities of perception and action, that have to be translated into textual representations in order to be used by computers. This is a huge technological challenge, and despite that it has been addressed very successfully by science and technology, it is nevertheless implausible that humans and non-human animals internalize representations and the external world into a conventional typographic representational format.

Representations used by general enough computing machines, such as Babbage’s Analytical Engine, differ from the TM in the medium and markings that they use, and also in their interpretation conventions. However, according to Church Thesis, all practical computing engines are physical implementations of Turing’s model. In fact, there is a set of known computing machines that are equivalent to the TM, and there may be a open set of machines yet to be discovered. However, the simple internalization of the TM or any equivalent machine into digital computers does not make it intentional, neither the corresponding representation.

The objective character of representations and its use in digital computers contrasts with the subjective notion of representation in cognitive psychology and philosophy of mind. Informally, representations in these latter disciplines are mental objects that are causal and essential to thought, intentional behavior and consciousness. These putative objects would be the product of evolution, and consequently, natural

representations. Although these mind objects cannot be observed, measured or characterized, they were nevertheless used in pre-scientific speculative or introspective methods, but later on were rejected as genuine objects of scientific study, as in behaviorism. However, the TM and the availability of practical digital computers suggested the use of TM representations for modeling natural cognition. Chomsky made the proposal explicit with the introduction of syntactic structures (Chomsky, 1957) and his refutation of Skinner’s *Verbal Behavior* (Chomsky, 1959). Chomsky’s implicit move was to establish a direct analogy between the material “external” representations and the representations processed by the mind and held in memory, which are “internal”. More generally, the representational view of cognition was extensively argued by Fodor (Fodor, 1975) who proposed that the object of cognition was a mental language, the Language of Thought (LOT) or *Mentalese*, with a syntactic or compositional structure and a propositional and representational character, that was inspired in the TM directly. However, the internalization of TM representations into humans and not human animals does not make them intentional neither.

The TM is external to the mind but the interpretation is a mental process; however, if the interpretation is also a computational process, then the natural computing engine that powers the mind needs to include both the external and the internal computing machinery. Otherwise, there would be an essential part of cognition, namely, the one that accounts for interpretations, that would not be computational. Hence, the “internal” representation and its interpretation cannot be dissociated.

A proposal to the form of such representations comes from the cognitive psychology and the neuroscience. In particular the objects held in working memory (Baddeley, 1992) and long term memory, both semantic and episodic (Tulving, 2002), are putative natural representations whose interpretations are presented to the mind when they are registered, recognized and retrieved. Baddeley’s working memory consists of the central executive, that stores and processes information simultaneously, the so-called phonological loop, which stores and rehearses speech-based information, and the so-called visuospatial sketchpad, which manipulates visual images. Such putative representations are conscious noetically, and allow the agent to be aware of the world, and possible auto-noetically when the agent is involved in reflective thinking. Semantic memory recall is noetic, and Tulving’s episodic memory recall, which involves a subjective notion of time and a sense of the self, is the paradigmatic form of autonoetic consciousness, that places the intentional agent in relation to its autobiographical episodes, his or her potential future and counterfactual situations. There is also evidence that there is a scene construction process using information from disparate sources that is used in diverse psychological functions such as episodic memory retrieval and episodic future thinking, but also in navigation, imagination and vivid dreaming, among others (Hassabis et al., 2007; Hassabis & Maguire, 2007). Whether such image and scene representations are equivalent to the TM would depend on their structure – although it may turn out that working and long term memory representations have essentially different properties – but the computing process would need to perform the interpretation too, which is not a part of the TM, as has been extensively argued. There may be as well procedural representations whose interpretation is experienced. The discovery of natural computing would involve finding out the properties of such representations so they can be interpreted – say, by an external observer – and why the interpretation process involves consciousness and experience. However, if it turns out that mental representations and interpretations cannot be not characterized, the whole idea of natural computing should be dropped.

3. Knowledge and system levels

The distinction between the two senses of representation – material versus mental – can be seen in terms of system levels. Complex phenomena or devices can be studied at different levels of abstraction

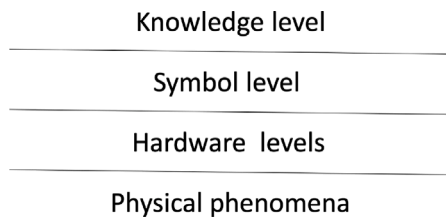


Fig. 1. Newell's main system levels.

or granularity, such that each level has its own theoretical terms and laws of behavior, and the phenomena at each level can be studied independently of other levels. For the case of digital computers Allen Newell distinguished, from bottom to top, the physical phenomena, the device, the electronic circuits, the logic circuits, the register transfer or computer architecture levels, with the symbol and the knowledge levels at the top of the hierarchy (Newell, 1982). A simplified version of Newell's levels collecting the hardware levels together is illustrated in Fig. 1.

The TM proper is defined at the symbol level, where symbolic manipulation takes place, and reduces to all the levels below, down to the physical phenomena. Meaning, on its part, belongs to the knowledge level. Newell stated that the knowledge level emerges from but is not reducible to the symbol level. He also sustained that the medium of this level is knowledge itself and that the only rule of behavior at this system level is what he called the *Principle of Rationality*. He – in conjunction with Simon – also postulated the *Physical Symbol System Hypothesis* sustaining that a system of grounded symbols provides the necessary and sufficient conditions to generate general intelligence (Newell & Simon, 1976). This latter hypothesis refers also to the symbolic manipulation performed at the symbol level. From Newell and Simon's claims it can be elicited that the knowledge level holds the representations that result from interpreting the structures at the symbol level, thus digital computers can understand and be conscious.

According to the previous discussion this latter claim cannot be sustained, but Newell's hierarchy can be still adopted, but holding that the knowledge level contains the interpretations ascribed to symbolic structures by people. Hence, in the present re-interpretation of Newell's hierarchy, the symbol level and all the levels below, down to the physical phenomena, are implemented in machines, but the knowledge level is human knowledge.

An alternative system level hierarchy was proposed by Marr (Marr, 1982). This has three levels which are, from top to bottom, the computational or functional, the algorithmic and the implementational. The first refers to the specification of the mathematical function that models the mind's faculty that is the object of study, such as vision or language; this is human knowledge and the top level in Marr's hierarchy corresponds to the knowledge level. The algorithmic level is constituted by the computer programs or algorithms that compute the function properly and corresponds to the symbol level. Finally, the bottom level includes all hardware and software aspects that sustain the algorithmic level, but are contingent to computations, such as the programming language in which the algorithm is coded, or the particular computer in which it is run.

From a third perspective, the sense of representation at the knowledge level corresponds to the one contested by Searle in the story of the Chinese Room, in which an English speaking person, who does not understand Chinese, answers nevertheless questions in Chinese by following instructions and data, without been aware of the meaning of the questions and their answers (Searle, 1980). In terms of the hierarchy of system levels, all operations of the person in the room are performed at the symbol or algorithmic level, but such individual is not aware or conscious of the meaning of the Chinese expressions because he or she does not represent at the knowledge level. Searle dubbed the

view that computers represent at the symbol level but do not do so at the knowledge level *weak AI*, in opposition to the view that computers can represent at the knowledge level, that Searle refers to as *strong AI*.

The attribution of meaning to computing systems should also be seen in relation to whether subjects of study are actual mechanisms, such as robots, or whether computers are used for developing and testing cognitive models. In the former, and considering that TMs do not ascribe meaning, such devices are similar to standard machinery – i.e., radios, TVs, etc. – which are unaware of the meanings of the signals they manipulate for human consumption. In the latter, computers are unaware of the meaning of the representations that they are supposed to model too, but the theorists who devise such models interpret and attribute meanings to such processes. So, once again, computer systems implement cognitive models at the symbol level but interpretation and representation belong to the knowledge level which resides in the mind of human experimenters.

Newell's, Marr's and Searle's models refer to standard computations using the TM, but do not intent to address the case of the putative natural computing. However, these views can be used as analogies of how the mind could relate to the brain: there would be a natural system level implemented in the brain, corresponding to the symbol level. The characterization of such system level would include the specification of the code used by representations, of the operations performed upon them, and of the interpretation process. Such natural computing level would be directly below the knowledge level at the top of the brain's system levels – the level of interpretations – which would need to be characterized too, including its conscious and experiential aspect.

4. Connectionism and alternative notions of computing

The view that the mind reduces to symbol manipulation was questioned by the Parallel Distributed Processing (PDP) program or Connectionism and its implementation through Artificial Neural Networks (ANNs). This program holds that symbolic systems or TMs failed to model appropriately most mind processes, such as perception, memory, language and thought – see the Preface of Rumelhart's text (Rumelhart et al., 1986) – and proposes instead that intelligence emerges from the interactions between a large number of simple computing units expressing *distributed representations* that are computed by massive parallel processes. Distributed representations are opposed to TMs in the relation between the units of memory and the units of content or the concepts that such structures represent: in the TM this relation is one-to-one, so basic concepts are stored in mutually exclusive memory regions, while in distributed systems it is many-to-many (Hinton et al., 1986), hence concepts expressed by distributed representation use memory regions that may overlap in arbitrary ways; for instance, a particular neuron can contribute to the representation of more than one concept, and a concept may share neurons with other concepts. In addition, distributed representations can generalize and the extension of the represented concept may contain individuals that were not considered when the representation was originally formed if they are similar enough to other individuals in such extension. This latter effect cannot be expected in local representations.

Whether symbolic and connectionist systems are different models of computing has been subject of intense debate. Fodor and Pylyshyn, for instance, emphasized that both symbolic and distributed systems are meant as representational, but highlighted the limitations of ANNs to express syntactic structure and for holding information as standard declarative memories, and denied the special character of connectionist algorithms (Fodor & Pylyshyn, 1988). Nevertheless, they granted that connectionist systems could implement the mentalesse, which emerges from the workings of the natural neural networks. This relation has been described in terms of system levels (Rueckl, 1991), but using Marr's system levels (Marr, 1982), as illustrated in Fig. 2.

A very large effort has been made to reduce the mentalesse to connectionist systems; in the computational front this is interpreted

Functional: *Mentalese*

Algorithmic: Neural Networks

Physical Substratum

Fig. 2. Mentalese and its implementation in ANNs.

as expressing symbolic representations and reasoning through ANNs (Chalmers, 1990; LeCun et al., 2015, 1998) or as integrating ANNs, Bayesian and logical systems (Besold et al., 2017); and implementing symbolic representation and reasoning with distributed representations is possibly the greatest challenge for current Deep-Learning Artificial Neural Networks (DL-ANNs) and there are large efforts to achieve this end — e.g., (Graves et al., 2016).

However, connectionism and ANNs deviated from the original program in practice because ANNs are specified as Turing-computable functions, implemented with standard data-structures and algorithms, and run in standard digital computers, thus ANNs are TMs. ANNs are expressed and computed with numerical matrices and operations, alike in every respect to standard scientific or engineering applications. It has also been shown that every TM can be expressed as a recurrent neural-network and these are equivalent machines (Siegelmann & Sontag, 1995; Sun et al., 1992). Hence, implementing ANNs with TMs fuse the symbolic and the connectionist at the algorithmic system level, and the interpretation is performed according to the standard configuration and interpretation conventions of the TM. If the claim is that ANNs express distributed representation, but practical implementations use local memories, the distributed property is only simulated but not actual. Furthermore, the global interpretation of the distributed representations is dissolved and it is only present as a subjective intuition of the human interpreter. Hence, ANNs, including current DL-ANNs, contrary to Rumelhart's claim, do not contest the TM, and do not challenge Church Thesis.

5. The computational quadrants

The distinction between declarative and procedural representations shows up in Artificial Intelligence (AI) as the strong opposition between *symbolic* versus *sub-symbolic* systems. According to the so-called Knowledge Representation Hypothesis, the knowledge exhibited by a computational process is considered symbolic or “representational” if its structural ingredients can be rendered as propositions with a linguistic character, that are also causal and essential to the behavior exhibited by the computational agent (Smith, 1985); otherwise, the systems are said to be sub-symbolic or “non-representational”. The former kinds of systems are declarative whereas the latter have implicit knowledge embedded in opaque structures — e.g., numerical algorithms, neural networks, etc. — which is used or deployed, and are procedural. However, the Knowledge Representation Hypothesis does not commit to whether or not computers make the semantic attribution.

Strong and weak AI are symbolic, but while the former is meant as representational, in the sense that such systems make interpretations, can be conscious and can experience the world, the latter is understood as non-representational, so computers carry on with the symbolic manipulation but people make the interpretation. ANNs and diverse algorithmic approaches are sub-symbolic, but are meant as representational too, at least in its initial formulation, and are alike to strong AI in this respect. There are also sub-symbolic approaches that are meant as non-representational. This seems to be the position adopted in Brooks's program of the so-called embedded architectures for modeling bio-inspired robotic mechanisms (Brooks, 1991a, 1991b).

	Symbolic	Sub-Symbolic
Representational	Strong AI	ANNs, Specific algorithms, etc.
Non-Representational	Weak AI	Embedded Architectures

Fig. 3. Computational quadrants.

Brooks's move was to construct robots with sensors and actuators of different sorts, including control and dynamical systems implemented physically or simulated with digital computers using procedural representations; however, the computing engine was no longer meant to be causal and essential to the behavior exhibited by the robot, the basic tenet of AI, and was only used as the modeling tool, as in most current scientific disciplines. Brooks dubbed his view with the lemmas “Intelligence without Representation” and “Intelligence without Reason” challenging directly the cognitive and AI tradition. Nevertheless his program gave rise to a large body of work developed within the so-called *Embodied Cognition* and is considered an AI paradigm (Anderson, 2003). In any case, Brooks's robots do not make interpretations by hypothesis, and the view is alike to Searle's weak AI in this respect. Brooks's proposal can be included in a psychological current of thought that goes back to functionalism, classical behaviorism, Gibson's ecological psychology and situated cognition, including enactivism (Froese & Ziemke, 2009), which is non-representational and non-computational, and opposes the computational view of mind and AI (Chemero, 2013).

In summary, AI systems are either symbolic or sub-symbolic, according to whether they are declarative versus procedural, and representational versus non-representational, depending on whether or not the machine is assumed to make the semantic attribution and/or experience the world. The AI views in the four quadrants resulting from the combination of these two variables with their corresponding values are illustrated in Fig. 3. All four quadrants are implemented with the standard digital computers, hence use local representations, and are interpreted according to the standard configuration and interpretation conventions of the TM.

The four quadrants are seen from the perspective of the human interpreter as follows:

1. Top-Left: the human makes conscious interpretations and thinks that the computer does so as well;
2. Bottom-Left: the human makes conscious interpretations but thinks that the computer does not do so;
3. Top-Right: the human interpreter thinks that the computer experiences the world;
4. Bottom-Right: the human interpreter thinks that the computer does not experience the world.

Of course, being conscious and experiencing the world are not mutually exclusive mental behaviors, and humans may have feelings associated to interpreting declarative representations, and experiencing the world may be accompanied to making conscious interpretations. Likewise, robots using symbolic representations for thinking and talking in addition to sub-symbolic representations supporting perception and motor behavior, would be thought of as having conscious content and experience the world in the strong AI view; but in the weak AI view, robots would be thought of as lacking consciousness and experience, even if their behavior is un-distinguishable from the behavior of humans and non-human animals in every respect.

Artificial Intelligence has today a very visible position in science and technology, and has created great social expectations. DL-ANNs,

reinforcement learning and machine learning in general, have permitted the creation of very sophisticated classification, prediction and diagnosis techniques that have allowed the development of very sophisticated natural language and vision systems, as well as a large number of scientific applications, with great actual and potential impact. This trend will continue most probably for the foreseeable future. However, the “mental competence” of today’s most sophisticated programs and robots is very limited when is compared with humans and non-human animals with a developed enough nervous system, and it is possible that there are limits to the explanatory power of the computational view of the mind. The challenge is not to AI but most fundamentally to the whole notion of cognition based on the TM. Hence, the dilemma is whether to abandon the computing paradigm and use computers only as modeling tools, or to extend the TM to model better the phenomena of the mind.

6. Relational-indeterminate computing

Turing Machine computations are determinate by necessity. Turing stated that the determinism of the machine is “rather nearer to practicability than that considered by Laplace” (Turing, 1950, s.5). This corresponds to the popular intuition that if something is programmed it is predetermined. Thus, if an algorithm terminates it always produces the same value for the given argument. In the original 1936 paper Turing distinguished between “the automatic” and “the choice” machines; the corresponding transition tables specify a function and a relation, respectively, so that the former machine has only one possible action at every state, while the latter can have many. This kind of indeterminacy shows up in the interpretation of ambiguous sentences or in the exploration of a problem space in which more than one solution can be found for a given problem, such as equally good moves in a chess position. However, all relations can be defined as compositions of functions, and computing a relation is construed as computing its constituent functions, although at the cost that its intrinsic indeterminacy is not considered. Turing did not use choice machines further in the 1936 paper, but they were later developed in the theory of non-deterministic automata e.g., (Hopcroft et al., 2006), which was well established when the 1950 paper was published. Hence Turing must have been well aware of this kind of non-determinism when he compared the determinism of digital computers with the one advocated by Laplace.

Turing himself suggested a stochastic extension of the machine in the 1950 paper, where he proposed to use the next digit of the expansion of π as a random element. However, although such number is not known in advance, it is predetermined, and computations using such pseudo-random numbers are still fully determinate. Furthermore, he stated that it would not be normally possible to distinguish whether such machine involve genuine or pseudo-random elements. Nevertheless, he described two problem solving strategies, which he called “the systematic method”, which is deterministic, and “the learning method”, which is stochastic. Turing suggested that the latter may be more efficient in large problems that may have many solutions. The observation gave rise to stochastic computing that employed pseudo-random numbers, and later on random numbers produced out of natural signals that could be measured in the environment, and whose value could not be predicted. However, the random parameter can be thought of as a hidden or unknown argument of the function being computed, and such form of stochastic computing is still subsumed within the TM paradigm. Stochastic algorithms are programmed and executed with standard programming languages in ordinary digital computers, and are widely used in many fields of computer science. However, the “indeterminacy” is due to lack of knowledge or incomplete information, but not to actual indeterminate phenomena, that cannot occur in a fully deterministic view of the world. In particular, the Central Processing Unit (CPU) of standard digital computers implements the fetch cycle that retrieves and executes the instructions from the memory in a fully deterministic fashion.

Genuine indeterminacy is measured with the entropy. Laplace’s determinism can be considered as the limiting case in which every possible state of the universe has only one possible next state. However, this condition can be relaxed allowing that every state has a number of possible next states. Hence every possible state has an associated probability distribution of moving to a next state, and the uncertainty of the transition can be measured with Shannon’s entropy directly. The overall uncertainty of the universe can be thought of as the average entropy of all possible states. The next state occurs randomly according to the probability distribution associated to the current state. On its part, the physical entropy of the universe augments whenever energy is used and work is done; so the universe moves into a new state whose indeterminacy is increased, and physical processes are not reversible. If it is assumed that the physical and the information entropy go hand in hand, the overall uncertainty of the universe augments as well — hence, the number of possible states and their corresponding probability distributions of moving to a next state “are updated” once a change of state has taken place. Laplace’s determinism corresponds to a reversible world with zero entropy. Likewise, the standard formulation of the Turing Machine does not include the entropy because it is fully determinate and its entropy is zero too. This discussion is elaborated in relation to agency, decision making and intentionality in Section 12.

However, the entropy can be included in the formulation of the computing engine. A proposal to this effect is *Relational-Indeterminate Computing* (RIC) (Morales et al., 2022; Pineda, 2021; Pineda et al., 2021). In this formalism the basic object of computing is the mathematical relation, instead of the function, and evaluating a relation is construed as selecting one of the objects associated to the given argument randomly. It is considered that this is a genuinely indeterminate choice, and a machine using this basic evaluation mechanism is indeterminate.

RIC has three basic operations: *abstraction*, *containment* and *reduction*. Let the sets $A = \{a_1, \dots, a_n\}$ and $V = \{v_1, \dots, v_m\}$ be the domain and the codomain of a denumerable or a finite relation $r : A \rightarrow V$. For any relation r there is a function $R : A \times V \rightarrow \{0, 1\}$ such that $R(a_i, v_j) = 1$ or *true* if the argument a_i is related to the value v_j in r , and $R(a_i, v_j) = 0$ or *false* otherwise. Let r_f and r_a be two arbitrary relations from A to V , and f_a be a function with the same domain and codomain. The operations are defined as follows:

- **Abstraction:** $\lambda(r_f, r_a) = q$, such that $Q(a_i, v_j) = R_f(a_i, v_j) \vee R_a(a_i, v_j)$ for all $a_i \in A$ and $v_j \in V$ — i.e., $\lambda(r_f, r_a) = r_f \cup r_a$.
- **Containment:** $\eta(r_a, r_f)$ is true if $R_a(a_i, v_j) \rightarrow R_f(a_i, v_j)$ for all $a_i \in A$ and $v_j \in V$ (i.e., material implication), and false otherwise.
- **Reduction:** $\beta(f_a, r_f) = f_v$ such that, if $\eta(f_a, r_f)$ holds $f_v(a_i) \in \{r_f(a_i)\}$ for all a_i , where $f_v(a_i)$ is selected randomly, with an appropriate distribution centered around f_a . If $\eta(f_a, r_f)$ does not hold, $\beta(f_a, r_f)$ is undefined — i.e., $f_v(a_i)$ is undefined — for all a_i .

The λ and η operations are fully determinate, but the relational evaluation β makes a random choice and introduces the indeterminate character to this form of computing.

The form of RIC in which the objects of computing are finite relations represented as standard tables is called *Table Computing*. The table’s columns correspond to the relation’s arguments, the rows to their values, and the cells are marked 1 or 0 depending on whether or not the corresponding argument and value are related. The λ and η operations are local to the corresponding cells of the tables representing the relations f_a and r_f , and β to the corresponding columns; and all three operations can be performed in parallel.

Relations have an associated entropy, which is defined as follows. Let μ_i be the number of values assigned to the argument a_i in r ; let $v_i = 1/\mu_i$ and n the number of arguments in the domain. In case r is partial, we define $v_i = 1$ for all a_i not included in r . The *computational entropy* e — or the entropy of a relation — is defined here as:

$$e = -\frac{1}{n} \sum_{i=1}^n \log_2(v_i)$$

The computational entropy is designated e to distinguish it from Shannon's information entropy, which is usually designated s , although both are different perspectives of the same phenomenon.

Functions are relations in which every argument has a unique value, or no value in the case of partial functions, but in any case are fully determined. Hence, the entropy of functions, either total or partial, is zero. The TM is the particular computing machine whose entropy is zero.

The number of functions ξ included in a relation r is the number of combinations that can be formed by selecting an argument of each column at a time, out of the μ_i possible values; if the relation is not defined for the argument i , $\mu_i = 1$. This is:

$$\xi = \prod_{i=1}^n \mu_i$$

However, the entropy is the average indeterminacy of the arguments, and $\xi = 2^{e^n}$ or simply $\xi = 2^{en}$. This can be seen directly considering that the indeterminacy of each argument is $-\log_2(1/\mu_i) = -(\log_2(1) - \log_2(\mu_i)) = \log_2(\mu_i)$, and $\mu_i = 2^{\log_2(\mu_i)}$; hence,

$$\prod_{i=1}^n \mu_i = 2^{en}.$$

Representations in RIC are construed as functions representing individual objects; and classes are construed as relations formed through the λ -abstraction of such functions. Individual membership is computed through the η operation directly. In this representation the relation of a memory unit or unit of form –a marked cell– may contribute to the representation of more than one unit of content or individual concept –a function– which may share marked cells with other functions; and the relation between memory units and units of content is *many-to-many*. The number of objects that are included in a relation is ξ . This is a very large number, even for moderate values of the entropy and a relatively small number of arguments.

The basic RIC model has been enriched with a weighted version in which the content of every cell of the table is an integer number representing the times such cell has been used in the representation of one or another function (Pineda & Morales, 2022, 2023). In this version, the column corresponding to each argument becomes a probability distribution Ψ , whose entropy is Shannon's entropy directly; the entropy of the relation is the average entropy of all columns; and the argument of a β -reduction operation becomes a normal distribution on the column centered at the actual argument's value, named ζ . Let be Φ the scalar product of ζ and Ψ . The value $f_v(a_i)$ for each argument a_i of the reduction operation $\beta(f_v, r_f) = f_v$, is selected randomly from Φ . The number of functions stored in the relation r is still 2^{en} , although this is now the number of salient functions with a large enough weight, instead of the total number of functions.

Table computing can be thought of as a distributed and stochastic extension of the TM. The table plays the role of the tape, and the symbols stored in the cells are instances of the machine's alphabet. The machine has a scanner placed on top of each cell, allowing the parallel manipulation of cells and columns directly. The machine operations are stated in a transition table, but the algorithms implementing the λ , η and β operations are performed in very few computing steps, always terminate, and constitute *minimal algorithms*. The conceptual extension is that representations are genuinely distributed, every state has an entropy, and the machine is intrinsically entropic.

7. Entropic associative memory

Table Computing has been used to build the Entropic Associative Memory (EAM) (Morales et al., 2022; Pineda et al., 2021; Pineda & Morales, 2022, 2023). EAM uses one or more tables for representing objects, which are called *Associative Memory Registers* (AMRs). Individual objects are represented through feature-value structures or functions, and memory register, recognition and retrieval are implemented with

the λ , η and β operations, respectively. The λ -register operation builds a distributed representation through the weighted disjunction of the registered objects; η -recognition accepts or rejects objects depending on whether or not the cue is included in the memory using the logical material implication; and β -retrieval constructs the representation of a novel object randomly, out of the AMRs but modulated by the corresponding cue. The memory has been used to store, recognize and recover hand written digits and letters, both with complete and partially occluded cues (Morales et al., 2022; Pineda et al., 2021); Mexican Spanish phones (Pineda & Morales, 2022); and pieces of clothe, shoes and bags, including associated and imaged objects, and to make associations chains (Pineda & Morales, 2023).

AMRs have an operational range of entropy values, and memory recognition and retrieval depend on the entropy of the AMR. For low entropy values precision is high but recall is low and vice versa; however, there is an entropy interval in which precision and recall are both satisfactory; and the performance of the system as a whole obeys an entropy trade-off.

The Entropic Associative Memory resembles natural memory in that it is:

1. *Associative*, in opposition to being accessed through addresses, such as standard RAMs used in symbolic systems;
2. *Declarative*, as the λ -register, η -recognition and β -register operations are declarative, and the stored objects are represented through overt functions, in opposition to sub-symbolic systems using numerical representations;
3. *Distributed*, in opposition to local memories of TMs;
4. *Parallel*, as memory register and memory recognition are cell-to-cell operations, and memory retrieval is a column-to-column operation, which can be computed in parallel if an appropriate hardware is provided;
5. *Abstractive*, as the λ -register operation produces the disjunctive abstraction between the registered object and the content of the memory;
6. *Productive*, as there are novel objects emerging from the combination of the objects stored explicitly; these objects do not constitute innate nor empirical knowledge, but are used as “pivots” for the recognition and construction of novel objects;
7. *Strong negation*, as the recognition operation uses a declarative test – i.e., material implication – in opposition to systems that “negate” by failing to find the sought object, which implement a form of the so-called close-world assumption;
8. *Constructive*, as memory retrieval always produces a novel object, in opposition to RAM memories or Hopfield's memories (Hopfield, 1982) that always reproduce a previously registered object;
9. *Indeterminate*, as the information is “overlapped” in the memory, hence has an entropy level.

A salient feature of the EAM is that the distributed property can be actual if the appropriate hardware is provided, and not only simulated, as is the case in computing paradigms using the TM. The distinction between local and distributed representations is reflected in the nature of the interpretation process: while the former is transparent and systematic – i.e., the meaning of the whole is a function of the meaning of the parts and their mode of grammatical composition, as in Frege's principle of compositionality – the latter is performed through a holistic act that presents the meaning to the interpreter's mind directly, and cannot be analyzed. The properties of the EAM system implement such functionality directly, as the retrieved object is presented to the interpreter's eye as a whole in a single un-analyzable act.

The indeterminate property seems to be necessary not only to store the huge number of objects that are registered in natural memories, but also to make them readily available when the cue “impacts” the indeterminate memory mass. This property can be better understood with an analogy to the interpretation of ambiguous images, such as

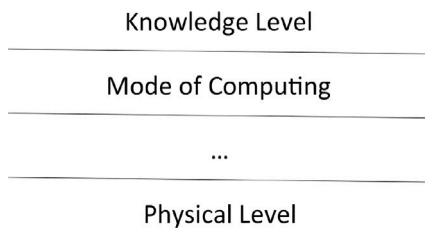


Fig. 4. The mode of computing.

Wittgenstein's famous duck-rabbit figure (Wittgenstein, 1953) or the visual images that emerge in diagrammatic proofs, such as the proof of the Theorem of Pythagoras (Pineda, 2007). The concrete shape of the diagram can be thought of as the cue to the memory, where the representations of the objects are stored within the indeterminate memory mass, and the retrieved determinate object is the concrete interpretation that appears in the mind's eye; however, the interpretations may be unstable and the memory retrieval operation may be performed recurrently, rendering new interpretations.

8. The mode of computing

The former discussion suggests that there is an open-ended set of intuitive notions of computing underlying different kinds of machines. This diversity can be seen in terms of the proposed hierarchy of system levels in which there is a distinctive level standing directly below the knowledge level and above all other levels, down to the physical one, that here is called *The Mode of Computing*, as illustrated in Fig. 4.

The mode of computing is the physical device or artifact, either artificial or a natural phenomenon, that provides the material support and carries on with the operations that map the representation standing for the argument of a function or a relation into the representation standing for its value. In the case of the TM and its equivalent formalisms the mode of computing is *Algorithmic Computing*. Formal languages and automata theory, the theory of the complexity of algorithms, the theory of computability and non-computable functions, such as the halting problem, were developed in relation to algorithmic computing but other modes may have different salient features.

There are modes that use symbolic manipulation but differ from algorithmic computing in principled aspects. For instance, Relational-Indeterminate Computing is stochastic and includes the computational entropy. There are also modes of computing that do not use symbolic manipulation and perform computations by other means such as analogical and quantum computing, and even sensors and transducers of diverse sorts. These modes do not rely on algorithms: electrical analogical computers transform inputs into outputs almost instantly and there is no sense in which these machines compute following a well-defined discrete procedure. Quantum computers can also be thought of as analogical and the use of the term "quantum algorithm" is informal for the same reason. Also, ANNs could be implemented with specialized hardware, using neuromorphic processors or other forms of unconventional computing, that do not use symbolic manipulation (Ziegler, 2020).

The level of the mode of computing can be partitioned into two or more sub-levels; for instance, a program written in a procedural language, such as Fortran, Pascal or C, stands in a sub-level immediately above the same program but compiled in assembler, which in turn stands above of corresponding binary program. Other examples are programming languages such as Prolog or Lisp whose interpreters are written in C, which in turn are expressed in assembler and binary code. These levels constitute different levels of abstraction for human interpretation but convey a trade-off from top to bottom between expressiveness and efficient computation. The AI distinction

between symbolic and sub-symbolic can be construed as sub-levels of the symbolic or algorithmic mode.

For analogical modes –such as quantum computing– there may be a symbolic interface through which inputs and outputs to the natural phenomenon that performs the computing process –the actual quantum engine– are presented to the knowledge level; similarly, standard analogical computers may have a symbolic interface through which the inputs and outputs are presented to the human interpreter.

However, there is no computation without the knowledge-level. The input and output of the computing engine at any mode of computing must be interpreted in relation to a predefined set of conventions, and the product of such interpretation is human knowledge. If there is computing there is interpretation but also if there is interpretation there is computing. A comprehensive notion of computing should involve both aspects of the phenomenon.

9. The mode of natural computing

Natural computing, in case it does exist, is carried out by natural brains of humans and animals with a developed enough nervous system. By analogy with the previous discussion there must be a system level directly below the knowledge level and above the physical brain structures that support the production of interpretations and experience that here is called *The Mode of Natural Computing*. Such system level should specify the code in which information is expressed and its interpretation process, and there must be specific brain structures that support such functionality. The relation between regions of the brain and mental functions is very likely *many-to-many* and the mode of natural computing may be supported by the functional organization networks, such as the orienting, alerting and executive attention networks, which play a strong role in the regulation of behavior, the control of affect and of the sensory input, that give rise to consciousness and voluntary behavior (Posner & Rothbart, 2007; Posner et al., 2019).

It is not necessary that all brain structures perform computations, which involve interpretations, and there may be structures that perform non-computational functions. In the world of artifacts invented by people there were control systems long before the TM was introduced and the notion of computing was available; similarly, in natural brains there may be structures performing control functions that precede the structures that are causal to consciousness and experience. Control structures are implemented by standard machinery that is there to be used but not to be interpreted, in opposition to computing machinery, whose only purpose is to map input into output representations at the mode of computing, that are interpreted and experienced by people.

The notion of the natural mode of computing is not necessary for non-representational and non-computational views of cognition, which deny that there is a computing engine that is causal and essential to the mind in the first place. For instance, the view that the brain is a dynamical system from which the mind emerges is not computational. Computers may be used as modeling tools, that aim to describe the functionality of the brain through computational models, but the models themselves are not causal nor essential to the modeled phenomena; in the same sense that computational models of the physical, chemical and biological world are not causal nor essential to the actual physical, chemical and biological phenomena. The same holds for computational models of psychological phenomena — unless such assumption is made explicitly, as in cognitive science and AI. Nature proceeds its course independently of whether or not it is modeled by a computer for the benefit of humans. A computational model requires representing the modeled object and make transformations upon such representation; but the representations are interpreted by people. In computing technology the mode of computing and the interpretation is performed by different entities, but in the case of the putative natural computing, the computing engine and the interpreter are the same individual. The whole idea that the mind is powered by a computing engine rests on the discovery and characterization of the mode of natural computing.

	Algorithmic Capacity	Memory Structure	Process Structure	Associative Memory	Entropy	Representational
Turing Machine	Very high	Local	Serial	No	No	No
Relational-Indeterminate-Computing	Low	Dual Local/Distributed	Serial/Parallel	Yes	Yes	No
Natural Computing	Very Low	Distributed	Parallel	Yes	Yes	Yes

Fig. 5. Artificial versus natural computing.

10. Artificial versus natural computing

Artificial computing based on the TM supposes the intensive use of algorithms for representing knowledge, both conceptual and of abilities, and for performing inferences. Algorithms can be seen as intentional descriptions of the functions they compute, whose purpose is simply to make explicit their extensions. Computing proceeds serially, supported by a local scanner that inspects the memory – which consists of a “passive recipient” – according to the structural ingredients of the TM. Computations are deterministic and are performed with great precision and at a high speed. Computations are normally performed forwards –from arguments to values– but not normally backwards –from values to arguments– because not all functions have an inverse.

However, whether the mind uses algorithms – assuming that there is indeed natural computing – is an open question. People are very limited to making extensive algorithmic calculations without the support of external aids and computing machinery. The information presented to living entities through perception is often extensional, and is plausible that a good deal of it is stored and processed in this form. Also, the amount of information and the structure of the brain suggest the use of highly distributed formats, with certain level of indeterminacy, that memory is fundamentally associative, and that computing is performed in parallel.

Natural computing, in addition, requires to perform inverse computations frequently; for instance, to recognize an object from its sensitive properties, or for diagnosing the causes of an event from its effects, often with uncertainty and indeterminacy. For this, it is plausible that natural computing uses formats that allow for establishing inverse associations, from values to arguments, very efficiently. ANNs have these properties to certain extent and reflect natural computing, but their simulation with TMs limits significantly their explanatory power. Table and Relational-Indeterminate Computing, on their part, have such properties directly and may model better some aspects of natural computing.

These considerations suggest to distinguish artificial or engineering computing, either in the form of the TM or RIC, versus natural computing in six salient dimensions: (1) algorithmic capacity; (2) memory structure; (3) whether the process is serial or parallel; (4) whether the mode supports associative memory; (5) whether the system is indeterminate and sustains a level of computational entropy; and (6) whether the computing agent sustains representations, and hence makes interpretations, experiences the world and can be conscious. The values of these dimensions for both artificial modes of computing and for natural computing are illustrated in Fig. 5. It is also plausible that the brain uses simple algorithms but also other modes of computing, and that a trade-off between modes is established.

11. Implications for church thesis

A very strong current of opinion in computer science, and also of popular opinion, holds that the TM is the most powerful computing engine that can ever exist (Copeland, 2019). The present discussion contests such position and it needs to be clarified the sense in which different modes of computing, including the mode of natural computing, differ from the standard theory of TM and computability theory.

Church thesis establishes that the set of functions that can be computed by the TM corresponds to the set of functions that can be computed intuitively by people, given enough time and paper. If a function does not have an algorithm it cannot be computed at all. If a non-computable function were computed by other means the thesis would be refuted. Of course, if all functions were computable the thesis would be empty. The thesis is based on the equivalence of all general enough formalisms or machines, such as the TM, the theory of recursive functions and the λ -Calculus, that compute exactly the same set of functions. The limits of computing depend on the impossibility to tell in advance whether a TM will halt for a given argument, the so-called Halting Problem. It is known that this problem cannot be solved by a TM. Hence, if a computing engine to such an effect were found – the so-called Halting-Machine – it would not be a TM and, as the TM is the most powerful machine by hypothesis, the Halting Machine cannot exist at all. If such machine were ever discovered Church Thesis would be refuted too (Boolos & Jeffrey, 1989).

A first observation is that Church Thesis presupposes that the object of computing is the mathematical function; hence it does not apply to modes of computing that are based in alternative notions. Nevertheless, the extent of the thesis is explored next in some detail to clarify its scope and limitations.

The set of all total and partial finite functions F – with finite domain and codomain – can be placed in a list. This set is different from the set of all total and partial functions, which is infinite and uncountable, as shown by Cantor’s anti-diagonal arguments — see Boolos & Jeffrey, 1989, cap. 2.

Let the sets $A = \{a_1, \dots, a_n\}$ and $V = \{v_1, \dots, v_m\}$ with cardinalities n and m be the domain and codomain of the set of functions $F_{n,m}$ such that $n, m \geq 1$. The arguments and values represent arbitrary individual entities, either concrete or abstract, although abstracting from their extension and quality, and the only requirement is that they can be placed on a list. Every function $f_k \in F_{n,m}$ can be expressed in a table with n columns and m rows, such that every column has at most one marked cell, where columns represent the arguments and rows the values, and the marked cells represent the functional relation.

Fig. 6 illustrates a table expressing a function whose domain and codomain are the sets $A = \{a_1, a_2, a_3, a_4\}$ and $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ with cardinalities $n = 4$ and $m = 7$, respectively. The function has an index f_k as shown in the top row, which in this case is 1247. This is

f_k	1	2	4	7
v_7				X
v_6				
v_5				
v_4			X	
v_3				
v_2		X		
v_1	X			
	a_1	a_2	a_3	a_4

Fig. 6. Diagrammatic representation of finite discrete functions.

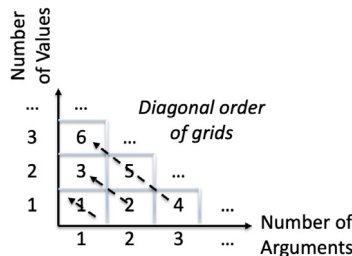


Fig. 7. Order of tables for finite discrete functions.

a number of n digits in base $m + 1$ which is formed with the index j of the value v_j for the argument a_i in case the function is defined for such argument and with 0 otherwise, for all the arguments. In particular, the index formed by a string of n 0s corresponds to the empty function that assigns no value to all of its arguments. The names of the arguments and values, as well as the index, are metadata, and are not considered part of the table, which is the object of computing proper.

The figure illustrates that the value of f_k ranges from “0000” to “7777” in base 8 — i.e., from 0 to 4095 in base 10. In general, the number of a function is $N_k = (f_k)_{10} + 1$, where (f_k) is a number in base $m + 1$ and $(f_k)_{10}$ denotes f_k in base 10. In the example, $N_k = (f_k)_{10} + 1 = 1247_{10} + 1 = 680$.

The number of total and partial functions that can be formed with n arguments and m values is $(m + 1)^n$. This can be seen directly by noticing that the number of functions is simply the number of combinations that can be formed by selecting one out of the $(m + 1)$ values at a time for the n arguments. The number of functions in Fig. 6 is $(7 + 1)^4 = 4096$.

Next, we order the set of functions $F_{n,m}$ that can be included in a table of size $n \times m$ for $m, n \geq 1$. All pairs (n, m) can be placed in a list; consequently, all sets $F_{n,m}$ can be placed in a list. Let t be the index of a set $F_{n,m}$. Fig. 7 shows a diagonal order of all tables of size $n \times m$ for $m, n \geq 1$, where each table is labeled with its index t . In particular, $t = 1$ is the index of the functions $F_{1,1}$ that have one argument and one value at the most. This set has two functions: f_0 that assigns no value to its argument, and f_1 that does so. In order to assign the index t to a table as a function of n and m :

1. It can be seen by direct inspection of Fig. 7 that:
 - Each diagonal j has j tables;
 - The index t of each table is increased over the diagonal from bottom to top and from right to left;
 - The table $n \times m$ lays on the diagonal $diag(n, m) = n + m - 1$.
2. It can be seen by diagrammatic reasoning that:
 - The largest index in diagonal j is defined recursively as follows: $max(0) = 0$ and $max(j) = max(j - 1) + j$;
 - $t = max(diag(n, m) - 1) + m$.

The number of a function N_f in table t is the sum of all the functions in tables 1 to $t - 1$ added on N_k — i.e., the number of the function in the table t . Hence, for finding out the number of a function in the table with cardinality (n, m) it is required to find n and m as a function of the index t . This is as follows:

1. Let $diag(i)$ be the diagonal of table $i = (n(i), m(i))$;
2. It can be seen through diagrammatic reasoning that:
 - The function $diag(i)$ is computed through the following minimization: $diag(0) = 0$ and $diag(i) = j$ such that $j = 0$; increment j until $max(j - 1) < i \leq max(j)$;
 - $n(i) = max(diag(i)) - i + 1$;
 - $m(i) = i - max(diag(i) - 1)$.

The absolute number of the function N_f is:

Definition 1. Number of function for $n, m \geq 1$

$$N_f = N_k \text{ if } t = 1;$$

$$N_f = \sum_{i=1}^{t-1} (m(i) + 1)^{n(i)} + N_k \text{ if } t > 1$$

Definition 1 is recursive and can be computed by a TM. It establishes that all total and partial functions with finite domain and range are enumerable, regardless of the size of n and m . So, given the size of the table and a function in such table, the absolute number of such function is provided. Conversely, all functions can be generated out of their indices. The procedure is simply iterate over the index i from $i = 1$ to t ; compute $n(i)$ and $m(i)$; and compute all indices of length $n(i)$ in base $m(i) + 1$ — i.e., from $0...0$ to $m...m$. This renders all tables or the extensional representation of all finite functions.

This enumeration is redundant. In particular all set of function F_{n,m_j} includes all functions in the set F_{n,m_i} if $j > i$. Consequently, all functions have a enumerable set of indices, but the enumeration is complete, and picks up all finite functions with cardinalities $n, m \geq 1$.

All finite functions in the table format can be computed by one single extensional table-computing non-recursive procedure, which consists of selecting the column corresponding to the argument, finding out the marked cell in such column, and selecting the value associated to the row. Table-computing can be thought of as a TM that uses the table as its tape, and the table-computing algorithm is recursive, although trivially. Alternatively, all finite functions can be computed extensionally by a TM by placing all the pairs of arguments and values on the tape, in the order given by Definition 1, and scanning the tape until the given argument is found and its associated value is returned. It follows that all finite functions are recursive, but trivially too. Hence, there are none non-computable finite functions, and Church Thesis holds but is empty for this set — i.e., the set of recursive finite functions and the set all finite functions is the same. Furthermore, such extensional machines provide the values of all the arguments of all finite functions by direct inspection and are immune to the halting problem.

However, algorithmic computing is generally understood as computing functions through effective procedures. Standard programs running in ordinary digital computers, that have a finite-size processing and memory registers (e.g., 32 or 64 bits), and finite memory, are the paradigmatic case of such non-trivial algorithms that compute finite functions. Computable functions have an underlying structure or regularity that is capitalized in the construction of their algorithms, but the table format suggests that there is a large number of finite functions that have none or very little structure, that may not have a non-trivial algorithm and would not be computable. If at least one such function were found, it could nevertheless be computed through table-computing, and Church Thesis would not hold in this non-trivial sense. The enumeration also shows that the halting problem for non-trivial algorithms computing finite functions could be solved if the number of a function can be determined out of the specification of the algorithm

and vice versa, although it is very unlikely that such correspondence can ever be established.

For these reasons the table and the algorithmic are different modes of computing. Table-computing can be employed only when the function is provided extensionally, which is plausible in natural settings and concrete problem-solving, but the extension of most mathematical functions is known intensionally and non-trivial algorithms are needed to compute them. Humans are very limited for computing algorithms and that is why computers were invented in the first place.

Now we turn to the case in which the domain and range of the functions are not finite. The TM is an abstract machine defined to have an enumerable infinite set of inputs. Hence, in addition to the finite functions enumerated by [Definition 1](#), there are uncountable many functions with infinite domains and codomains. In this latter case, there may be a very large number of functions that lack an algorithm and are not recursive, which of course would not have a table representation, and could not be computed through other means. These are the non-computable functions. If a single functions of this kind could be computed by other means, Church Thesis would be refuted, in the intended relevant sense.

Computational models of physical systems are stated through equations with real domains and codomains, and their computation requires algorithms with infinite many inputs. However, real numbers can be approximated accurately enough through their discrete representation in digital memory registers, and physical systems can be modeled through digital computers with the only limitation of the available time and memory resources. It may also be the case that the solution of a model of a physical system is a non-computable function, in which case it cannot be computed neither by people nor by machines. If a model of a physical system whose solution is a non-computable function were solved by other means, Church Thesis would be refuted too.

The discussion up to this point shows the sense in which the TM is the most powerful computing machine, which can be summarized as stating that there is an effective procedure or algorithm for each computable function, with the only limitation established by the halting problem; and that there are functions that cannot be computed in an absolute sense. Functional evaluation is a fully determined process by necessity. Hence, the indeterminacy of stochastic processes and non-deterministic automata is dissolved in a theory of functions.

Next, we address the sense in which the TM can be the most “powerful machine”. This is a complicated notion in several respects, starting with the fact that different machines or formalisms can be more powerful or weaker than others in particular aspects, even though all of them may compute the full set of computable functions. For instance the TM is better than the von Neumann architecture to study the general properties of computing machines, but the von Neumann architecture is better to implement practical computations, despite that both of these formalisms compute the same set of functions. Also, all models of computing assume particular trade-offs; for instance, algorithmic computing assumes a fundamental trade-off between the expressive power of representations and their tractability, or the possibility of perform a computation with finite time and memory resources. This trade-off is explicit in Chomsky’s hierarchy of formal languages that includes, from bottom to top, the regular languages, the context free languages, the context sensitive languages and the languages without restrictions of any sort ([Hopcroft et al., 2006](#)). Regular languages have a concrete character and can be computed very efficiently but can express only very limited abstractions while the languages in the other extreme can express very deep abstractions but the computational cost can be very high and cannot be afforded always. The algorithmic mode also assumes *The Knowledge Representation Trade-Off* stating that concrete representations can be computed efficiently but have expressiveness restrictions and that abstract representations can very expressive but cannot be computed effectively ([Levesque, 1988](#); [Levesque & Brachman, 1985](#)). For instance, the consequences of the knowledge expressed in propositional logic can be found very easily but if the full expressive

power of predicate logic is required the computational cost may be too high and computations may be unfeasible. Another example is ANNs that can simulate distributed representations very efficiently but cannot express syntactic structure and cannot store declarative information.

Similarly, there are symbolic modes more powerful than a TM in particular aspects. For instance, while the TM is directed to make complex calculations but is not decidable due to the halting problem, computing with tables is oriented to make direct associations and computations always terminate. Also, while computing a function and its inverse, whenever it exists, requires different algorithms, computing with tables provides the inverse function or a relation directly by inspection using the same algorithm for all inverse functions and relations. Another example is the opposition between the TM and Relational-Indeterminate Computing: while the former is deterministic the latter is indeterminate, stochastic and entropic. From this perspective the TM is the machine with zero entropy, but there are computing engines that have an entropy larger than zero. On its part, analogical computers are very efficient and computations are performed through the physical phenomena almost instantly but compute a specific set of functions, are indeterminate and have no memory capabilities.

More generally, every mode of computing assumes some fundamental trade-offs that define its explanatory capability and its potential applications. There may be properties of different modes that can be compared directly, such as the speed or memory capacity of digital versus quantum computers, but comparing different modes of computing in general is a category mistake.

The notion of effective procedure or Turing’s computability has been extrapolated and Church-Turing Thesis has been interpreted as stating that the TM can simulate all possible mechanisms, in particular in the forms that Copeland ([Copeland, 2019](#)) calls the Maximality Thesis, sustaining that all functions that can be generated or performed by machines can be computed by a TM; the Simulation Thesis, holding that Turing’s results imply that the brain – and any physical or biological system – can be simulated by a TM; and, more radically, the Physical Church-Turing Thesis stating that any physical computing device and physical thought experiment that is conceived or designed by the present and any future civilization can be simulated by a TM. These theses taken together and other similar propositions are informally called the strong version of the Church-Turing Thesis stating that the TM is the most powerful computing machine that can ever exist in any possible sense. These views seem to presuppose that the universe or nature is governed by a set of causal fundamental rules, that determine all possible events, at all system levels, that can be characterized as functions, in the mathematical sense, and that the TM is the deterministic device that can compute such functions. This is a reissue of Laplace’s determinism, that is hard to die.

12. The potential productivity of decisions

“Decisions” performed by TMs are predetermined. In Laplace’s deterministic view, if an “agent” makes a decision with the intention of reaching a state of the world that is not within the predetermined path, such “decision” is wishful thinking. Genuine decision making underlies that the world is not fully determinate; that the world states have a number of possible next states, that can be reached with some associated transition probability; and that decisions made by agents may lean the world to move to a desired state, which may not be the most probable one.

The space of indeterminacy of an agent can be measured with Shannon’s entropy. In the Mathematical Theory of Communication ([Shannon, 1948](#)) the information source – an entity emitting messages – is modeled using Markov processes, and the entropy is defined as a property of the source: the overall uncertainty or indeterminacy of the emitted messages. Shannon reasoned that the lower the uncertainty of the message the shorter the size of the code needed to transmit it and, conversely, the larger the uncertainty the longer the code. To achieve

this end, he proposed to code messages in terms of their probability of emission over the distribution of all possible messages, so the length of message i with probability $p(i)$ should be coded with $-\log_2(p(i))$ bits. For instance, if the probability of emitting a message is $1/4$ its length should be $-\log_2(1/4) = 2$ bits. In particular if a message has probability of 1, the size of its code is 0 bits — i.e., there is no need to send such message. The entropy s is the average length of the coded messages produced by the source, and is calculated with an expected value formula: $s = -\sum_{i=1}^n p(i) \log_2(p(i))$. If all the messages have the same probability, the entropy has the maximal possible value, which corresponds to the largest uncertainty of the source. Conversely, if the entropy is zero, the world — from the point of view of the source — is fully determinate and there is no need of communicating. The entropy is a fundamental parameter for the design of communication systems, and as communicating demands energy, it is also a measure of the value of communicating for the information source.

As discussed in Section 2, the ability to communicate is a sign that the agent represents interpretations and has a mind already. Communication represents mental content, and the entropy of the agent reflects such content. The lower the agent's entropy the larger its determination; the lower his or her need to communicate; and the lower his or her agency. Hence, animals with a developed enough neural system that are able to communicate are agents — but animals and plants without such endowment lack agency. Conversely, entities that do not communicate, such as inert masses —e.g., stars, planets, oceans and mountains— but also material forces —either mechanical, electrical or magnetic— or electromagnetic waves, are not agents. Artifacts made by people, such as tools and machinery, are neither agents — machines interchange signals but do not represent interpretations, hence do not communicate.

Communication allows agents to share knowledge, beliefs, intentions, feelings and experience, and enhances their ability to making decisions and performing actions to the effect that their quality of life and their chances of survival are improved, over the states that would be reached if such decisions and actions had not been made, and the world had followed its most probable path. I propose the conjecture that there is a relation between the space of indeterminacy that the agent enjoys for genuine decision making — the agent may be an individual, an animal species, or even a society or an institution — and the potential value of such decisions, which I call *The Potential Productivity of Decisions* τ (Pineda, 2021). This notion is illustrated in Fig. 8. The horizontal axis corresponds to the agent's entropy, and the vertical one to the overall value of the decisions made by the agent at a given value of the entropy. The notion presupposes that the agent makes decisions and performs actions to satisfy his or her intentions — which may or may not be satisfied in the state of the world that is reached — and τ reflects the overall effect of such decisions and actions. The indeterminacy can be assessed by measuring the average length of the messages emitted by the agent over a period of time, or the energy that he or she employs in communicating, and τ may be assessed empirically in relation to an appropriate metric. The proposal is that agents have an overall τ or a particular τ over an aspect of their behavior and/or window of time.

The shape of the curve is a bell, which in a first approximation can be modeled as a normal distribution; the zero entropy point and the main entropy intervals are described as follows:

- Zero entropy: The actions of the entity are fully determinate, so things happen “automatically”; hence, the entity cannot benefit from its “decisions” and the value of τ is zero. This is the case of tools and machines, including automaton and the TM. For instance, of robots in automated factories that have all of their actions programmed.
- Low entropy: The world is highly determined and the agent has very limited decision making capability. This is the case, for instance, of human workers in a highly automated factory

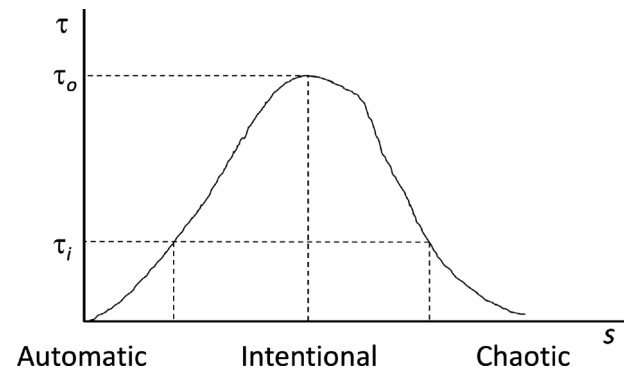


Fig. 8. Potential productivity of decisions τ .

who do not make decisions or if they do, such decisions are of no consequence; of the behavior of animals or humans when they are highly conditioned by the external world or conditioned internally — as in obsessive or addictive behavior — and of highly automated perceptual and motor behavior, such as the abilities deployed by agents.

- Moderate entropy: The agent has a space to making productive decisions that have a satisfactory value. This is the space of natural language communication. There may be an optimal point of indeterminacy in which τ is maximal, where cooperative behavior and creativity may occur. This is the region where the free will is exercised, so the decisions and the actions of the agent are directed to achieve his or her intentions.
- High or too high entropy: the indeterminacy and uncertainty is too high due to the external physical or social environment — for instance, when there is a pandemic, an earthquake or a hurricane — or to the state of the agent's mind; the agent may make a large number of decisions but these may not be feasible to achieve or may not be directed; hence, τ decreases and in the limit may be very low. Communication in this condition has a pretty poor ratio of cost to benefit, and cannot be sustained in the long term.

The potential productivity of decisions helps to clarify Searle's position on the kind of the entities that can possibly be conscious and experience the world. In the refutation of the so-called “System Reply” of the Chinese room story —the counter argument that consciousness is a property of the room as a whole— Searle argues that if such refutation were sound it would also follow that every device, natural or artificial, that transforms inputs into outputs, such as biological organs —e.g., stomachs, livers and hearts— and also artifacts —e.g., thermostats, telephones, adding machines, electric switches, etc.— should be granted consciousness too (Searle, 1980). From the present perspective, organs are placed very near the origin in the entropy dimension —they can be thought of as having a natural control system, as in cybernetics— and artifacts are placed at the origin exactly, as any other piece of machinery. Hence, organs and artifacts have very little or none agency, and cannot be intentional. The predictive phenomena of the physical world stands at the origin too, where the entities have no agency and follow the natural forces passively, in opposition of agents, that can act upon the world with the purpose of satisfying their intentions.

13. Computing, consciousness and experience

The computational metaphor of mind rests on the discovery and characterization of the mode of natural computing. Without such construct computing has no explanatory power in cognition and related disciplines. Artificial Intelligence and cognition are traditionally thought of within a form of functionalism that is realized by the TM, giving rise to the so-called functional computationalism (Puccinini, 2010),

but this view is limited because the TM cannot make interpretations, and consciousness is not explained. Mental states of humans and no human animals with a developed enough nervous system have an “intrinsic intentionality” – e.g., (Egan, 2014) – but the states of the TM do not have such property, neither in the structural or functional ingredients of its abstract specification nor in its physical realizations. Computing states are material configurations, such as the gears and levers of Babbage’s analytical engine, that needed to be interpreted by people. The assumption that embedding a physical realization of a TM – or of another artificial mode of computing – in a physical body that interacts with the environment, provides intentionality to the machine states, due to such sole fact, does not follow, as has been extensively argued. Closing the gap between computing states and mental or intentional states seems to be an utopian enterprise, even for the original proponents of functionalism (Shagrir, 2005). Nevertheless, supporters of such doctrine claim that most criticism to functional computationalism are miss-understandings of one or another sort, including Searle’s arguments, due to his position that consciousness is a subjective phenomenon, that is realized in biological brains, although it is not known why (Milkowski, 2018). The confusion comes from the identification of artificial and natural computing, which seems to be taken as the same by the supporters of functional computationalism.

The ontological status of mental objects at the knowledge level is problematic, but if these objects are postulated at all, a parsimonious criteria about their nature is considering that they reflect the properties or processes of their corresponding objects at the level of the mode of computing, for all modes that are used in natural computing. This view is called here *the representational view*. The algorithmic mode uses symbolic manipulation whose interpretations at the knowledge level can be thought of as propositions. Fodor’s mentalese corresponds to this view. Other symbolic modes, such as Relational-Indeterminate Computing, are characterized by the use of the space as the medium and the indeterminacy of its structures. Representations at the knowledge level would require no further interpretation; otherwise, interpretation would involve an infinite regress. In the representational view it has to be held as well that human interpreters are conscious by the mere fact of having such representations in mind, at least when they are the focus of attention, including the qualitative, experiential or phenomenological aspect of consciousness.

However, it is not necessary to hold that there are mental objects – in an ontological sense – at the knowledge level. This can be elicited as claiming either that the so-called knowledge level consists of a continuous interpretation process of the objects at the level of the mode of natural computing, which we call *the interpretative view*. There is also the view that the knowledge level does not exist at all, neither in artificial nor in natural computing, which is called here *the non-interpretative view*.

The interpretative view allows theorists to hold that in natural computing the knowledge level and the mode of computing fuse in a single level that is causal and essential to consciousness and experience. If the mode of natural computing uses a declarative format, such structure would hold the actual code in which information is stored in the brain, and the operations on it are actual interpretations that allow the agent to be aware, understand and experience the world. Such representations would be stable and could be consciously inspected in distinctive brain states. If there are not declarative structures at the mode of natural computing, the interpretation is performed upon the input and output information of continuous processes. The interpretation is also causal to experience and consciousness, but the states are too proximal between each other, as the individual pictures of a movie, that produce a continuous experience. The interpretative view is perhaps more parsimonious than the representational one, which can be dispensed with altogether, and allows to hold that there are declarative and procedural natural modes of computing, that can co-exist coherently.

Lastly, in the *the non-interpretative view* the knowledge level is absent altogether in both artificial and natural computing; there are not interpretations, neither meanings, nor consciousness nor experience. In this perspective, computing is an objective property of mechanisms which can be ascribed to any kind of artifact and even to organs, as pointed out by Searle. Machinery would carry on computing even if the whole of the human kind – as well as all no-human animals endowed with natural computing – were wiped out from the face of the earth. Furthermore, arbitrary physical, chemical or biological phenomena could also be considered computing machines, including the universe as a whole. But then, the whole notion of computing becomes rather empty.

However, there is no computation without representation and interpretation. Computing is not an objective property of mechanisms, but a relational phenomenon, involving a mode of computing and a conscious interpreter. The interpretation is causal to experience, to the ascription of meaning and to consciousness. Experience or awareness would be how computing with a particular mode feels, for all modes used in natural computing. Computing, interpreting and consciousness are three aspects of the same phenomenon; or consciousness and experience are the manifestations of computing/interpreting. Interpreting is what distinguishes computing machines from standard machinery. In the case of artificial computing the process is split into two different entities: the one that supports a mode of computing and the one who makes the interpretation. In the case of natural computing the entity that holds the mode of computing and the one that makes the interpretation is the same individual.

The present proposal does not advocate rejecting that the mind is a computational process, but holds that finding the mode or natural computing is required. This is an open question for science altogether and there is not or does not seem to be any plausible answer in sight. But for the time being, in the absence of the characterization of the mode or modes of natural computing, computing has no explanatory power in cognition and related disciplines, and can be removed from current talk entirely, such as other theoretical constructs that were held as fundamental for some time but lacked a real explanatory power and were dropped on the face of the evidence, such as the ether. If it turns out that the mode of natural computing does not exist, the mind would not be a computational process, but something else; conversely, discovering it would come with finding the property that makes interpreters ascribe meanings to representations and have subjective experience; and such a discovery would equate to solving the hard problem of consciousness.

CRediT authorship contribution statement

Luis A. Pineda: Conceptualization, Formal analysis, Investigation, Methodology, Writing – original draft.

Declaration of competing interest

I declare that I have no conflict of interest in relation to the paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

I thank Rafael Morales and Gibrán Fuentes for intense discussions and long-time collaboration; Carlos Velarde for comments and suggestions to Section 11; Enrique Sucar and Christian Lemaitre for discussions on Section 12; and Noé Hernandez and Nydia Pineda De Ávila for comments and proofreading. I also acknowledge encouraging conversations on previous drafts with Mike Posner, Ed Hovy, Thomas Eiter and Tom Froese. I have also enjoyed fruitful interactions with colleagues at

Academia Mexicana de Computación and Academia Mexicana de Ciencias. I have benefited from the long-term support of the National System of Research of the Mexican National Council of Humanities, Science and Technology (formerly CONACyT), and of UNAM's DGAPA-PAPIIT. Finally, I acknowledge the deep and productive feedback from the reviewers.

References

- Anderson, M. L. (2003). Embodied cognition: A field guide. *Artificial Intelligence*, 149, 91–130.
- Baddeley, A. D. (1992). Working memory. *Science*, 255, 556–559. <http://dx.doi.org/10.1126/science.1736359>.
- Besold, T. R., Garcez, A. d., Bader, S., Bowman, H., Domingos, P., Hitzler, P., Kuehnberger, K.-U., Lamb, L. C., Lowd, D., Lima, P. M. V., de Penning, L., Pinkas, G., Poon, H., & Zaverucha, G. (2017). Neural-symbolic learning and reasoning: A survey and interpretation. <http://dx.doi.org/10.48550/ARXIV.1711.03902>, arXiv URL <https://arxiv.org/abs/1711.03902>.
- Boolos, G. S., & Jeffrey, R. C. (1989). *Computability and logic* (3rd Ed.). Cambridge: Cambridge University Press.
- Brooks, R. (1991a). Intelligence without representation. *Artificial Intelligence*, 47, 139–159.
- Brooks, R. A. (1991b). Intelligence without reason. In *Proceedings of the 12th international joint conference on artificial intelligence - volume 1* (pp. 569–595). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
- Chalmers, D. J. (1990). Syntactic transformations on distributed representations. *Connection Science*, 2(1–2), 53–62. <http://dx.doi.org/10.1080/09540099008915662>.
- Chemero, A. (2013). Radical embodied cognitive science. *Review of General Psychology*, 17(2), 145–150. <http://dx.doi.org/10.1037/a0032923>.
- Chomsky, N. (1957). *Syntactic structures*. The Hague/Paris: Moutons.
- Chomsky, N. (1959). A review of B. F. Skinner's verbal behavior. *Language*, 35(1), 26–58.
- Copeland, B. J. (2019). The Church-Turing thesis. In E. N. Zalta (Ed.), *The stanford encyclopedia of philosophy* (Spring 2019). Metaphysics Research Lab, Stanford University.
- Egan, F. (2014). How to think about mental content. *Philosophical Studies*, 170(1), 115–135. <http://dx.doi.org/10.1007/s11098-013-0172-0>.
- Fodor, J. A. (1975). *The language of thought*. Harvard University Press.
- Fodor, J. A., & Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1–2), 3–71.
- Froese, T., & Ziemke, T. (2009). Enactive artificial intelligence: Investigating the systemic organization of life and mind. *Artificial Intelligence*, 173, 466–500.
- Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., Badia, A. P., Hermann, K. M., Zwols, Y., Ostrovski, G., Cain, A., King, H., Summerfield, C., Blunsom, P., Kavukcuoglu, K., & Hassabis, D. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626), 471–476. <http://dx.doi.org/10.1038/nature20101>.
- Hassabis, D., Kumaran, D., & Maguire, E. A. (2007). Using imagination to understand the neural basis of episodic memory. *The Journal of Neuroscience*, 27(52), 14365–14374. <http://dx.doi.org/10.1523/JNEUROSCI.4549-07.2007>, arXiv:<https://www.jneurosci.org/content/27/52/14365.full.pdf>.
- Hassabis, D., & Maguire, E. A. (2007). Deconstructing episodic memory with construction. *Trends in Cognitive Sciences*, 11(7), 299–306. <http://dx.doi.org/10.1016/j.tics.2007.05.001>, URL <https://www.sciencedirect.com/science/article/pii/S1364661307001258>.
- Hinton, G. E., McClelland, J. L., & Rumelhart, D. E. (1986). Distributed representations (chapter 3). In D. E. Rumelhart, & J. L. McClelland (Eds.), *Parallel distributed processing, explorations in the microstructure of cognition, vol.1: foundations*. Cambridge, Mass: The MIT Press.
- Hopcroft, J., Motwani, R., & Ullman, J. D. (2006). *Introduction to automata theory, languages, and computation* (3rd Ed.). Boston, MA, USA: Addison-Wesley Longman Publishing Co. Inc.
- Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the USA*, 79(8), 2554–2558.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(11), 436–444. <http://dx.doi.org/10.1038/nature14539>.
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.
- Levesque, H. L. (1988). Logic and the complexity of reasoning. *Journal of Physiology (Cambridge, Eng)*, 17, 355–389.
- Levesque, H. L., & Brachman, R. (1985). A fundamental trade-off in knowledge representation and reasoning. In R. Brachman, & H. Levesque (Eds.), *Readings in knowledge representation* (pp. 41–70). Los Altos, CA: Morgan and Kaufmann.
- Marr, D. (1982). *Vision: a computational investigation into the human representation and processing of visual information*. New York, NY, USA: Henry Holt and Co. Inc.
- Milkowski, M. (2018). Objections to computationalism: A survey. *Roczniki Filozoficzne*, 66, 57–75. <http://dx.doi.org/10.18290/rf.2018.66.3-3>.
- Morales, R., Hernández, N., Cruz, R., Cruz, V. D., & Pineda, L. A. (2022). Entropic associative memory for manuscript symbols. *PLoS ONE*, 17(8), Article e0272386, URL <http://dx.doi.org/10.1371/journal.pone.0272386>.
- Newell, A. (1982). The knowledge level. *Artificial Intelligence*, 18, 87–127.
- Newell, A., & Simon, H. (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3), 113–126.
- Pineda, L. A. (2007). Conservation principles and action schemes in the synthesis of geometric concepts. *Artificial Intelligence*, 171, 197–238.
- Pineda, L. A. (2021). *Racionalidad computacional*. Ciudad de México: Academia Mexicana de Computación, A. C..
- Pineda, L. A., Fuentes, G., & Morales, R. (2021). An entropic associative memory. *Scientific Reports*, 11, 6948, URL <https://www.nature.com/articles/s41598-021-86270-7>.
- Pineda, L. A., & Morales, R. (2022). Weighted entropic associative memory and phonetic information. *Scientific Reports*, 12, 16703, URL <https://www.nature.com/articles/s41598-022-20798-0>.
- Pineda, L. A., & Morales, R. (2023). Imagery in the entropic associative memory. *Scientific Reports*, 13, 9553, URL <https://www.nature.com/articles/s41598-023-36761-6>.
- Posner, M., & Rothbart, M. K. (2007). Research on attention networks as a model for the integration of psychological science. *Annual Review of Psychology*, 58, 1–23.
- Posner, M., Rothbart, M. K., & Ghassemzadeh, H. (2019). Restoring attention networks. *Yale Journal of Biology and Medicine*, 92, 139–143.
- Puccinini, G. (2010). The mind as neural software? Understanding functionalism, computationalism, and computational functionalism. *Philosophy and Phenomenological Research*, LXXXI(2), 269–311.
- Rojas, R. (2021). The computer programs of charles babbage. *IEEE Annals of the History of Computing*, 43(1), 6–18. <http://dx.doi.org/10.1109/MAHC.2020.3045717>.
- Rueckl, J. G. (1991). Connectionism and the notion of levels. In T. Horgan, & J. Tienson (Eds.), *Connectionism and the philosophy of mind. studies in cognitive systems. Vol. 9*. Springer, Dordrecht.
- Rumelhart, D. E., McClelland, J. L., & the P. D. F. Research Group (1986). *Parallel distributed processing, explorations in the microstructure of cognition, vol.1: foundations*. Cambridge, Mass: The MIT Press.
- Searle, J. R. (1980). Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(3), 417–457.
- Shagrir, O. (2005). The rise and fall of computational functionalism. In Y. Ben-Menahem (Ed.), *Hilary Putnam (Contemporary Philosophy in Focus)*. Cambridge University Press.
- Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3), 379–423. <http://dx.doi.org/10.1002/j.1538-7305.1948.tb01338.x>.
- Shields, C. (2016). Aristotle's psychology. In E. N. Zalta (Ed.), *The stanford encyclopedia of philosophy* (Winter 2016). Metaphysics Research Lab, Stanford University.
- Siegelmann, H. T., & Sontag, E. D. (1995). On the computational power of neural nets. *Journal of Computer and System Sciences*, 50(1), 132–150. <http://dx.doi.org/10.1006/jcss.1995.1013>.
- Smith, B. C. (1985). Prologue to “reflection and semantics in a procedural language”. In R. Brachman, & H. Levesque (Eds.), *Readings in knowledge representation* (pp. 31–40). Los Altos, CA: Morgan and Kaufmann.
- Sun, G. Z., Chen, H. H., Lee, Y. C., & Giles, C. L. (1992). Turing equivalence of neural networks with second order connection weights. In Anon (Ed.), *Proceedings. IJCNN - international joint conference on neural networks* (pp. 357–362). Publ. by IEEE.
- Tulving, E. (2002). Episodic memory: From mind to brain. *Annual Review of Psychology*, 53, 1–25.
- Turing, A. M. (1936). On computable numbers, with an application to the entscheidungs problem. *Proceedings of the London Mathematical Society*, 42, 230–265.
- Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59, 433–460.
- Wittgenstein, L. (1953). *Philosophical investigations*. Basil Blackwell.
- Ziegler, M. (2020). Novel hardware and concepts for unconventional computing. *Scientific Reports*, 10, 11843, URL <http://dx.doi.org/10.1038/s41598-020-68834-1>.