Orbis Idearum, Volume 10, Issue 2 (2022), Pages 41-63
ISSN: 2353-3900

THE CHILD PROGRAM AND THE HISTORY OF AI

Fabio Grigenti
University of Padua
fabio.grigenti@unipd.it

ABSTRACT

In this contribution I try to illustrate some of the most significant moments in the history of AI (Artificial Intelligence), imagining that they have been a series of subsequent stages in the development of the so called "Child Program." With this name Alan Turing – in *Computing machinery and intelligence* (1950) – described for the first time the idea of an algorithm "capable of being educated." After discussing Turing's prophecy, I will present some milestones of research in AI considering that each is a piece of a "learning scheme" of extended intelligence, i.e. an intelligence ascribable not only to humans beings, but also to any subject able to support thinking functions. I propose that from the elements of this scheme it is possible to draw useful suggestions for human education in general. In the final part of the contribution, I will collect the elements into a synoptic vision, which I will call "the educational protocol of the extended intelligence." This will define the minimum requirements that a training course should possess in order to conduct a "child program" at the "adult program" stage.

Keywords: Programming, language, common sense, learning

1. THE CHILD PROGRAM

The possibilities that (AI) Artificial Intelligence can offer to the field of education are not a recent theme, but something foreseen right from its historical origin. In a famous passage of *Computing machinery and intelligence* (1950) Alan Turing wrote:

> In the process of trying to imitate an adult human mind we are bound to think a good deal about the process which has brought it to the state that it is in. We may notice three components. (a) The initial state of the mind, say

Received 9 December 2022, Revised 29 December 2022,
Accepted 3 January 2022, Available online 16 January 2023

> at birth, (b) The education to which it has been subjected, (c) Other experience, not to be described as education, to which it has been subjected. Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's? If this were then subjected to an appropriate course of education one would obtain the adult brain. Presumably the child brain is something like a notebook as one buys it from the stationer's. Rather little mechanism, and lots of blank sheets. (Mechanism and writing are from our point of view almost synonymous.). Our hope is that there is so little mechanism in the child brain that something like it can be easily programmed. The amount of work in the education we can assume, as a first approximation, to be much the same as for the human child.[1]

This passage, which today is considered prophetic,[2] had no resonance at the time and was perhaps perceived as a provocation. Turing imagines that, if a machine can be made capable of simulating the symbolic behavior of a mind at a certain stage of its development, the same machine will also be programmable to simulate the development from an child-uneducated stage to an adult-trained one – a process which largely corresponds to what our culture has always considered as "education." Turing imagines a child's mind to be like a notebook filled with blank sheets, to which a writing mechanism is attached. Evidently, in the adult stage, the mind will be full of notes, and there will be less and less space to write something new. What Turing proposes then is not to write a program that simulates an exercise book already full of information, but a program that replicates the way in which a child begins to write something on his own mind sheet. Now, the questions are the following: what kind of instructions should this program have? Does it act without any previous knowledge or does it start from particular presuppositions? How does it select and capture data from educational training and learning environment?

As we will see below, some breakthroughs in the field of knowledge artificialization look exactly like attempts to answer all or some of these questions. The current and almost unique interest in machine learning seems to confirm this working hypothesis. Even if we are unable to predict future developments, the panorama of research in AI is objectively dominated by the problem of "educating machines to learn by themselves" rather than by the concern to make computational processes even more rigorous. This brings us to the second thesis that I intend to defend. It can be formulated as follows:

---

[1] A. M. Turing, "Computing machinery and intelligence," *Mind*, 59 (1936): 433-460.

[2] As we shall see, this issue is central for the current development of machine learning. On Turing's *Child Program* see S. Muggleton, "Alan Turing and the development of AI," *AI Communication*, 27, 1 (2014): 3-10.

if the story of AI embodies the effort to fully develop the "Child program," this same story can also be read as the unfolding – at each stage – of a process of an ever better understanding of what education is in a broader perspective than the human one. However, precisely for this reason, it is also valid for humans. I will call this point of view "extended intelligence educational protocol." In the conclusions, I will give a brief description of it.

2. THE BIRTH OF A NEW RESEARCH PARADIGM

At the beginning of 1956, John McCarthy – helped by Claude Shannon and Nathaniel Rochester – managed to bring together U.S. most relevant researchers interested in automata theory, neural nets, and the study of intelligence. Finally it was decided – and this happened in the summer of 1956 – that the workshop would take place for two months at Dartmouth College in Hanover, New Hampshire. The *abstract* of the invitation addressed to researchers reported the following words:

> The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.[3]

Among the ten participants in the event, in addition to the already mentioned John Mc Carthy, Shannon and Rochester – we have also to remember Trenchard More from Princeton, Arthur Samuel from IBM, and Ray Solomonoff and Oliver Selfridge from MIT. All the scholars presented their works and gave their contribution, but the real protagonists were without any doubt Allen Newell and Herbert Simon from Carnegie Tech. Arousing the admiration of their colleagues, Newell and Simon presented a reasoning program, the *Logic Theorist* (LT), now considered the first AI program.[4]

Soon after the workshop, the program was able to prove 38 of the first

[3] Perhaps the Dartmouth workshop did not lead to any new breakthroughs, but it did introduce all the major figures to each other. For the next 20 years, the field would be dominated by these scholars and their students and colleagues at MIT, CMU, Stanford, and IBM.

[4] On the historical importance of this program see D. Cravier, *AI: The Tumultuous Search for AI* (New York: BasicBooks, 1993); P. McCorduck, *Machines Who Think* (Natick: A. K. Peters, 2004); S. Russel, and P. Norvig, *AI: A Modern Approach* (Upper Saddle River, New Jersey: Prentice Hall, 2010).

52 theorems in Chapter 2 of Russell and Whitehead's *Principia Mathematica*. Russell was reportedly delighted when Simon showed him that the program had come up with a proof for one theorem that was shorter and more elegant than the one in *Principia*. Newell and Simon attempted to publish the new proof in *The Journal of Symbolic Logic* but it was rejected on the grounds that a new proof of an elementary mathematical theorem was not notable, apparently overlooking the fact that one of the authors was a computer program!

Beyond any doubt, this brief account of the extraordinary summer of 1956 shows us that AI embraced from the beginning the idea of simulating human faculties such as language use, conceptual reasoning and problem solving – capabilities that until then were retained to be exclusively human. None of the other scientific research programs were addressing these issues. The *Logic Theorist* interpreted this mission perfectly. Its existence proved that it was possible to make a machine conduct non-numerical reasoning.

Thanks to the new program, several concepts were introduced, which were bound to become central in AI research. I will try to summarize them starting from the image of the *Child Program*. More precisely, I now ask this question: how did the child *Logic Theorist* (starting from a hypothetical stage zero) learn to prove logic theorems? In other words, what was the training path that led him to achieve this particular skill? To answer this question, we need to isolate at least two elements:

- A) *Logic Theorist* explored a search tree. The root was the initial hypothesis and each branch was a deduction based on the rules of logic. The goal was somewhere in the tree, being the proposition the program intended to prove. The pathway along the branches that led to the goal was a proof – a series of statements, each deduced using the rules of logic that led from the hypothesis to the proposition to be proved.

- B) Newell and Simon realized that the search tree would grow exponentially and they needed to "trim" some branches, using "rules of thumb" to determine which pathways were unlikely to lead to the solution. They called these *ad hoc* rules "heuristics" and they also clarified their function. As will be better understood in the following story of AI, a proof may become too complex or go down a road that leads nowhere. In this case, to develop a method to overcome the intractable combinatorial explosion of exponentially growing searches is necessary.

The *Logic Theorist*'s first way of proceeding (A) tells us that *reasoning is a research* that does not proceed along a single path, but that explores alternatives and possibilities. The second aspect (B) is particularly relevant. It

shows that the effective acquisition of knowledge does not depend exclusively on the correctness of the logical elaboration, but also on non-logical rules, derived from experience – rules which avoid the useless accumulation of passages and the so called "dead ends."

Now, if *Logic Theorist* could be the trainer of our *Child Program*, it would teach them that the starting point of their educational path is the essential pre-requisite of any future advancement. If we do not know what we already know, we cannot even understand what we do not know. In particular, without an exact awareness of our initial level, we would not be able to develop a program to achieve our goal of improvement. Regarding this last aspect, namely the achievement of an educational goal, the *Logic Theorist* teaches that it must be rigorously formulated. In many cases, a change of purpose frustrates the work already done and forces us to start over with deleterious effects on the effectiveness of the performance. Finally, our *Child Program* should learn the most important thing from the *Theorist*: the best educational path is made up of choices at crossroads that are not the strict consequences of reasoning. At some point, it is all about making decisions on shortcuts and alternative routes to avoid unnecessary work and wasted time.

3. Problem Solving and Symbolic Rationality

The *Logic Theorist* was an undeniably successful program, but its capabilities were still limited to a narrow domain of knowledge. For this reason, Newell and Simon designed another milestone in the history of AI, namely *the General Problem Solver*, or *GPS* (1957). Unlike *Logic Theorist*, this program was designed from the start to imitate the human *problem-solving* protocols beyond the mere domain of formal logic. This intent was expressed by Newell and Simon in the presentation report, as follows:

> Construction and investigation of this program is a part of research effort by the authors to understand the information processes that underlie human, adaptive and creative abilities. The approach is synthetic – to construct computer programs that can solve problems requiring intelligence and adaptation, and to discover which varieties of these programs can be matched to data on human problem solving.[5]

The salient features[6] of GPS were: a) the recursive nature of its problem

[5] A. Newell, and H. A. Simon, *Report on a General Problem-Solving Program* (Pittsburgh: Carnegie Institute of Technology, 1958): 11.
[6] Ibid.

solving activities, b) the separation of problem contents from resolution techniques, to increase the generalization of the algorithm, c) means-ends analysis and planning as the two general resolution algorithms d) the language used to code the algorithm and the IPL. In general, any problem that can be expressed as a set of well-formed formulas and that is a directed graph with one or more axioms and desired conclusions could be solved, at least theoretically, by GPS.

Particularly significant for our perspective is the circumstance that GPS was the first computer program that *separated the definition of the problems from the strategy to solve it*. The clear distinction between these two moments was not clear at all before the invention of GPS. A problem can be defined formally by five components. 1) The *initial state* that the agent starts in. For example, I am in the city of Padua and I want to reach the city of Milan. Of course it is crucial to know that I am in Padua and that I have to leave Padua if I want to solve my problem. Had I been in Bologna, my different start point would have required another route to get to Milan. 2) *A description of the possible actions available to the agent*. Given a particular initial state (for example: I am at the Padua train station), *I have to know the set of actions that can be executed starting from the train station* (for example: look at a timetable, decide which train to take, buy a ticket…). 3) *A description of what each action does*. Most of the time, starting from my initial state, I can have more actions available to achieve my goal (from Padua, there are several routes that can lead me to Milan; I need to know where each one would take me in relation to my goal). 4) At this point it is clear that the city of Milan is not a state like any other, *but is my final state, namely my goal*. The good problem solver must know and distinguish his goals well, otherwise he would not know what to look for. 5) Finally, the solver must be able to evaluate the cost of its actions in relation to an *optimal measure of its performance.* For getting to Milan, time is essential, so the cost of a path might be its length in terms of kilometers. Now, let us imagine that we wanted to extract an educational protocol from GPS and that we used it with our *Child Program*. What cognitive aspects should we highlight about it?

First of all, we should insist on the need to have a set of objectives already at the beginning of the process. We cannot solve a problem if we do not know what we want to achieve. The final goal cannot be found during the journey or changed from time to time depending on the circumstances. It must be clearly defined before leaving. Solving a problem is like traveling. We proceed through successive stages, we explore alternative routes, but then we try to get to the destination as quickly as possible. This means, however, that the whole process must be governed by a careful evaluation of

costs and benefits. A road that is too long can be convenient if it returns some advantages, otherwise it must be abandoned.

GPS was an indisputable success; at the beginning, it proved able to solve a set of simple problems that could be sufficiently formalized, but it turned out to be impossible to apply it to any real-world problem because the search got easily lost in the combinatorial explosion. Put in another way, the number of "walks" through the inferential digraph became computationally untenable.

The results achieved by GPS and other similar programs led Newell and Simon to formulate the famous *physical symbol system hypothesis,* which states what follows:

> The necessary and sufficient condition for a physical system to exhibit general intelligent action is that it be a physical symbol system.
>
> *Necessary* means that any physical system that exhibits general intelligence will be an instance of a physical symbol system.
>
> *Sufficient* means that any physical symbol system can be organized further to exhibit general intelligent action.
>
> *General intelligent action* means the same scope of intelligence seen in human action: that in real situations behavior appropriate to the ends of the system and adaptive to the demands of the environment can occur, within some physical limits.[7]

The consequences of this hypothesis – which was immediately questioned – are many and important. First, it establishes the following rule: *if* a physical system (the letters of an alphabet, the digits of a numerical (0 and 1) computer's memory, the material pieces of a game) exhibits symbolic behavior – that is, it is capable of manipulating its elements according to rules – *then* the property of general intelligence must be ascribed to this system. The implication between symbolism and intelligence established by Newell and Simon is close. It implies, for example, that, if both our *Child Program* and the mind of a real child demonstrate the ability to deal with symbols in certain domains, we should consider both as having the same cognitive capacities for the concerned symbolic field.

The second consequence is equally significant: symbolic intelligence is *action*, that is, it can be expressed in public and describable behaviors, which transform a certain state of the physical world. More precisely, it is not that there is a mental manipulation of the symbols and, in a second moment, an

[7] A. Newell, "Physical Symbol Systems," *Cognitive Science*, 4 (1980): 170.

action; on the contrary, intelligence and manifest behavior are indistinguishable, even by thought. The position of Newell and Simon is certainly reductionist and I do not intend to defend it at all. However, it identifies the general core of extended intelligence, which consists in the *ability to manipulate a physical system of symbols* and in the possibility of establishing rules that *make it possible to implement, modify and evaluate this capacity*.

As we will see later, this hypothesis has been challenged from many directions.

4. COMMON SENSE

1958 was an extraordinary year for the development of AI. John McCarthy moved from Dartmouth to MIT and there made several crucial contributions within a few months. In the first place, he defined the high-level language LISP, which was to become the dominant AI programming language for the next 30 years. LISP is an acronym for the expression "list processing," which indicates the basic structure that permeates the syntax of this functional language: the list. A list is a sequence, whose length can vary, which can contain numbers, symbols or other lists nested in it. LISP programs are themselves represented as lists and can therefore be manipulated like any other data. All this ensures enormous advantages to the programming activity and allows to quickly define a wide range of problems. As one of the oldest programming languages still in use, LISP offers several different dialects and has influenced the development of other languages.

In relation to this technical creation, McCarthy published in the same year the article *Programs with Common Sense*,[8] where he explained his "philosophy of learning." In his contribution McCarthy described the *Advice Taker*, a hypothetical program that can be considered as the first complete AI system. Like the *Logic Theorist*, McCarthy's program was designed to use knowledge to search for solutions to problems. But unlike other programs, it aimed to embody general knowledge of the world. For example, it showed how some simple axioms would enable the program to generate a plan to drive to the airport. McCarthy is thus able to give a precise definition of *common sense*, which he understands as a dynamic and cognitively improvable property. His words are as follows:

[8] *Programs with Common Sense* was probably the first paper on logical AI, i.e. AI in which logic is the method of representing information in computer memory and not just the subject matter of the program. The paper was given in the Teddington Conference on the Mechanization of Thought Processes in December 1958 and printed in the proceedings of that conference. It may also be the first paper to propose common sense reasoning ability as the key to AI.

> This property is expected to have much in common with what makes us describe certain humans as having common sense. We shall therefore say that *a program has common sense if it automatically deduces for itself a sufficiently wide class of immediate consequences of anything it is told and what it already knows*.[9]

According to McCarthy, *common sense* is not a particular set of knowledge, but the ability to *independently* derive information from what we already know. The objective of McCarthy was to make a program – Advice Taker – that could learn from its experience as effectively as humans do.

And what humans do is accumulating knowledge by listening to a master and then learn how to derive contents not explicitly communicated from this background. Now, McCarthy argues, "*in order for a program to be capable of learning something it must first be capable of being told it*."[10] This is to say that Advice Taker (and this explains its name!) would be able to learn in similar way to humans, if we could tell him how to do it.

There is an important distinction – McCarthy finally observes – between the way in which a computer is programmed and the way in which a human being is educated. A machine is educated mainly in the form of a sequence of imperative sentences, while a human is educated mostly in declarative sentences describing the situation where an action is required together with some imperatives that express what is wanted. The superiority of the declarative model of education/programming is evident when one considers its advantages in comparison to the imperative one. He writes:

> Advantages of Imperative Sentences:
> 1. A procedure described in imperatives is already laid out and is carried out faster.
> 2. One starts with a machine in a basic state and does not assume previous knowledge on the part of the machine.
>
> Advantages of Declarative Sentences:
> 1. Advantage can be taken from previous knowledge.
> 2. Declarative sentences entail logical consequences, and they can be arranged in such a way that the machine will have access to sufficiently simple logical consequences of what it is told and what it previously knew.
> 3. The meaning of declaratives is much less dependent on their order than is the case with imperatives. This makes it easier to have afterthoughts.
> 4. The effect of a declarative is less dependent on the previous state of the

[9] J. McCarthy, *Programs with common sense, in Proceedings Symposium on Mechanisation of Thought Processes*, Vol. 1 (London: Her Majesty's Stationery Office, 1958): 77-84.
[10] Ibid.: 79.

> system so that less knowledge of this state is required on the part of the instructor.[11]

In these passages McCarthy establishes an essential difference between imperative programming/education and declarative programming/education. In the first case, it is about giving our *Child program* all the instructions (the program) to solve a problem, with the recommendation to follow them without any deviation. No prior knowledge of the domain is required with this method. Each *Child Program* can easily be led from the zero stage to the expert stage. What it may already know has no role in the learning process. This means that our student will be very skilled in solving problems in environments whose features he knows, but will be quite "clumsy" in new situations for which he received no rules. In these cases, the knowledge he already possesses will be useless to him. To acting effectively, he will have to receive further training. This happens because imperative programming languages dictates *how* something should be done. They are written as a step-by-step guide (how!) for the computer and they describe which passages must be performed to reach the desired solution.

Vice versa, the declarative programming paradigm does not provide absolutely precise descriptions. The characteristic of declarative programming languages is that they always describe the desired *end result* instead of showing all the work steps. The declarative training focuses on the *what,* not on the *how*. To achieve the goal, the declarative programming determines automatically the path leading to the solution. It provides an approach that works well as long as the specifics of the final state are clearly defined, and accordingly there is an appropriate execution process. If both of these conditions are satisfied, declarative programming is very effective. An example can help to understand the difference between imperative and declarative programming. Let us imagine that we have to build an Ikea wardrobe. There are two ways to proceed: either we imperatively follow the assembly instructions (HOW) or we look at a perfectly built wardrobe (WHAT) and imagine the steps we would have to take in order to get to the final result. However, these states must be thought of as relatively independent of the state that immediately precedes them, as they have consequences that have value in relation to the goal and not to the process. Ideally, there are various ways to build a wardrobe. What matters is the result, and this depends on the measurable consequences of each step we take. But this means that what we have learned about building a wardrobe declaratively can be used, with minimal adaptation, to build a kitchen furniture. By virtue of its level of abstraction and of the open space left for the HOW, a declarative training will allow

[11] Ibid.

more flexibility because it teaches to use prior knowledge in unexpected environments.

5. TRANSLATION, COMPLEXITY AND KNOWLEDGE-BASED SYSTEMS

The first stages of development of the AI paradigm were accompanied by great optimism about possible developments in the future.[12] However, when the scope of the programs began to include broader and more complex problems than the simple "microworlds"[13] used up to then, the first difficulties emerged. An interesting case is represented by the first attempts to automatically translate from one language to another. It must be said that the actual concept of *Machine Translation* developed in a more concrete way around the 1930s. Around this time, French-Armenian engineer Georges Artsrouni[14] and Russian engineer Peter Troyansky[15] presented two innovative proposals concerning the first patents for translation machines. The so called "mechanical brain" conceived by Artsouni was a not so innovative device for the general processing (archiving, searching, consulting) of the information on tape, which could be used as a bilingual dictionary thanks to a word-by-word substitution mechanism. Instead, the most advanced model designed by Smirnov-Trojansky used a bilingual dictionary and a method for correlating grammatical rules existing in different languages. The translation process was divided into three phases: transformation of the original text into a *logi-*

[12] From the beginning, AI researchers were not shy about making predictions of their future successes. The following statement by Herbert Simon in 1957 is often quoted: «It is not my aim to surprise or shock you, but the simplest way I can summarize is to say that there are now in the world machines that think, that learn and that create. Moreover, their ability to do these things is going to increase rapidly until – in a visible future – the range of problems they can handle will be coextensive with the range to which the human mind has been applied», see H. A. Simon, and A. Newell, "Heuristic Problem Solving: The next Advance in Operations Research," *Operations Research*, 6, 1 (1958): 8.

[13] The most famous microworld was the blocks world, which consists of a set of solid blocks placed on a tabletop. A typical task in this world is to rearrange the blocks in a certain way according to parameters such as: "the highest one" or "the lowest one" and so on. See T. Winograd, "*Understanding natural language*," *Cognitive Psychology*, 3, 1 (1972).

[14] Georges Artsrouni (1893 -1922). Engineer of Armenian origin who emigrated to Paris. It is believed that he was the first not only to conceive, but also to build an automatic translation machine. For more information see: M. Daumas, "Les machines à traduire de Georges Artsrouni," *Revue d'histoire des sciences*, 18, 3 (1965): 283-302.

[15] Pyotr Petrovich Smirnov-Troyansky (1894 -1950). Of humble origins, he was a professor of social sciences and technology at institutions of higher learning. An important Technical encyclopedia is owed to him, even if he dedicated his whole life to the construction of a translation machine. For an overview of his work see: J. Hutchins, and E. Lovtskii, "Petr Petrovich Troyanskii (1894-1950): A Forgotten Pioneer of Mechanical Translation," *Machine Translation*, 15, 3 (2000): 187-221.

cal form modeled on the basis of the source language; transformation of this *logical form* into a second *logical form* modeled on the basis of the target language; transformation of this second *logical form* into a text in the target language. The focus of the translation process was the syntactic relationships between the terms in a sentence and the role – the logical form – that each of them played. Obviously, these systems could not function adequately, except in casual manner or for very short texts.

The actual history of MT (Machine Translation) begins in 1949 with engineer Warren Weaver, who first proposed to create a computer program capable of translating text from one language to another without any human intervention. In the paper entitled *Translation*,[16] written for the Rockefeller Foundation's Natural Science Division, the American engineer and mathematician formulated some hypotheses about the potential and the methods of TA. He argued for the validity of the word-for-word substitution method, and proposed supplementing it with applied statistical techniques to detect the frequency of words and characters in parallel texts. Weaver's idea quickly managed to capture the attention of several companies, which decided to fund the project. In the 1950s the limits of any machine translation procedure began to emerge. At a conference in 1952, Yehoshua Bar-Hillel[17] acknowledged for the first time that fully automated translation could be achieved only at the cost of some degree of carelessness and that accordingly a Fully Automatic High Quality Translation was an unattainable goal. Bar-Hillel was convinced that semantic ambiguity and syntactic complexity were the major obstacles for machine translation systems, so he developed a prototype machine translator using simplified forms of English such as Basic English, created by linguist and writer Charles Ogden[18] around the 1930s.

The problems and difficulties highlighted at the time by Bar Hillel have persisted to this day. Even the most advanced translator needs human assistance to produce a good quality text in the target language. This happens for two closely interrelated reasons: the first is that, especially at the beginning, machine translation relied exclusively on syntax, entirely neglecting the semantic value of symbols; the second, which is a consequence of the first, is

[16] W. Weaver, "Memorandum July 1949," *MT News International*, 22 (1999): 5-6.

[17] Yehoshua Bar-Hillel (1915-1975) was a philosopher, mathematician, and linguist. He was a pioneer in the field of machine translation and formal linguistics. His most cited contribution in this field of research is Y. Bar-Hillel, "Some Linguistic Problems Connected With Machine Translation," *Philosophy of Science*, 20, 3 (1953): 217-225.

[18] Charles Ogden (1889-1957) was a philosopher, linguist and writer. He was defined as a linguistic psychologist and is now mostly remembered as the inventor and propagator of Basic English. Ogden's science project on language is found in the very famous C. Ogden, *Basic English: A General Introduction with Rules and Grammar* (London: Paul Treber & Co., 1930).

that, in order to translate, we need to know more than logic. Speaking a language – Wittgenstein would say – is part of a *form of life*, in which meanings are closely connected to a context that cannot be shared by a machine. Keeping fixed our need to establish the best educational training for our *Child Program*, the case of translation teaches us that – for domains as complex as languages actually spoken – the rules of logic are not enough. Equally important are the understanding of word meanings and the constraints determined by the context in which a term is used – which are both elements dependent on language practice.

The second problem – more general than that of translation – is indicated today by the very evocative expression "complexity monster." As early as the late 1960s it was clear that many of the problems that AI was trying to solve were intractable. Most of the early programs solved problems by trying out different combinations of steps until the solution was found. This strategy worked for two obvious reasons: a) the domains considered contained very few objects b) the actions that could be taken by the program were limited and the resolution sequences were quite short. There was hope that more powerful hardware and larger memories could help to deal with larger problems. As became clear from the subsequent development of computational complexity theory, however, this hope was entirely misplaced.

The programs – despite technological advances – continued to prove capable of achieving solutions only in relation to small numbers of facts. *It was evident that the circumstance that a program can – in theory – find a solution, does not imply that the program itself is in practice capable of finding it*. The computational power of the artificial mind appeared imprisoned by seemingly insurmountable limits – limits so binding that it was thought that the AI paradigm should be abandoned. It seemed, in the end, that our *Child Program* had reached the end of its educational training, and that, unfortunately, it had learned very little from it and rather inaccurately. It mastered a general-purpose search automatism, which was carried out by successive steps of reasoning until a solution was found. However, when the problems became more complex and the object domains more numerous, the *Child Program* lost its ability to navigate its assigned field and did not achieve any consistent results.

The way seemed blocked. But, as is often the case, something happened. In 1969, at Stanford University, three very different figures – Ed Feigenbaum, Bruce Buchanan, and Joshua Lederberg[19] – attempted to reconstruct

[19] Edward Albert Feigenbaum (1936) – often considered as the "father of expert systems" – was a student of Herbert Simon. Bruce Buchanan (1939), on the other hand, was a philosopher who had retrained as a computer scientist, while Joshua Lederberg (1925) was an accomplished geneticist, so much so that he was awarded the Nobel Prize in 1958. The results of their joint work can be found in B. G. Buchanan, G. L. Sutherland, and E. A. Feigenbaum,

molecular structures from data provided by a mass spectrometer. A mass spectrum of a compound is produced by a mass spectrometer and is used to determine its molecular weight, that is the sum of the masses of its atomic constituents. For example, the compound water (H_2O) has a molecular weight of 18 since hydrogen has a mass of 1.01 and oxygen 16.00, and its mass spectrum has a peak at 18 units. The input of the DENDRAL program consisted of the formula of water, knowledge of the atomic mass numbers of individual molecule fragments (e.g., H_2 and O), and valence rules. Its output was to determine the possible combinations of atomic constituents whose mass added together would give 18. However, as the weight increases and the molecules become more complex, the number of possible compounds increases dramatically. As one can easily guess, the problem becomes intractable already for average-sized molecules. Again the monster of complexity!

The researchers of the project DENDRAL did not give up and tried a new move in the game. They learned from analytical chemists that their method was not to compute every possible formula step by step, but to look for well-known patterns of spectral peaks to which corresponded "compounds" of common occurrence from the molecule. What does it mean? Take, for example, the formula for Glucose: $C_6H_{12}O_6$. The Glucose has a mass of 180, but smaller mass groups can be generated from it, such as 0_2, which could have given, in the mass spectrum, a peak of 36. Now, to recognize a subgroup Water H2O, which has weight 18, DENDRAL reasoned in the following way:

> **if** there are two peaks at x_1 and x_2 such that
> (a) $x_1 + x_2 = M + 18$ (M is the mass of the whole molecule);
> (b) $x_1 - 28$ is a high peak;
> (c) $x_2 - 28$ is a high peak;
> (d) At least one of x_1 and x_2 is high.
> **then** there is a *water* subgroup.

By incorporating the heuristics of expert chemists, DENDRAL avoided the monster of complexity and greatly reduced the number of searchable structures. The program knew not only the formal rules of elemental composition (a chemical theory) but also other special instructions. In a way, DENDRAL had learned and cleverly used the Mc Carthy's approach, that is, the separation between knowledge of a domain (rules and objects) and reasoning about this domain. DENDRAL could provide our *Child Program*

Heuristic DENDRAL. A program for generating explanatory hypotheses in organic chemistry, in B. Meltzer, D. Michie, and M. Swann (Eds.), *Machine Intelligence* (Edinburgh: Edinburgh University Press, 1969).

with a new and more powerful educational training. It did not consist of general rules potentially applicable to every domain, but of in-depth knowledge of a specific field – which in this case was analytical chemistry. DENDRAL could perform broader steps of reasoning and could more easily handle significant cases that typically occur in narrow areas of expertise. Somehow, the program solved a difficult problem from what it already knew, and what it already knew was the *solution*.

6. THE NEURON AS A COMPUTATIONAL MACHINE

It is very often forgotten that one of the earliest contributionsto the field the field of AI was the paper *A logical calculus of the ideas immanent in nervous activity*[20] – appeared in 1943 and written by Warren McCulloch and Walter Pitts. In this work, they combined three theoretical elements: knowledge of the physiology and functioning of neurons in the brain; a formal analysis of propositional logic based on the symbolism of Language II (Carnap, 1938) and augmented with various notions taken from Russell and Whitehead (1927); and, finally, Turing's theory of computation. They proposed a model of artificial neurons in which each neuron is characterized by being "on" or "off," with a switch to "on" occurring in response to stimulation by a sufficient number of neighboring neurons.

McCulloch and Pitts established that a) the activity of neuron is an "all-or-none" process and that b) "the 'all-or-none' law of nervous activity is sufficient to ensure that the activity of any neuron may be represented as a proposition" so that the physiological relations existing among nervous activities "correspond to relations among the propositions."[21] In this way, to each reaction (state) of any neuron can be associated a corresponding assertion (V/F or 0 1) of a simple proposition. With this (perhaps too hasty) assimilation of neural activity to propositional calculus, McCulloch and Pitts showed that any computable function could be computed by some network of connected neurons, and that all the logical connectives (and, or, not, etc.) could be implemented by simple net structures. McCulloch and Pitts were certainly the first to suggest that suitably defined networks could learn and, in general, that they were capable not only of reacting, but of making distinctions and fixing the information.

This intuition found its first determination in the subsequent *Hebb's rule*. In his book *The organization of Behavior* (1949) Donald Hebb demonstrated

[20] W. S. McCulloch, and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysic*, 5 (1943): 115-133.
[21] Ibid.: 3.

a simple updating rule for modifying the connection strengths between neurons.

> Let us assume that the persistence or repetition of a reverberatory activity (or "trace") tends to induce lasting cellular changes that add to its stability. [...] When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.[22]

This axiom, with its apparent simplicity, says something very important. In particular, it states that in the brain information is not located in a single point, but is represented by several interconnected elements. A single concept corresponds to multiple neurons, and each neuron participates in the representation of multiple concepts. According to Hebb, neurons that excite each other form so-called cell clusters. These groups of cells, connected in the most intricate ways and belonging to distant regions of the brain, are the true *medium* of information, even of the learned one.

The second consequence of Hebb's model is even more significant. Unlike what happens with the Turing Machine, the attention to the physiology of the brain makes us understand that learning is not (exclusively) a sequential process. If we imagine that educational growth is the passing from one step to another in a demonstration, or something like the successive accumulation of information – we are probably dominated by a bias. In fact – as the connectionist models lead us to believe – neurons learn simultaneously. For each educational event, the brain learns as an interconnected system, involving billions of neurons even for the simplest representation.

The first "artificial" implementation of this learning model was the perceptron proposed in 1958 by the American psychologist Frank Rosenblatt.[23] Although it is now considered outdated, the perceptron nevertheless represents the starting model for the design of complex networks. It is a more general computational model than the McCulloch-Pitts model, since it introduces numerical weights and a special interconnection path. In the original model, the computational units are threshold elements and the connection is derived stochastically. Learning or storage is achieved by feedback by adjusting the numerical weights until the output is made equal to the desired output. The classical perceptron is actually a true network for solving path recognition problems, as it is based on the fundamental idea of having the system learn a method of recognizing specific input paths. More precisely,

---

[22] D. Hebb, *The organization of Behavior* (New York: Wiley & Sons, 1949): 56.

[23] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, 65, 6 (1958): 386-408.

the perceptron represents a special case of a neural network with mutually independent outputs. It is precisely this property that makes the study of the perceptron basic to its use as an elementary model of a larger neural network composed of numerous neurons.

The computational capabilities of a single perceptron was not high and the achievable performance depended on both the set of inputs and the function to be implemented. However, the perceptron imposed on the field of AI two thought-provoking elements: a learning model based on pattern recognition and classification (not just symbol manipulation!) and the possibility of improvement based on training. Appropriately equipped with a perceptron, our *Child Program* would have been able to recognize the images and shapes of its environment, perceive the features of its parents and make precise classifications of objects in the world. A new way of knowing was emerging. As it does among us humans, the "child" required to be educated, and it seemed to be able to improve its performance as its experience increased. Despite this, it was still not entirely clear how to do this with the necessary precision. The AI paradigm needed to further expand its theoretical horizons.

7. MACHINE LEARNING

A specific subset of AI is *Machine Learning* (ML). The goal of this discipline is to develop artificial systems, now called agents, to whom is given the *ability to learn from already acquired knowledge*. It is not a matter of inputting data and programs, but of teaching machines to autonomously *learn something new*. The difference between the two approaches is decisive. The expert systems such as DENDRAL were usually built without any learning component and the basic knowledge needed to perform the referenced task was usually constructed through a manual process called "knowledge engineering," where a computer scientist collaborated with an expert in the field, with the goal of transferring some of the expert's knowledge in the system, obviously using the appropriate formalism. These programs exhibit skills at the same level as humans, but only in very narrow areas such as, for example, diagnosing limited classes of diseases, suggesting specific antibiotic treatments, or determining the optimal configuration of a computer... and so on.

Nevertheless, the expert systems approach quickly showed its limitations. The unsolvable problem was the difficulty of knowledge acquisition. The main source to make an expert system work is in fact the so-called knowledge base, i.e., a repository of skills related to a specific domain used by the system to accomplish its task. The content of the knowledge base had

to be acquired manually, through a collaborative activity between an expert in the application domain and a computer scientist, who knew nothing about the domain under consideration. This procedure, which had worked in the beginning, became increasingly complicated, uneconomical, as well as inapplicable in many domains.

Once again, an innovative idea was needed. It was found by Judea Perl in 1988, in a groundbreaking book[24] that opened the current history of AI. Pearl proposed the use of probability calculus to deal with both modeling and inference in intelligent systems. The use of probability had been rejected by some of the fathers of AI (particularly by John McCarthy) as it was considered epistemologically and computationally inappropriate for AI. Pearl defended his view by demonstrating that a consistent interpretation of probabilistic theory was possible through a graph-based formalism called *Probabilistic Graphical Models*, the main representative of which are *Bayesian Networks*. The use of such formalisms allows compact and efficient modeling of uncertain knowledge, and the use of specialized inference algorithms allows to answer all kinds of probabilistic questions; moreover, the ability to learn both the structure and parameters of the graphical model made it possible to overcome many limitations of logic-based systems and opened the way for more real-world applications.

Over time, different statistical methods were developed such as the *Support Vector Machines (SM)* and the *Ensemble Learning approaches*. Ensemble approaches implement the so-called "folk wisdom" idea: if a certain process can perform a given prediction with a particular performance, then considering several processes performing the same prediction can – in theory – increase the overall performance with respect to the required task. The process can be differentiated using different algorithms or data. For example, it is possible to run a number of different algorithms that produce different prediction models on the same data set, and then send the prediction provided by the majority of the models – possibly weighing the result according to the respective certainty of the prediction – so that the models that predict a result with higher certainty have a greater weight in producing a final answer. In contrast, a different ensemble approach might be to use the same algorithm or model several times, but with a different set of data. This type of approach is closely related to some statistical computational methodologies, such as the renowned *bootstrap* method, which is an effective way to improve the final performance of a prediction model. Performance is usually

[24] J. Pearl, *Probabilistic Reasoning in Intelligent Systems* (Burlington: Morgan Kaufmann, 1988). On the technological side, we cannot forget the development of neural networks and deep learning. I can only refer to an author who is considered the "Godfather" of deep learning, Geoffrey Hinton. For an introduction see Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, 521 (2015): 436-444.

measured by the accuracy metric, which consists of the percentage of correct predictions compared to the entire set of predictions provided. Obtaining different opinions on the same set of data or the same opinion on a similar set of data is the method used by the ensemble approach to increase final accuracy.

In any case, whichever approach is chosen, the accuracy of the result will depend on the large availability of input data. In 2010, *The Economist Magazine* came out with a significant headline on its cover, *The data deluge*. The cover featured a man with an upside-down umbrella under a deluge of data. The upside-down umbrella collected some of this data and a plant was watered with the collected rain. The era of *Big Data* was beginning, and AI effectively became a way to exploit a huge amount of data made available by both electronic devices and human activities.

We have arrived at the present time, but what does the advent of *Machine Learning* mean for our *Child Program*? The answer is the following: it means a complete reversal of perspective. First, machine learning enables knowledge and action in environments characterized by uncertainty, where reasoning is required based on degrees of belief and not on the simple true-false dichotomy. The point of view becomes that of expectation and probability that something will happen. Data are the evidence for reasoning on which new agents rely – and the *data* are real evidence, that is what we already know and not abstract assumptions. Even more important, *Machine Learning* radically changes the status of algorithms. Every program has an *input* and an *output*. Data are provided by the environment, and the program does what it necessarily has to do, following what someone else has determined for it. The Machine learning starts from the data and the desired result. Only later it does arrive at the algorithm capable of passing from one (*data*) to the other (*result*). The Learning algorithms are algorithms that write other algorithms. They do not just learn how to deal with data. *They learn how to learn better*.

8. CONCLUSIONS

We imagined that the history of AI has been a "pedagogical" history. At the heart of our fiction was the following problem: can we artificially simulate not the finished result of an educational process, but the process itself? What are the steps through which a hypothetical child program proceeds from an initial – uneducated state – to an "adult" and trained one? I believe that some decisive moments in the development of AI systems have been (not always in a conscious way) attempts to answer this problem.

Starting with the idea that any improvement in the capabilities of the *Child Program* simulated similar capabilities of a real child, I assumed that the programming methods created in the history of AI could help us better define the nature of education in general – and thus also the nature of human education. I want to make myself clear. I am not going to establish identity of process and method between program implementation and human training, but to look at human education from the perspective marked by the attempts to educate the machines. After all, those who have made the history of AI – and we have seen this clearly – have always considered the human processes as the starting model for their research. Why then not consider the opposite path, which looks at humans from the perspective of machines, epistemically permissible? As a result of all this, I try to define below in list form the elements of extended education (for machine and human being), which should be considered valid for any educational training:

Training of Extended Education (TEE)

Starting State	The starting state, the zero state, must be well defined. If we do not know where we start from, we cannot know which path to take to reach our goals. The zero state never corresponds to a *tabula rasa*, but to a system of prior information and rules capable of conditioning what follows.
Target Status	In any educational process, objectives must be precisely defined. Without a goal, neither a path nor a strategy can be determined. In relation to the target, two paths of approach are possible: either we formulate a step-by-step program *a priori* or we set the goal and only then we define the best strategy to achieve it.
Heuristics	The education in complex fields of knowledge needs to identify heuristic strategies. The complexity of domains produces intractability and does not allow solutions to be reached in reasonable time. An educational process cannot be protracted indefinitely. Heuristics do not follow a clear path, but rely on intuition and the temporary state of circumstances in order to achieve the goal or generate new

	knowledge. It introduces progressive leaps and cheap and effective deviations from the time and quality of a performance.
Rules	In every educational field we need imperative rules, but also open, declarative rules. Sometimes the rule must be found based on the data we already have. With complex domains only the invention of appropriate rules guarantees deep knowledge.
Data	In knowledge acquisition, data are never neutral elements. Their quality and quantity also determine the kind of rules we use as well as the amount of certainty we can achieve. Knowing a lot about our environment helps us to act more effectively.
Orient yourself	The metaphor that should guide us in education is not that of the vessel to be filled or even that of the paradigm to be declined, but that of *navigating through an unknown territory.*
Deductive/Inductive training	The best educational training should involve a mastery of both deductive and inductive strategies, encouraging – as needed – the transition from one to the other.

We tried to educate our *Child Program* in the way we would educate a human child. Perhaps the time is coming to understand whether the methods we invented for the artificial child are now useful for educating our flesh-and-blood children.

BIBLIOGRAPHY

Bostrom, N. 2014. *Superintelligence: Paths, dangers, strategies*. Oxford: Oxford University Press.

Buchanan, B. G., Sutherland, G. L. and Feigenbaum, E. A. 1969. *Heuristic DENDRAL. A program for generating explanatory hypotheses in organic chemistry*. In Meltzer, D., Michie, M. and Swann B. (Eds.) 1969. *Machine Intelligence.* Edinburgh: Edinburgh University Press, N. 4: 209-254.

Cravier, D. 1993. *AI: The Tumultuous Search for AI*. New York: Basic Books.

Daumas, M. 1965. “Les machines à traduire de Georges Artsrouni.” *Revue d’histoire des sciences*, N. 183: 283-302.

Domingos, P. 2012. “A few useful things to know about machine learning.” *Communications of the ACM*, N. 55: 78-87.

Fernández-Delgado, M., Cernadas, E., Barro, S. and Amorim, D. 2014. “Do we need hundreds of classifiers to solve real world classification problems.” *Journal of Machine Learning Research*, N. 15: 3133-3181.

Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J., Schlaefer, N. and Welty, C. 2010. “Building Watson: An Overview of the Deep QA Project.” *AI Magazine*, N. 31: 59-79.

Hebb, D. 1949. *The organization of Behavior*. New York: Wiley & Sons.

Hutchins, J. and Lovtskii, E. 2000. “Petr Petrovich Troyanskii (1894-1950): A Forgotten Pioneer of Mechanical Translation.” *Machine Translation*, Vol. 15, N. 3: 187-221.

Markoff, J. 2016. *Machines of Loving Grace: The Quest for Common Ground Between Humans and Robots*. Ecco, Reprint Edition.

McCarthy, J., Minsky, M., Rochester, N., and Shannon, C. E. 1955. “A proposal for the Dartmouth Summer Research Project on AI.” Reprint 2006 in *AI Magazine* , Vol. 27, N. 4: 12-14.

McCarthy, J. 1959. *Programs with common sense,* in *Proceedings Symposium on Mechanisation of Thought Processes*. Vol. 1, London: Her Majesty’s Stationery Office.

McCorduck, P. 2004. *Machines Who Think*. Natick: A. K. Peters.

McCulloch, W. S. and Pitts, W. 1943. “A logical calculus of the ideas immanent in nervous activity.” *Bulletin of Mathematical Biophysic*, Vol. 5: 115-133.

Muggleton, S. 2014. “Alan Turing and the development of AI.” *AI Communication*. Vol. 27, N.1: 3-10.

Newell, A. and Simon, H. A. 1958 (rev. 1959). *Report on a General Problem-Solving Program*, Carnegie Institut of Technology, Pittsburgh (USA): 1-27.

Newell, A. 1980. “Physical Symbol Systems.” *Cognitive Science*, N. 4: 135-183.

Roitblat, H. L. 2020. *Algorithms Are Not Enough: Creating General AI*. Cambridge (MA): MIT Press.

Roitblat, H. L., Kershaw, A. and Oot, P. 2010. “Document Categorization in Legal Electronic Discovery: Computer Classification vs. Manual Review.” *Journal of the American Society for Information Science and Technology*, N. 61 (1): 70-80.

Rosenblatt, F. 1958. “The perceptron: A probabilistic model for information storage and organization in the brain.” *Psychological Review*, N. 65: 386-408.

Russell, S. and Norvig, P. 2020. *AI: A Modern Approach*. Upper Saddle River, New Jersey: Pearson.

Samuel, A. L. 1959. “Some studies in machine learning using the game of checkers. Computation & intelligence: collected readings.” *American Association for AI*: 391–414.

Shafer, G. 1976. *A mathematical theory of evidence*. Princeton NJ: Princeton University Press.

Simon, H. A. and Newell, A. 1958. "Heuristic Problem Solving: The next Advance in Operations Research." *Operations Research*, Vol. 6, N. 1: 1-10.

Turing, A. M. 1936. *On computable numbers with an application to the Entscheidungsproblem*. In Davis, M. (Ed.) 1965. *The undecidable*. NY: Raven Press . Original work published in *Proceedings of the London Mathematical Society*, Ser. 2, Vol. 42, N. 7: 230-265.

Turing, A. M. 1947. *Lecture to the London Mathematical Society on 20 February 1947*. Reprinted in Ince D. C. (Ed.) 1992. *Collected works of A. M. Turing: Mechanical intelligence*. Amsterdam: North Holland: 87-105.

Turing, A. M. 1950. "Computing machinery and intelligence." *Mind*, N. 59: 433-460.

Weaver, W. 1999. "Memorandum July 1949." *MT News International*, N. 22: 5-6.

Winograd, T. 1972. "Understanding natural language." *Cognitive Psychology*, N.3 (1): 25-42.