COMPUTATIONAL INTELLIGENCE

K. Mainzer
Institute of Interdisciplinary Informatics, University of Augsburg, Germany

Keywords: Affective Computing, Artificial Intelligence, Cellular Automata, Computability, Computational Complexity, Decidability, Genetic Programming, Intelligent Agents, Internet, Knowledge-based Systems, Learning Algorithms, Neural Networks, Swarm Intelligence, Ubiquitous Computing.

Contents

Summary

Can machines think? This famous question of Alan Turing has inspired computational intelligence. The historical roots of computational intelligence stem back to the 17th century and the program for mechanizing thinking (mathesis universalis). The modern theory of computability distinguishes complexity classes of problems, meaning the order of corresponding functions describing the computational time of their algorithms or computational programs. Modern computer science is interested not only in the complexity of universal problem solving but also in the complexity of knowledge-based programs. Famous examples are expert systems simulating the problem solving behavior of human experts in their specialized fields. But the algorithmic mechanization of thinking with program-controlled computers has some severe obstacles which cannot be overcome by growing computational capacities. For example, pattern recognition, the coordination of movements, and other complex tasks of human learning cannot be

mastered by conventional computer programs. Artificial neural networks realize the principles of complex dynamical systems. They are inspired by the successful technical applications of nonlinear dynamics to solid-state physics, spin-glass physics, chemical parallel computer, optical parallel computer, laser systems, and the human brain. There are already some applications of artificial neural networks in neurobionics, medicine, and robotics. Besides computational intelligence or "artificial intelligence" (AI) as a classical discipline of computer science "artificial life" (AL) is a newgrowing field of research. Genetic programming made computational evolution possible. Virtual agents are designed with different degrees of autonomy, mobility, reactivity or learning capabilities for communicating and cooperating with the Internet. Distributed computational intelligence and life transform the World Wide Web into an computational ecology with ubiquitous computing and e-commerce.

1. Review of Subject Articles

1.1. General Principles and Purposes of Computational Intelligence

L. Reznik considers computational intelligence as integrity of theories, applying methods and models analogous to those demonstrated by biological intelligent systems in problem solving and decision making. Among such theories are those which today are often collected as 'soft computing'.

1.2. Neural Networks

I. Vajda describes the development and foundations of neural networks from Rosenblatt's perceptron up to Kohonen's self-organizing maps and further present approaches used as tools to solve practical problems without any biological analogies.

1.3. Simulated Annealing

From Statistical Thermodynamics to Combinatory Problem Solving **D. Thiel** presents the principle of the Simulated Annealing (SA) algorithm, corresponding to Boltzmann's thermodynamics, which has been particularly successful in solving various combinatorial decision and optimisation problems of immense computational complexity (e.a., statisfiability problem, travelling salesman problem, quadratic assignment problem).

1.4. Adaptive Systems

R. Pla-Lopez suggests a general theory of adaptive systems changing their behavior through interaction with their environment, in order to simulate the computational intelligence of living systems.

1.5. Biological Intelligence and Computational Intelligence

G. A. Chauvet compares biological intelligence (e.a., motoric intelligence) with computational intelligence which may be implemented on a computer by means of a program or by means of a processor with an architecture designed for an analogical

representation of the physiological function. Such neuromimetic circuits may be used for the replacement of deficient organs in medicine or for the execution of human functions by machine.

2. Introduction

The five articles briefly commented summarize some of the main tendencies in today's development of Computational Intelligence. But to gain better insight into the process, and on the basis of these works, it is worthwhile to retrace the first historical steps in this direction. The mechanization of thoughts begins with the invention of mechanical devices for performing elementary arithmetic operations automatically. A mechanical calculation machine executes serial instructions step by step. First, there is an input mechanism by which a number is entered into the machine. A selector mechanism selects and provides the mechanical motion to cause the addition or subtraction of values on the register mechanism. The register mechanism is necessary to indicate the value of a number stored within the machine, technically realized by a series of wheels or disks. If a carry is generated because one of the digits in the result register advances from 9 to 0, then that carry must be propagated by a carry mechanism to the next digit or even across the entire result register. A control mechanism ensures that all gears are properly positioned at the end of each addition cycle to avoid false results or jamming the machine. An erasing mechanism has to reset the register mechanism to store a value of zero.

Wilhelm Schickard (1592-1635), professor of Hebrew, oriental languages, mathematics, astronomy, and geography, is presumed to be the first inventor of a mechanical calculating machine for the first four rules of arithmetic. The adding and subtracting part of his machine is realized by a gear drive with an automatic carry mechanism. The multiplication and division mechanism is based on Napier's multiplication tables. Blaise Pascal (1623-1662), the brilliant French mathematician and philosopher, invented an adding and subtracting machine with a sophisticated carry mechanism which in principle is still realized in our hodometers of today.

But it was Leibniz' mechanical calculating machine for the first four rules of arithmetic which contained each of the mechanical devices from the input, selector, and register mechanism to the carry, control, and erasing mechanism. The Leibniz machine became the prototype of a hand calculating machine. If we abstract from the technical details and particular mechanical constructions of Leibniz' machine, then we get a model of an ideal calculating machine which in principle is able to calculate all computable functions of natural numbers.

Leibniz (1646-1716) even designed a mechanical calculating machine for the binary number system with only two digits 0 and 1, which he discovered some years earlier. He described a mechanism for translating a decimal number into the corresponding binary number and vice versa. As modern electronic computers only have two states 1 (electronic impulse) and 0 (no electronic impulse), Leibniz truly became one of the pioneers of computer science.

Leibniz' historical machines suffered from many technical problems, because the

materials and technical skills then available were not up to the demands. Nevertheless, his design is part of a general research program for a *mathesis universalis* intended to simulate human thinking by calculation procedures ("algorithms") and to implement them on mechanical calculating machines. Leibniz proclaimed two basic disciplines of his *mathesis universalis*. An *ars iudicandi* should allow every scientific problem to be decided by an appropriate arithmetic algorithm after its codification into numeric symbols. An *ars iveniendi* should allow scientists to seeks and enumerate possible solutions of scientific problems. Leibniz was deeply convinced that there are universal algorithms to decide all problems in the world by mechanical devices.

In the 19th century it was the English mathematician and economist Charles Babbage who not only constructed the first program-controlled calculation machine (the "analytical engine") but also studied its economic and social consequences. A forerunner of his famous book *On the economy of machinery and manufactures* in 1841 was Adam Smith's idea of economic laws, which paralleled Newton's mechanical laws. In his book *The Wealth of Nations*, Smith described the industrial production of pins as an algorithmic procedure and anticipated Henry Ford's idea of program-controlled mass production in industry.

3. Computability, Decidability, and Complexity

The hand calculating machine can easily be generalized to Marvin Minsky's register-machine. It allows the general concept of computability to be defined in modern computer science. An ideal register machine has a finite number of registers which can store any finite number of a desired quantity. The program of a register machine is defined by two elementary procedures like adding and substracting a digit in a register. Programs of these two elementary devices can be chained or combined with some control mechanism. A numerical function is called computable by a register machine if there is such a program combining the function. Historically, some other, but equivalent formulations of machines were at first introduced independently by Alan Turing and Emil Post in 1936. A Turing machine can carry out any effective procedure provided it is correctly programmed. It consists of

(a) a control box in which a finite program is placed,
(b) a potentially infinite tape, divided lengthwise into squares,
(c) a device for scanning, or printing on one square of the tape at a time, and for moving along the tape or stopping, all under the command of the control box.

If the symbols used by a Turing machine are restricted to a stroke 1 and a blank ⋆, then a function computable function by a register machine can be proved to be computable by a Turing machine and vice versa. We must remember that every natural number x can be represented by a sequence of x strokes (for instance 3 by $|||$), each stroke on a square of the Turing tape. The blank ⋆ is used to denote that the square is empty (or the corresponding number is zero). In particular, a blank is necessary to separate sequences of strokes representing numbers. Thus, a Turing machine computing a function f with arguments $x_1, \ldots, x_n$, starts with tape $\ldots * x_1 * x_2 * \ldots * x_n * \ldots$ and stops with $\ldots * x_1 * x_2 * \ldots * x_n * f(x_1, \ldots, x_n) * \ldots$ on the tape.

From a logical point of view, a general purpose computer - as constructed by associates of John von Neumann in America and independently by Konrad Zuse in Germany - is a technical realization of a universal Turing machine which can simulate any kind of Turing program. Analogously, we can define a universal register machine which can execute any kind of register program. Actually, the general design of a von-Neumann computer consists of a central processor (program controller), a memory, an arithmetic unit, and input-output devices. It operates step by step in a largely serial fashion. A present-day computer à la von Neumann is really a generalized Turing machine. The efficiency of a Turing machine can be increased by the introduction of several tapes, which are not necessarily one-dimensional, each acted on by one or more heads, but reporting back to a single control box which coordinates all the activities of the machine. Thus, every computation of such a more effective machine can be done by an ordinary Turing machine. Concerning the complex system approach, even a Turing machine with several multidimensional tapes remains a sequential program-controlled computer, differing essentially from self-organizing systems like neural networks.

Besides Turing- and register machines, there are many other mathematically equivalent procedures for defining computable functions. Recursive functions are defined by procedures of functional substitution and iteration, beginning with some elementary functions (for instance, the successor function $n(x) = x + 1$) which are obviously computable. All these definitions of computability by Turing machines, register machines, recursive functions, etc., can be proved to be mathematically equivalent. Obviously, each of these precise concepts defines a procedure which is intuitively effective.

Thus, Alonzo Church postulated his famous thesis that the informal intuitive notion of an effective procedure is identical with one of these equivalent precise concepts, such as that of a Turing machine. Church's thesis cannot be proved, of course, because mathematically precise concepts are compared with an informal intuitive notion. Nevertheless, the mathematical equivalence of several precise concepts of computability which are intuitively effective confirms Church's thesis. Consequently, we can speak about computability, effectiveness, and computable functions without referring to particular effective procedures ("algorithms") like Turing machines, register machines, recursive functions, etc. According to Church's thesis, we may in particular say that every computational procedure (algorithm) can be calculated by a Turing machine. So every recursive function, as a kind of machine program, can be calculated by a general purpose computer.

Now we are able to define effective procedures of decision and enumerability, which were already demanded by Leibniz' program of a *mathesis universalis.* The characteristic function f_M of a subset M of natural numbers is defined as $f_M(x) = 1$ if x is an element of M, and as $f_M(x) = 0$ otherwise. Thus, a set M is defined as effectively decidable if its characteristic function saying whether or not a number belongs to M is effectively computable (or recursive).

A set M is defined as effectively (recursively) enumerable if there exists an effective (recursive) procedure f for generating its elements, one after another (formally $f(1) = x_1$, $f(2) = x_2, \ldots$ for all elements $x_1, x_2 \ldots$ from M). It can easily be proved that every

recursive (decidable) set is recursively enumerable. But there are recursively enumerable sets which are not decidable. These are the first hints that there are limits to Leibniz' originally optimistic program, based on a belief in universal decision procedures.

Concerning natural and artificial intelligence, the paradigm of effective computability implies that mind is represented by program-controlled machines, and mental structures refer to symbolic data structures, while mental processes implement algorithms. Historically the hard core of AI was established during the Dartmouth Conference in 1956 when leading researchers such as John McCarthy, Alan Newell, Herbert Simon, and others from different disciplines, formed the new scientific community of AI. They all were inspired by Turing's question "Can machines think?" in his famous article "Computing machinery and intelligence" in 1950.

In the tradition of Leibniz' *mathesis universalis* one could believe that human thinking could be formalized with a kind of universal calculus. In a modern version one could assume that human thinking could be represented by some powerful formal programming language. In any case, formulas are sequences of symbols which can be codified by natural numbers. Then assertions about objects would correspond to functions over numbers, conclusions would follow from some kind of effective numerical procedure, and so on.

Actually, the machine language of a modern computer consists of sequences of numbers, codifying every state and procedure of the machine. Thus, the operations of a computer can be described by an effective or recursive numerical procedure.

If human thinking can be represented by a recursive function, then by Church's thesis it can be represented by a Turing program which can be computed by a universal Turing machine. Thus, human thinking could be simulated by a general purpose computer and, in this sense, Turing's question must be answered with "yes". The premise that human thinking can be codified and represented by recursive procedures is, of course, doubtful. Even processes of mathematical thinking can be more complex than recursive functions. Recursiveness or Turing computability is only a theoretical limit of computability according to Church's thesis.

Complexity classes of problems (or corresponding functions) can be characterized by complexity degrees, which give the order of functions describing the computational time (or number of elementary computational steps) of algorithms (or computational programs) depending on the length of their inputs. The length of inputs may be measured by the number of decimal digits. According to the machine language of a computer it is convenient to codify decimal numbers into their binary codes with only binary numbers 0 and 1 and to define their length by the number of binary digits. For instance, 3 has the binary code 1 1 with the length 2. A function f has linear computational time if the computational time of f is not greater than $c \cong n$ for all inputs with length n and a constant c.

A function f has quadratic computational time if the computational time of f is not

greater than $c \cong n^2$ for all inputs with length n and a constant c. A function f has polynomial computational time if the computational time of f is not greater than $c \cong n^k$, which is assumed to be the leading term of a polynomial $p(n)$. A function f has exponential computational time of f is not greater than $c \cong n^{p(n)}$. Many practical and theoretical problems belong to the complexity class P of all functions which can be computed by a deterministic Turing machine in polynomial time.

NP means the complexity class of functions which can be computed by a non-deterministic Turing machine in polynomial time. By definition every P-problem is an NP-problem. But it is a crucial question of complexity theory whether P = NP or, in other words, whether problems which are solved by non-deterministic computers in polynomial time can also be solved by a deterministic computer in polynomial time. In his chapter, D. Thiel discusses some examples of NP-problems like the Satisfiability Problem (SAT), Travelling Salesman Problem (TSP), and Quadratic Assignment Problem (QAP) (see *Simulated Annealing*).

Obviously, complexity theory delivers degrees for the algorithmic power of Turing machines or Turing-type computers. The theory has practical consequences for scientific and industrial applications. But does it imply limitations for the human mind? The fundamental questions of complexity theory (for example N = NP or N ≠ NP) refer to the measurement of the speed, computational time, storage capacity, and so on, of algorithms. It is another question how one sets out to find more or less complex algorithms. This is the creative work of a computer scientist which is not considered in the complexity theory of algorithms.

On the other hand, Gödel's famous theorems are sometimes said to limit the mathematical power of computers and the human mind. His incompleteness theorem says that in every consistently axiomatized enlargement of formal number theory there is a (closed) formula which is not decidable. Actually, his theorem states that any adequate consistent arithmetical logic is incomplete in the sense that there exist true statements about the integers that cannot be proved within such a logic. Even if we enlarge our axiomatization by the undecidable formula, then there is another formula which is not decidable in the enlarged formalism.

But Gödel's theorem is only a limitation on human thinking under some essential assumptions: we must accept theorem-proving as the key to human intelligence. The theorem can only be applied to a mind-model that consists of a machine with all its knowledge carefully formalized. Furthermore, Gödel's theorem only places limitations upon consistent machines, while fuzziness, inconsistency, and puzzles are typical features of human decisions. We would not survive very long if we required a long period of careful theorem-proving before deciding whether or not we should act. It should also be considered that Turing machines have a fixed data structure, while the human mind is open to novel experience and able to learn from its mistakes. So Gödel's theorem limits a machine as much as a human closing his or her mind to new information.

-

-
-

Bibliography

Anderson J. A., Rosenfeld E. (eds.) (1988). Neurocomputing. Foundations of Research. The MIT Press, Cambridge MA [Interdisciplinary guide to the background of concepts in neuroscience, computer science, engineering, physics, cognitive science, and psychology]

Farmer D., Toffoli T., Wolfram S. (eds.) (1984). Cellular Automata. North-Holland, Amsterdam [Proceedings of an interdisciplinary workshop with leading experts of cellular automata]

Kohonen T. (1989). Self-Organization and Associative Memory. Springer, Berlin [Foundations of self-organizing neural maps and unsupervised learning algorithms]

Mainzer K. (1995). Computer - Neue Flügel des Geistes ? De Gruyter, Berlin/New York [History and Foundations of Computer Science with logical and epistemic analysis of artificial intelligence]

Mainzer K. (1997). Thinking in Complexity. The Complex Dynamics of Matter, Mind, and Mankind. Springer, New York [3rd enlarged introduction into nonlinear complex systems and their application in natural sciences, brain research, computer science, economics, social sciences and philosophy]

Mainzer K.(1997). Gehirn, Computer, Komplexität. Springer, Berlin [Introduction to brain research, artificial intelligence, and theory of neural networks]

Mainzer K. (1999). Computernetze und virtuelle Realität. Leben in der Wissensgesellschaft. Springer, Berlin [Computational foundations of computer networks, artificial life, bioinformatics, virtual agents, virtual reality and applications in the World Wide Web]

Neumann J. v. (1958). The Computer and the Brain. Yale University Press, New Haven [The computer pioneer John von Neumann analyzed analogies between a general purpose computer and the human brain.]

Picard R. W. (1997). Affective Computing. MIT Press, Cambridge MA [If we want computers to be genuinely intelligent and to interact naturally with us, we must give computers the ability to recognize, understand, even to have and express emotions.]

Rothermel K., Popescu-Zeletin R. (eds.) (1997). Mobile Agents. Springer, New York [Overview on agents which can be sent by bytecodes into the World Wide Web without online connection to the user]

Biographical Sketch

Prof. Dr. Klaus Mainzer, born 1947, studied Mathematics, Physics and Philosophy. After his Ph.D. (1973) and Habilitation for Philosophy (1979) at the University of Münster, he was Heisenberg scholar (1980), Professor for Foundations and History of Mathematical Science (1981-1988), and Vicepresident of the University of Konstanz (1985-1988). Since 1988 he is full professor for Philosophy of Science, director of the Institute of Philosophy and director of the Institute of Interdisciplinary Computer Science at the University of Augsburg. He is also professor of the Bavarian Elite Academy (Munich) and member of several interdisciplinary institutions (e.a. Gottlieb Daimler and Karl Benz Foundation). Since 1996 he is president of the German Society for Complex Systems and Nonlinear Dynamics.
He wrote **books** on "History of Geometry" (B.I.: German edition 1980), "Symmetries of Nature" (De Gruyter: German edition 1988, English translation 1996); "Thinking in Complexity. The Complex Dynamics of Matter, Mind, and Mankind" (Springer: English edition 1994, 3rd English edition 1997,

Japanese translation 1997, Chinese translation 1999), "Computer - New Wings of the Mind?" (De Gruyter: German edition 1994, 2nd German edition 1995), "Time" (C.H. Beck: 1995, 3rd German edition 1999), "Matter" (C.H. Beck: German edition 1996, Chinese translation 2000), "Brain, Computer, Complexity" (Springer: German edition 1997), "Computer Networks and Virtual Reality" (Springer: German edition 1999), "Hawking" (Herder: German edition 2000).
He also **edited** various books on „Numbers" (Springer: German edition 1983, 3rd English edition 1990, Japanese translation 1992, French translation 1999); „Philosophy and Physics of Space-Time" (Spektrum: 1988, 2nd German edition 1994), "The Beginning of the Universe" (C.H. Beck: 1989, 2nd German edition 1990), "How many Lives has Schrödinger's Cat? Physics and Philosophy of Quantum Mechanics" (Spektrum: 1990, 2nd German edition 1996), "The Question of Life" (Piper: German edition 1990), "Natural Sciences and Humanities" (Springer: German edition 1990), "Economy and Ecology" (P. Haupt: Swiss edition 1993), "Quanta, Chaos, and Demons. Epistemic Aspects of Modern Physics" (B.I.: German edition 1994), "From Simplicity to Complexity" (Vieweg: English edition 1998), "Complex Systems and Nonlinear Dynamics in Nature and Society" (Springer: German edition 1999).