Computer

Were it only for its number crunching ability, the computer would be no more than a better abacus. From a semiotic perspective, such a device would be relevant in regard to the possibility to automate operations corresponding to a well-defined category of signs, i.e., representing, in some way, quantities. That quantities can be represented iconically, indexically, or symbolically is a matter of cultural record. So is the fact that there are many number systems (e.g., binary, decimal, hexadecimal). Nevertheless, the type of representation (which extends to number systems also) is telling not so much of history as it is relevant for defining the ever-changing cognitive condition of the human being. Some representations, such as visual descriptions, are very close to the represented; others, such as mathematical or logical notation, tend to become more general or to reach increasing levels of abstraction.

One important thing should be pointed out from the start: the computer is a machine of a different condition than all previously known apparati. This condition is its semiotic nature. And in this respect, the literature dedicated to the semiotics of machines, including computers or not, is relevant only as a historic reference.

Many types of devices dedicated to operations involving quantities have been built over time. They were supposed to accommodate the ever-growing need for calculations posed by practical tasks of augmented complexity, or by theoretic endeavors. Such devices embody, literally, human knowledge of numbers: at the end of the nineteenth century in England, the word computer applied to people who carried out astronomical calculations as a profession. They also embody knowledge of proportions and arithmetic rules for addition and subtraction, in the sense that their physicality (how they are made) and their functionality (what they perform) are interdependent.

Once the notion of a machine is introduced, regardless of whether it is a theoretic device or an apparatus, a semiotic foundation is set. Whatever takes place between the machine and the user is ultimately of a semiotic nature. In other words, a peculiar type of sign process originates within the human praxis of conceiving, building, and using machines. In the literature of semiotics, this type of sign process is named *machine semiosis*. It attracted the attention of a number of semioticians (e.g., Andersen, Eco, Nadin, Rossi-Landi); and it inspired the suggestive activity of semiotic engineering (cf. de Souza), to which we shall return.

The abstraction of a machine, which was inspired by the tools and apparati of the Industrial Age, and whose logic and mathematics advanced as these became more abstract, fundamentally changed this limitation. One can say that before the first computer was ever built, computers had already been conceived of as theoretical machines able to process on a symbolic level. The Turing Machine, the archetype of the modern computer, is such a theoretical construct, but by no means the only one.

On the object level, and in respect to which signs are constituted, at least three aspects of semiotic relevance can be defined in regard to computers: computation, i.e., the use of computers for particular purposes of direct or indirect semiotic relevance; interface, i.e., the interaction between persons and computers; networking, i.e., the integration of machines and programs in an

underlying structure facilitating human interaction.

On a meta-level, semiotically relevant aspects pertain to: representation, i.e., the optimal forms of embodying information (e.g., as continuous signals, as digital simuli, as samples); understanding, i.e., the semantics of computation (e.g., what stands for what); and learning or self-awareness, i.e., reaching levels of self-organization that makes life-like properties (adaptation, self-learning, awareness, self-criticism, immunity, etc.) possible. Obviously, these lists are not exclusive.

Computers are basically semiotic engines. A formal proof and a discussion of the implications of such a statement were already given (Nadin, 1977), using the correspondence between the Peircean definition of sign and the definition of machines (the Turing machine, in particular). Regardless of their implementation, as analog or digital computers, or as devices working in any number system, computers process symbols at some level of abstraction dependent upon their condition (e.g., sequential, parallel, neural networks). Embodied in a computer are a logic (Boolean in current computer technology), data (today almost exclusively in digital format), and instructions (in the form of programs). Semiotically, this meeting of logic, signs (the digital system as a substratum), and operations can be described as computer semiosis, i.e., a particular kind of machine semiosis, in principle an open-ended process. The logic, defined for a binary universe, is embedded in the hardware and reflected in the structure and function of programming languages. Programs can also be – at least partially – embedded in matter (in particular, in the silicon chip), although they are written as "applications" and provided in the form of instructions. The data upon which logic is applied and operations are carried out, stands for measurements, thoughts, emotions, etc. As a semiotic engine, the computer is fed by the inexhaustible energy of sign-based human activities, interactions included. That the binary system does not exhaust all computational possibilities should not go unnoticed. Over time, attempts have been made to seek alternatives. (Peirce's assertion on the triad inspired several attempts towards building triadic machines.)

Such thoughts, of an analytical nature and sometimes of cultural import, could not shed much light on the subject of computers and computation were they only yet another attempt to capture for semiotics a field of inquiry and practical application to which semiotics itself contributed little or nothing. But the intellectual history of computers does not start with Boole, binary numbers, or silicon, as it does not prove relevant in the exclusive terminology of high-speed chips, fast input and output devices, large memory, or even multi-threaded operating systems. It was the magnificent semiotic project of Leibniz that articulated the goal of a *calculus ratiocinator;* many other projects, bearing the names of Raymond Lull, Pascal, Napier, and others, prepared the stage for the applications pursued by Babbage and embodied in his mechanical contraption called the *Analytical Engine.* Charles Sanders Peirce, himself probably involved in the attempt to build devices able to support sign processes (cf. *Charles Sanders Peirce and Marquand's Logical Machines,* 1984), discussed the semiotic significance of such attempts, suggesting through the operations of insertion, omission, and substitution a complete set for performing any sign calculus. Joseph-Marie Jacquard (1754-1834), Herman Hollerith (1860-1929), Howard Aiken (1900-1973). J.W. Mauchly (1907-1980), Konrad Zuse, John Atanasoff, and others constructed machines. Yet others, such as Norbert Wiener, Herbert Simon, Allan Newell, Vannevar Bush, Marvin Minsky (to name only a few) gave them their underlying

semiotic identity – even if not always aware of semiotics or taken with its broad perspective. In the process of dedicating a great deal of effort to designing languages suitable for programming, computer science appropriated the convenient semiotic distinction of syntax, semantics, pragmatics.

Broad statements that reaffirm the identifier *semiotic machine* for the computer, are only marginally productive (mainly, in a cultural sense) if they are not further substantiated. In mentioning various contraptions celebrated as precursors of modern computers, we have already formulated the principle of the progressive semioticity characteristic of these semiotic machines. Let us be more precise: Initial machines that qualify as computers process signs that are very close to the objects they stand for. The program embodied in an abacus extracts all that is known about arithmetic and ensures effective procedures as long as the data does not transcend a certain level of complexity. The slide rule embodies more programs, but requires familiarity with its sign operations – it requires learning, in other words. As program and machine become distinct, the semiotic condition is captured at the various levels at which programming is possible: machine language (binary language), assembler, compiler, etc. The semiotic condition of the process of assigning addresses (e.g., storage cells, registers) is different from that of defining operations at a more general level. The higher the semiotic level, that is, the higher the degree of generality of the sign processes in a computer, the higher the efficiency of the programming effort and of computing. In a way, this is best expressed by saying that a good word processing program is the sum total of all typewriters in every language known, and for any human activity that leads to communication through written texts.

In order to achieve this level of semioticity, layer upon layer of signs are introduced and embodied in hardware. In parallel, high-level descriptions of what it takes to initiate and successfully pursue a semiotic process are produced in the form of algorithms. The assumption is that everything can be reduced to an algorithm – a thought in the meanwhile abandoned since certain sign processes are not reducible to a finite sequence of sign operations.
In retrospect, technological progress has been driven by expectations of efficiency and by the advancement of science; but it was economically and socially justified by semiotic considerations. This means, in effect, that a fast semiotic machine is justified only to the extent that it supports meaningful semiotic processes. Again, this has to be explained. As number crunchers, computers operate at low levels of semioticity. But once the requirements of number crunching (the brute force approach) are transcended – for instance, in visualization applications such as representation of data aimed at extracting knowledge above and beyond that afforded by numbers and calculations – the need for faster processing, as well as for better representation of images justifies the effort to design chips capable of high performance and of displays for graphics. In parallel, languages appropriate to programming increasingly complex operations embody semiotic principles pertinent to forms of representation of increased semioticity. The progressive semioticity principle introduced above pertains to all aspects of computation: chip design, storage media, memory (random access, read-only-memory, dynamic, and other types), input and output devices, networking, browsers, and whatever else might emerge as computation matures and becomes integrated in human practical experiences. The various semiotic considerations pertinent to these components are integrated in the semioticity of each given type of computation.

In the model of human-machine interaction, this results in what are known as *interfaces*. Within computational experiences, such as those known as virtual reality or computer-enhanced reality, the interface as semiotic entity is transcended by the hybrid dynamic entity that can be defined as semiotic integration. With the advent of bio-computing (which focuses on the use of living matter for computational purposes), this integration will support even higher levels of semioticity. Thus, it will become more difficult to distinguish between machine- and human-based sign processes.

Frieder Nake, H. Zemanek, R.W. Floyd, M. Nadin, de Souza, and recently a group of semiotic engineering companies have approached the issues of computer programming and program evaluation from a semiotic perspective. What characterizes this work is the application of semiotic concepts and methods to a design domain still in search of its own identity.
Albeit, not too many computer scientists, and even fewer semioticians, have recognized the need for integrating semiotic considerations in the current dynamics of technological change. They are aware that semiotic considerations have proved very useful in approaching the fundamental problem of interface (the iconic interface is the better-known example of this application), and in the design of computer-supported interactions, such as the ones pertinent to the networked world. Vannevar Bush (1890-1974) will probably be remembered less for his technological genius, embodied in the Differential Analyzer (an analog computer built in 1930 at MIT), and more for his anticipation of non-linear thinking applications (*As We May Think*, 1947), such as those embodied in the World Wide Web. Semiotic engineering, to revisit the subject, is in the process of gaining well deserved legitimacy.

Indeed, to program means to master a science. But programmers are the first to acknowledge that programming is far from being a precise science with rules that, when applied, will have a predictable outcome. The mirage of automatic programming, at time resuscitated with renewed optimism based on lack of semiotic awareness, testifies to a condition of programming that is at least as much art as science. The design of user interfaces involves two particular communication acts: between the designer and the user, and between the system and the user (de Souza, 1993). Semiotic principles (originating in Eco's work, 1976) are used in engineering the message that identify the functions of a certain software package. More recently, the design of software itself has been subjected to semiotic considerations. But here the difficulties are different in nature since programming itself is an activity that defies easy definition.

The semiotic machine, like all sign-based human activities, is subject to rules, but also to creativity. Were programming only a matter of reproducing – very quickly and on a very broad scale – previous sign processes, it would not imply creativity. But the use of computers contributes to the expansion of knowledge and epistemology, resulting in a rapid increase of the variety and breadth of sign processes. This makes the whole issue of programming very complex, and also explains why the self-satisfying analytical semiotic approach is not matched by anything comparable in semiotically based computer synthesis. Programming languages incorporate premises that make impossible an authentic synthesis, independent of their syntactic limitations or of their limited semantic scope. It is, however, conceivable that this implicit limitation will be overcome. Visual programming is one promising avenue.

Computation, as a semiotic substratum and supported by semiotic considerations, is only part of the effort to deal with issues of human intelligence, virtual reality, and artificial reality. Semiotics has little, if anything, to contribute to implementations of neural networks, algorithmic and non-algorithmic computation, the technology of parallel processing, and similar issues. The significance of semiotics becomes apparent in addressing notions of appropriateness – which signs optimally support a certain human endeavor – of distinction – which features and which correlations support a certain process, e.g., pattern recognition, image understanding – and of integration – how a multimedia expression should be designed and implemented. Multimedia, which unite various data types is a computational challenge. But it is even more a semiotic experience of a type different from that embodied in the processing of single homogeneous data types.

What generally qualifies the semiotic approach is dedication to the entire effort of computation, that is, the commitment to ensure the coherence of the integrated sign processes facilitated and carried through computationally (in sequential machines, parallel machines, neural networks, or any other form of computation). A good interface will never automatically guarantee the success of a program. A good program (as relative as this is in a world of rapidly succeeding versions) with difficult interactions will perform only at a percentage of its potential. A coherent integrated semiotic strategy extends to everything that supports and defines the activity. In some way, such a semiotic strategy is the meta-program that unites software, data flow, input/output performance, connectivity, process and human interface, cultural and social acceptance, and learning. In his dissertation on computer semiotics (1990), Peter Bogh Andersen defines it as "the discipline that studies the nature and use of computer-based signs." He also works in the direction of defining computers as media, i.e., "carriers of meaning." The many implications, in particular social and cultural, of the increased use of computers will preoccupy semioticians in the years to come.

As a still young technology, in a phase of rudimentary evolution, computers maintain the semiotic engine on an exogen level, still detached from the application at hand, instead of becoming part of it. The challenge probably lies in the integration of computers in human pragmatics, making them appear as an extension of human intellect and skill. At that level, the semiotic engine should display awareness of the sign processes and should be able to initiate semioses appropriate to the goal pursued. The level of self-awareness, probably in conjunction with reaching levels of autopoiesis, is appropriate to semiotic synthesis. Autopoietic systems (i.e., systems that reproduce themselves) are not necessarily characterized by self-awareness, but they can emerge as self-aware. Recognition of signs and of sign processes would be the logical consequence of such a development. Nothing should prevent us from trying to achieve such a status in machines that are not only driven by a semiotic engine, but can also actually account for their morpho-genesis.

Bibliography

Andersen, Peter Bogh.

Semiotics and informatics: computers as media. In, *Information Technology and Information Use. Towards a Unified View of Information and Information Technology* (P. Ingwersen, L. Kajberg, A. Mark Pejtersen, Eds.). London: Taylor, Graham, 1986, pp. 64-97.

Design and professional languages (with K. Halskov Madsen). In *Computers and Language at Work* (P.B. Andersen, T. Bratteteig, Eds.). Oslo: Oslo Universitet, 1989, pp. 117-156.

A Theory of Computer Semiotics. Semiotic Approaches to Construction and Assessment of Computer Systems (doctoral dissertation). Cambridge: Cambridge University Press, 1990.

The force dynamics of interactive systems. Towards a computer semiotics. In *Semiotica* 103, 1/2, 1995, pp. 5-45.

Machine semiosis (with P. Hasle and P. A. Brandt). In *Semiotics: A Handbook About the Sign-Theoretic Foundations of Nature and Culture,* vol. 1 (R. Posner, K. Robering, T.A. Sebeok, Eds.). Berlin: Walter de Gruyter, 1997, pp. 548-570.

Gorn, S. The identification of the computer and information sciences: their fundamental semiotic concepts and relationships. In *Foundations of Language*, 4, 1968, pp. 339-372.

Ketner, Kenneth Laine and Stewart, Arthur W. The Early History of Computer Design, Charles Sanders Peirce and Marquand's Logical Machines, *The Princeton University Library Chronicle*, vol. XLV, no. 3, spring 1984, pp. 187-222.

Nadin, Mihai.

The repertory of signs. In *Semiosis*, Heft 1, 1976, pp. 29-35.

Sign and fuzzy automata. In *Cahiers de Linguistique Théorique et Appliquée*, XIV, 1, 1977.

Interface design and evaluation. In *Advances in Human-Computer Interaction*, vol. 2 (R. Hartson, D. Hix, Eds.). Norwood NJ: Ablex Publishing Corp., 1988.

The bearable unbearability of the rational MIND, *Ästhetik, Semiotik, Informatik* (F. Nake, Ed.). Baden-Baden: Agis Verlag, 1993, pp. 63-102.

Interface design: a semiotic paradigm, *Semiotica* 60, 3/4. Amsterdam: Mouton de Gruyter, 1988, pp. 269-302.

The best computer is invisible, *Computer Art Faszination*. Frankfurt/Main: Dr. Dotzler Media Institute, 1996, pp. 210-212.

Nake, Frieder.

A proposed language for the definition of arbitrary two-dimensional signs, *Pattern Recognition in Biological and Technical Systems* (O.J. Grüsser, R. Klinke, Eds.). Berlin/Heidelberg/New York: Springer Verlag, 1971, pp. 396-402.

Ästhetik als Informationsverarbeitung. Grundlagen und Anwendungen der Informatik im Bereich Ästhetischer Produktion und Kritik. Vienna/New York: Springer Verlag, 1974.

Human-computer interaction: signs and signals interfacing, *Languages of Design*, 2, 1994, pp. 193-205.

Zeichen und Gebrauchswert. Beiträge zur Maschinisierung von Kopfarbeit (F. Nake, Ed.). Bremen: Universität Bremen, FB Mathematik/Informatik, Bericht Nr. 6/94, 1994.

Der semiotische Charakter der informatischen Gegenstände. Beitrag zur *Festschrift für Elisabeth Walther zum 75. Geburtstag* (U. Bayer, Ed.). Baden-Baden: Agis Verlag, 1997.

Peirce, Charles Sanders. Logical Machines, *The American Journal of Psychology*, 1(1887), pp. 165-170.

Souza, Clarisse Sieckenius de.

The semiotic engineering of user interface languages, *International Journal of Man-Machine Studies*, No. 39, 1993, pp. 753-773.

Supporting end-user programming with explanatory discourse, *Proceedings of the ISAS.* Gaithersburg MD: NIST, 1997.

Stamper, R.

Information in Business and Administrative Systems. London: Batsford, 1973.

The semiotic framework for Information systems research, *Information Systems Research: Contemporary Approaches and Emergent Traditions* (H.E. Nissen, H. Klein, H Hirschheim, Eds.). Amsterdam, 1991, pp. 515-528.

Signs, organizations, norms, and information systems, *Proceedings of the Third Australian Conference on Information Systems.* Department of Business Systems, University of Wollongong, Australia, 1992.

Zemanek, H. Semiotics and programming languages, *Communications of the ACM*, 9/3, 1966, pp. 139-143.