What Kind of Sign Is A Bit?

(Lecture: Semiotics and the Information Sciences, Victoria College, University of Toronto, 13-15 October 1998)

Computation comes in many forms: digital, analog, algorithmic, non-algorithmic, serial (von Neumann's paradigm), parallel, interactive, numeric, symbolic, centralized, distributed. To leave these distinctions to scientists and engineers and to focus exclusively on the outcome of computation is probably appropriate as long as one positions himself or herself in the now established role of user. It should be remarked from the outset that 80% of what is defined as computation concerns users. Word-processing is the user application that leads by far; but desktop publishing (which involves text, layout, and computer graphics), database applications (from pre-programmed tax return calculations to keeping records such as addresses, recipes, financial information), and more recently networking (e-mail, Web presence/Websites, Web-publication, remote teaching, cooperative projects, and so much more) make up an increasing complementary set of applications. Some of these applications assume a user different from the one limited to word-processing, but in the end still not a computation professional. Embedded computation (or ubiquitous computing) effectively overwrites the role of the user.

Again, one would be better off leaving a comprehensive evaluation of these particular applications in the hands of those who invented them, since for better or worse, all users have to say is that one or the other program still does not work as well as expected, or that the price-performance ratio is in some cases better than in others. Allowing myself a rather innocuous exaggeration, I can say that computation users are merely the most cost-efficient quality control agents ("debuggers") of a very interesting science and technology that the term computation denotes, but by no means describes. Ideally, computation is an expression of knowledge (in the forms of algorithms, processing procedures, interactions, programs, etc.) subjected to a wide variety of tests. It embodies the positivist expectation of validity, effectively erasing the distinction between science and humanities. It claims universality and is, together with its twin sibling genetics, constitutive of an epistemological horizon of unprecedented characteristics. For reasons well beyond the scope of this paper, I have dedicated quite some effort to understanding the relation between computers and signs. This effort has allowed me initially (Nadin, 1977) to establish the correlation between computation and sign processes. My early mathematical, philosophic, semiotic, and computational effort resulted in proving the equivalence between automata and sign processes (in Peirce's definition):

$$S = S(R, O, I, o, i), \text{ which is equivalent to } A = A(X, Y, Q, \alpha, \beta)$$

in which S stands for sign processes resulting from the open relation among objects, *representamina*, and the *interpretant* process; A stands for automata processes; X and Y, respectively, for the signs of input and output; and Q for the set of states. The transition function and the output function describe how output is generated from a certain input. Every automaton is a generative semiotics.

Once the equivalence was proven, it has henceforth justified the introduction of a notion many times quoted, but never really understood: The computer is a semiotic engine. The proof, however, goes back to only one of the many types of computation mentioned above in the first sentence. Moreover, it relates one of the concepts of the sign – the one based on the triadic-trichotomic distinction implicit in Peirce's semiotics – to what can be called the dominant form of computation (the von Neumann serial computer. If my assertion that the computer is a semiotic engine should be of any consequence for both semiotics and computer science, the initial limitations of the proof need to be overcome. Moreover, the consequences of my statement should become clear, if indeed there are consequences to be expected beyond giving semiotics that much needed boost of credibility without which its future relevance outside academic endeavors remains, as always, doubtful. Let us address these two requirements, not only for the sake of addressing them – intellectual goals often end up becoming relevant in themselves, but of no consequence for anything else – but foremost because if they can be clearly pursued, neither semiotics nor computer science will remain the same. This assertion is, I know, of a tall order and poses many challenges to those interested in and willing to pursue its consequences.

Computation is knowledge

Regardless of what kind of computation we consider, there is one characteristic that they all share: the outcome is an expression of something that could not be explicitly identified before the process took place. All the ingredients in the process – data, instructions, memory management, process and user interface – can be described in detail, and still the outcome cannot be predicted. What matters is the process. Therefore, to compute means to design a type of processes fundamentally different from those we are familiar with from physics, chemistry, biology, and other sciences. Computation can unfold on virtual or on real machines, in machine-based time or in (almost) real time, in single or multiprocessing sequences. What counts is its inherent dynamic condition and the fact that knowledge is generated at the intersection between human cognition and machine-based cognitive functions.

This knowledge can be of various kinds, like human knowledge itself. To be more specific: word-processing is the knowledge of all the elements involved in generating and disseminating texts. It is, in the first place, a comprehensive theory of all the variables involved in the human or machine experience of generating texts in a context of acknowledged rules that embody grammar, syntax, etymology, linguistics, as well as rules for structuring and presenting ideas in written form. This theory, still in the making, is embodied in particular programs that allow for spell-checking, for instance, or stylistic refinement, or for various visual forms of structuring (through layout rules, for instance). Its use is neither more no less than the test of the text knowledge embodied in the model of a specific computational word-processing implementation. As people use this knowledge, they test it beyond everything a particular person or group (developers) could even imagine.

However, at this moment in the development of computation – a relatively young discipline, whose main products are still rudimentary – knowledge generated in computation processes is predominantly acknowledged outside the process, i.e., in the interaction between human beings and the machines supporting these processes. In other words, like the abacus, the computer does

not know right from wrong, and even less, significant from insignificant, meaningful from meaningless.

Instead of revisiting the formal descriptions of the various types of computation known so far (many more will come, if we consider the extraordinary multiplication of means and methods dedicated to computation), and inferring from such descriptions to sign processes (in Peirce's sense, or in some alternative fundamental concepts of semiosis) and vice versa, I prefer an alternative path. Under the assumption that computation is knowledge pertinent to a new moment in the evolution of the species, and in the knowledge that there are no known cognitive processes whose underlying principle is not semiotic (on this topic, many authors could and should be quoted), it follows that the statement, "The computer is a semiotic engine," needs not to be further formally proven, since it is the necessary consequence of the condition of computation. Granted, the assertion might be weakened if someone could come up with a type of computation that is not knowledge-based, but even if one could produce such an example, it would not automatically exclude semiotic processes, but rather prompt more adequate definitions of what we call *semiosis*.

The point I am trying to make is far from trivial. Many scientists, technologists, semioticians – you name them – consider computers a technology, and what happens in a computer, a matter of moving electrons, heat dissipation, and electromagnetism, i.e., physical processes. They are not totally wrong. After all, computation as process does not happen in a vacuum (after the disappearance of vacuum tubes, this sentence holds true even in the literal sense), but with the participation of matter (organic or anorganic), or better yet, at the meeting point between matter and human cognitive capabilities.

In one of his famous statements (probably quoted as frequently as his theory of relativity), Einstein declared: 'It would be possible to describe everything scientifically, but it would make no sense. It would be without meaning, as if you described a Beethoven symphony as a variation of wave pressure.' Up to a certain point, to which I shall return, he was right. Indeed, electronics – the science and technology of all that made computers possible – is a necessary but no sufficient condition for computation. All the circuits can be perfectly designed and produced, the power supply in good order, and the input and output devices correctly integrated, and still there would be no computation at this stage. Something else, of a higher order (if we agree to accept that abstraction is of a higher order than the concreteness of matter) makes the function of computation possible. Alternatively, a situation in which we have no machine whatsoever but in which we conceive a program and execute it mentally or on paper (granted, slowly, step-by-step, with many intermediate steps) can be seen as computation, insofar it is part of a cognitive process involving a representation, a logic, data, and instructions applied to them. Again, "no machines whatsoever" does not mean that the biological machine – to use the old machine metaphor – that we humans are is not the substratum of the process. The Turing machine is an example, and my older demonstration (Nadin, 1977) that the mathematical category describing it is equivalent to the mathematical category describing sign processes only confirms why one can claim that the engine of computation is semiotic.

Where Einstein is off the mark, or to put it more exactly, where Einstein could not foresee that there is a mapping that could as well describe a Beethoven symphony while preserving its unique

nature, is where the subject becomes interesting, and of extreme semiotic importance. Leibniz, whom Newton shadowed by using a better semiotic device for expressing the same mathematical theory when he also discovered differential calculus, thought that such a description was possible. He was obsessed with it; and my claim that Leibniz is the forefather of computation is based on the fascinating contributions he made to a universal language and the applications he delivered. But let's not get stuck in history, or in vain disputes on who did what and when. Our subject is computation, not only as process, but as semiotic process. The qualifier semiotic means that a sequence of interpretations is generated in each and every computation. By this I mean much more than permutations, even more than tractability, that is, whether, in order to compute, one transcends the time limitations by which we humans live (finite intervals).

Having ascertained this, here is a good place to draw attention to the fact that algorithmic computation and interaction computation (two of the many types of computation mentioned) are fundamentally different. If the difference were only scientifically significant, we would be ill-advised to spend our energy in even making the distinction. For reasons yet to be disclosed and discussed, this distinction is semiotic in the first place, and consequently it affects the functioning of the semiotic engine that constitutes the underlying foundation of any form of computation.

If computation, regardless of its nature, is not reducible to electric processes but involves semiotic entities, the question is: What are they? A short answer would be: The same entities that make cognitive processes possible. Somewhere along the line, we end up at the one and only culprit of semiotics – the sign. Thus, we close the infamous circle:

The sign as an underlying element of thinking = The sign as a product of thinking.

Computation has it easier. Bits and bytes (which are only strung-together bits) are processed but not necessarily defined through computation; rather, they are defined beforehand, as a condition of computation. This is the meaning of the question posed in the title of this paper. Can we consider computation a subset of the encompassing set defined as semiotic processes, more precisely, the subset dedicated to one precise sign, the bit, defined outside computation (in information science, as the elementary unit of information)? Or is the bit only a provisional name for the various signs finally subject to computation?

A closer look at the bit.

As a measure of information, the bit describes quantities. As a unity among what is represented, the representational means, and the infinite process of interpretation, the sign emerges as individuals constitute themselves through whatever they do. The bit itself was generated in such an experience of generating, transmitting, and receiving information. As a sign, the bit can be seen at the syntactic level as the string of letters b, i, t, or as whatever the syntax of the information it embodies is; at the semantic level, as the univocally defined unit of information pertinent to the simplest imaginable choice (heads or tails); at the pragmatic level, as the relation between the information it describes, the many ways in which it can be expressed, and the infinity of actions it can trigger, or, alternatively, inhibit. Insistence on clarifying concepts at this juncture stems not from a pedantic instinct, but from a pragmatic necessity: If the relevance of

semiotics to computation is to be established, then it is obvious that one more analytical tool will be exactly what other analytical tools are, i.e., maybe an instrument of validation, a method for evaluating, or, at best, an optimizing procedure. There is nothing against such possibilities, which, as we know, semioticians took advantage of, producing lectures, articles, even books about them. However, the nature of computation is such that semiotics belongs to its premises, and accordingly, a legitimate semiotic approach can and should be part of the computation, not only of its validation after it was finished.

In more detail, what this means is nothing else than the rethinking of computation in semiotic terms, and their effective integration in the means and methods through which knowledge is computationally expressed. That involves transcending the quantitative level of the bit and the integration of qualitative signs, with the implicit understanding that quality is not reducible to quantity. This major understanding is far from being trivial, especially in a context of technological innovation within which some – I repeat, some – aspects of qualitative distinctions were successfully translated into quantitative distinctions. Point in case: music. Thus, Einstein's assertion on representing Beethoven digitally comes back to haunt us. Indeed, the high generality of the bit, as opposed to the concreteness of wave pressure differences, explains the perfect digital rendition of a Beethoven symphony, without, of course, making it identical or equivalent to a live performance (in a studio or before an audience). We can even imagine an automated performance, by virtual musicians, directed by a virtual conductor, faithful to Beethoven's musical text to any extremes we can think of. But that again is Beethoven as quantity, measurable and controllable, while a performance, with its implicit deviations, results as a living product and ceases in this definition once the performance has taken place. This is no an elaboration on music, or on the arts. It is an elaboration on what happens when the semiotic engine human being is replaced or complemented by a semiotic engine of a different nature. Feigenbaum's confessions to calculations he performed in his mind and which resulted in valid outcomes different from that of computation by powerful computers is but one example of how the means of representation are not a passive constituent of the semiotic processes in which they are used.

Semiotics brings to computation the awareness of the fact that sign processes depend on the nature of the signs, that they are constitutive of new realities, and as such, not unlike notation systems (e.g., numbers, letters, colors, shapes), they are present not only in the input (what goes into a sign process, what goes into computation), but also in the output. A digital rendition of a Beethoven symphony could be as fascinating as any other we can think of, provided that it can make possible the closure through which representamina are integrated in an interpretant process. Circumstances for this to happen are provided. We experience a fundamentally new pragmatic framework, i.e., that of semiotically driven human experiences. Indeed, the species moved another notch away from its natural condition to its human, i.e., semiotic condition. To elaborate on this here would be presumptuous, to say the least. My book (Nadin, 1997), a work of well over 15 years, describes in detail the process through which human beings arrived at this juncture. That this paper builds on the body of hypotheses and inferences set forth in the book should not surprise. What I attempt here is to take my inquiry further and, in particular, to see how computation can be grounded not only in electronics, logic, algorithms, mathematics, etc., but can also integrate the enormous semiotic experience that the species has acquired so far.

Computation as semiosis

To nobody's surprise, semiotic considerations in respect to computation were first articulated in respect to so-called man-machine interaction. Considerable experience originating from past challenges posed by all kinds of artifacts used by individuals was brought to table. Even line editors, those precursors of the current interfaces, were subjected to semiotic scrutiny. Commands had to be abbreviated, made as clear and univocal as possible, presented in legible form, and according to cognitive principles pertinent to the human processing of words. But this is prehistory, even for those active in the new computer museum business. Iconic interface was a definite semiotic statement, inspired, as we know, by semiotic terminology. To its fame, and to its shame, semiotics contributed to the desktop metaphor – a huge step forward in making new forms of computation available to a large number of users, but also a dead-end street in which computation has remained stuck to our day.

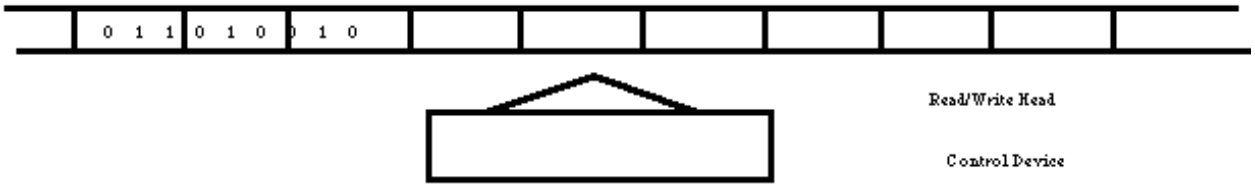
Much more interesting was the attempt to enlarge the notion of computation itself to include varieties of signs extending from those elements making up the elusive domains of the visual, the aural, and multimedia. In the virtual realm, much more than in the pseudo-3D realm, all kinds of semiotic devices found their usefulness in, or contributed to, the periodical moments of confusion that mires computation. To a lesser extent, semiotic considerations were present in neural networking, biocomputing, molecular and quantum computation, to name a few. But it remains to be seen whether this situation will eventually change. In some areas, extremely intricate semiotic considerations are a dominant, though rarely identified as such, component. Data-mining, the magic formula of the networked computation dedicated to the use of information leading to more individualized forms of interaction (dissemination of the new, e-commerce, healthcare, culture, etc.) is, after all, the embodiment of abductions in the strictest sense of Peirce's definition. Almost all known inference engines deployed today encode semiotic elements, although at times, those who designed them are rather driven by semiotic intuition than by semiotic knowledge.

To start a search on the Web today is to literally start sign processes, to either watch how these unfold or to affect their unfolding by controlling the syntax level, the semantic involved – still the dominant dimension of any Web activity, or the pragmatics – in cooperative projects, remote learning, and interactive publishing. These forms of computation as semiosis will continue to attract more and more people. Their efficiency can be improved only if more methodical, and more professional, semiotic elements will be integrated and fine tuned in their use. But such and similar considerations can be generated *ad libitum*.

Let us return to the distinction algorithmic computation-interaction computation. Lynn Andrea Stein (1996), addressing professionals in computer science, claimed that the difference between the two can be understood by analyzing the beginning example in introductory classes. In algorithmic computation, you start with a question. Programming turns out to be the description of operations required to answer the question. "How can I put a message on the screen?" is such a question. Today the answer is, "Print 'Good morning!' " (this being a possible message). The program, which uses a facility describing printing (on paper, on a screen, on any other output device) performs the operation and exits. The interaction model leads to something else:

```
while (TRUE) {
echo ( ) ;
}
```

This first example is characteristic of a read/write device practically isolated from the outside world. Remember, the Turing machine consists of an infinite tape on which the values 0 and 1 are written, a reader and a control device



[Fig. 1]

As this theoretic machine computes, i.e., reads the values on the tape, it is disconnected from the external world. The sign processes that take place here actually correspond to an indexing procedure within a closed world. This is a single thread operation, and it is characteristic of the von Neumann paradigm of computation. Its semiotic engine functions on a very simple syntactic level – an alphabet of two signs – to which we can add, *afortiori,* a semantic dimension.

But once you reserve your flight tickets, or make your searches on the World-Wide Web, or build a robot, you are in a different world. This semiotic engine no longer supports the type of computation that makes such interactive tasks possible. Interactivity requires that the semiotic engine is connected to the world because out there, other sign processes extend those intrinsic to computation. Again, Lynn Andrea Stein provides a simple example:

```
while (TRUE) {
if (left_brighter) {
drive_left ( );
} else {
drive_right ( );
}
}
```

This is a good description for someone following a light – the semiotic interpretation of light in darkness is elementary to anyone conversant in semiotics – be this a human being or a robot (actually, these lines of code are for a robot using a differential photo-receptor). The process is multi-threaded (in this case, two threads control the orientation), open-ended, interactive. We are in a computation that the Turing machine no longer adequately describes. Important is the fact that we are in a dynamic context. Semioticians should look at these two examples of code not as a test of their computational skills, but as an expression of their fundamental concerns with sign processes.

As we know, semioses, regardless of their nature, are dynamic sign processes. Through semioses, minds interact, and thus become identified in a course of action (pragmatics) definitory of their characteristics (Nadin, 1991). As we move towards evolutionary computation, with evolvable hardware, we need to make sure that the semiotic engine on which they are by nature based is designed having in mind the requirements of semiotic processes as we know them from human interaction. That new classes of such semiotic processes might evolve is probably beyond dispute. However, as sign-based, they will reflect the epistemological nature of the sign, and thus replicate semiotic awareness. Indeed, a semiotic engine is not pure and simple an engine, but one with a certain self-awareness. The bits processed are bits that know where they are and to which string they belong. More precisely, the operation to which they belong is not mechanical, but semiotic, that is, with the mechanism of self-interpretation embedded in the process. When representations of digital circuits are placed at the level of the chromosome – as it takes place in our days – a foundation is laid for computation that involves and facilitates self-awareness.

But I do not refer only to the future. Those familiar with data transmission know that circuit switching, i.e., establishing a physical connection between two parties, was replaced by packet-switching. The latter is a simple operation of dividing messages into parts, attaching an address to each of them, and letting them find the most economic path to the receiver. There the parts are reassembled as the complete original message. The element of awareness that I mentioned is reduced to an address. Texts, speech, music, and images all travel in packets and are synchronized in order to carry whatever information is important for our interpretation, or for anything else we do with these complex semiotic entities.

An associative procedure

With all this in mind, one still wants to return to those issues of computation for which semiotic input is expected today, rather than in the future of high hopes and high expectations. In order to make such statements more concrete – as I did when discussing iconic interface and the opportunities that semiotics has in this area (Nadin, 1988) – allow me to shortly present a project (in progress as I write this text) dedicated to Interactive Multimedia Navigation (1997).

Within this project, we understand knowledge as an infinite sequence of associations between what we know and what we are in the process of acquiring as knowledge. The most important cognitive processes supporting knowledge acquisition are associations. How associations come into being is difficult to describe because association processes occur in open systems of heterogeneous semiotics. The basic assumption is that knowledge is expressed semiotically and further subject to processes of interpretation. Associations that have taken place can be documented, refined, or, if some turn out to be irrelevant, discarded. Associations are almost always multimedial, i.e., we associate texts, sounds, images, movement, etc. Implicitly, associations are multi-threaded. The structure of an interactive multimedia encyclopedia based on associations includes:

1. a knowledge space/domain, which can be described semiotically;
2. an associative search procedure;
3. a function for storing associative traces

In this particular project, knowledge space is implemented through the well-known metaphor of knowledge as a sphere. Within various spheres of knowledge, several perspectives are defined:

- historical axis (timeline), defining a diachronic semiotic dimension
- gnoseological axis (knowledge and paradigms of knowledge)
- cognitive axis (different kinds of thinking)
- aesthetic axis (aesthetic as a structuring feature of knowledge).

Our goal is to fashion a distributed system of knowledge through combinations of storage media and the World-Wide Web.

The associative search procedure allows each reader to pursue connections, which can be historical, gnoseological, cognitive, or aesthetic. Associations can be formal or they can concentrate on temporal, logical, or aesthetic criteria.

Association traces are saved by storing the associations that the reader pursues. They can be retrieved and visualized in order to provide the reader with the possibility to pursue the most promising trails.

Within our model, the static nature of CD-ROM publications in transcended because search and retrieval are documented in the background in order to facilitate further extraction of knowledge from available data. Our model belongs to the current research trend that is in the course of establishing itself under the label data-mining. The combination of storage media (CD-ROM, DVI, DVD, etc.) and World-Wide Web leads to a very dynamic structure. Semiotically, we allow for heterogeneous signs and a variety of composited operations. Starting with the fundamental semiotic operations (insertion, substitution, omission), we proceed in defining composited operations and making sure that the rules of operations remain clearly defined. The model supports learning not as a form of information consumption (how many bits processed), but as a form of knowledge generation – the assigning of meaning and understanding to the information processed. Learning is expressed through associative maps that document a process of understanding in contrast to learning by heart (memorization).

I refrain from details here because the purpose of this paper is rather to look deep inside computation in order to see at which level the bit becomes a semiotic entity that transcends its informational definition. The semiotic engine that is at the heart of any and all computation embodies our current semiotic awareness. Whether this engine could anticipate semiotic processes different from those we are aware of remains an open question. By advancing the notion of anticipation and self-configurations, I tend to answer it affirmatively.

References

Nadin, Mihai

(1977). Sign and Fuzzy Automata (Max Bense, Ed.). In Semiosis, Heft 1,5, 19-26.

(1988). Interface design: A semiotic paradigm. PDF In Semiotica 69-3/4, 269-302.

(1991). Mind – Anticipation and Chaos. Zurich/Stuttgart: Belser Presse.

(1997). The Civilization of Illiteracy. Dresden: Dresden University Press.
(1997) Navigation tools for dissemination of knowledge via interactive multimedia. Project of the German National Science Foundation (DFG).

Stein, Andrea (1996). Rethinking CS101: Or, How Robots Revolutionize Introductory Computer Programming.