Prof. Dr. Mihai Nadin

Artikel in: Signs and Systems, A semiotic introduction to Systems Design, Cambridge: University Press, erscheint im Sommer 1997

Signs and Systems

A relatively benign observation: The more powerful the technology gets, the more obvious it becomes that access to this power is crucial for its deployment. Useful integration of computation depends more upon the means of human interaction than upon chip speed, bandwidth, or packet switching. Still, in the race for doubling chip performance, the rate of 18 months (the famous Moore law) seems to obliterate the adjacent law: No machine knows what it can do!

At this juncture, dropping the word semiotics into the ring should by no means be construed as a declaration that a magic formula for... has been discovered. There is no magic, neither in moving data faster and on broader lanes, nor in reassessing the means we need for assuring meaningful interaction with them. Semiotics, embraced early on by designers of so-called “user friendly systems,” has a history of fascination with and distrust of signs that we’d better not ignore. Just as there was no magic alchemy formula for turning lead into gold, there is no magic semiotic formula for turning a piece of silicon into a thinking machine with which we can carry on dialogue. Disappointed alchemists of the Middle Ages started hating semiotics with probably the same intensity as disappointed computer scientists will if we do not set up a meaningful set of expectations. The difference—because ther is one—is that computer scientists are using semiotics, whether they know it or not.

The birth of a medium

In the the last 30 years, we have witnessed the birth of a new medium whose importance equals Gutenberg’s invention of movable type. The new medium is the computer, and this textbook aims to qualify the reader to work professionally with this medium.

What is the point of defining computers as media instead of machines? Whereas machines consume energy and aim at manipulating inanimate nature, most computer applications consume thoughts and feelings and aim at influencing living human beings. This is exactly what other media do. Viewing present-day computer applications as machines misses the main purpose of these systems, and therefore leads design solutions that are for the most part inappropriate.

A brief historical description of the birth of the computer medium is justified here by the need to put the semiotic perspective in the broader context of the changes that made the computer possible. Since a medium is, by definition, a carrier of signs, the birth of a medium can be described by following the development of its signs—in our case, computer-based signs. In a language-based model, the computer processes what Ferdinand de Saussure called signifiers.—that is, ... Structures are preserved through computation, and meaning is attached to them.

The defining rule is synchronism. In the logic-based model of signs, what is of importance is the process of sign replication as the triadic unity between an object, its representation, and the open, infinite series of its interpretations (Figure 1).

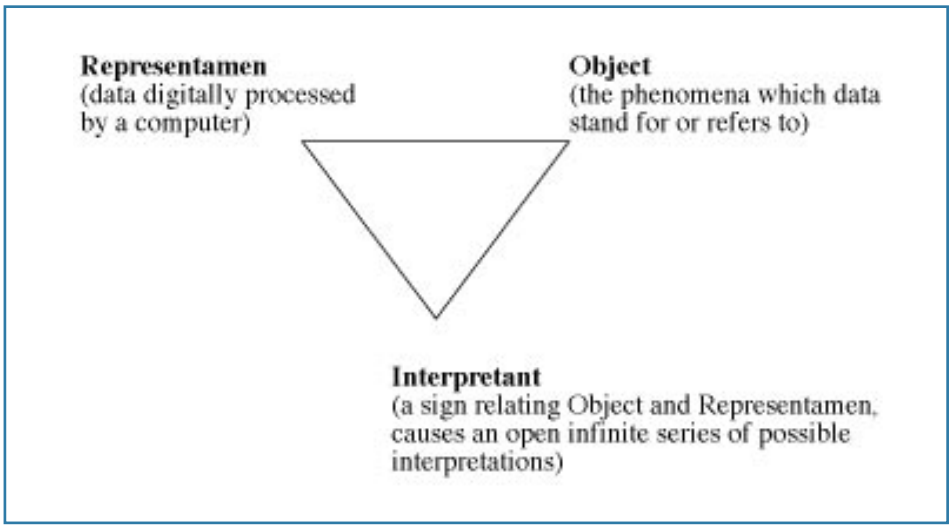


Fig. 1. Computer-based signs in Peirce's semiotic framework.

Signs must exist physically (the representamen); they stand for something other than themselves (their object); and the relation between representamen and object must be related by laws and conventions and cause reac-tions in the interpreter. Signs are a necessary, but not sufficient, part of communication. They are used by people or groups of people for managing and coordinating the affairs of daily life. Signs are means of social interaction.

In their infancy in the 40s, when technology was mostly geared towards number crunching, computer-based signs lacked most of the properties mentioned.They were signs of a very limited representation of the world; and once processed, they made possible only a limited interpretation.

The object: from machine reference to domain reference

Invention of higher programming languages, beginning with the construction of assemblers, caused the object of program texts to change from the physical machine to the domain of application. The object of the assembler code given below is the machine: registers, numbers, and the code itself ("jmp mloop" means that the program should return to the line named "mloop").

The sign process is probably not as expressive as that of natural language (in

```
mloop:movr      1,1 snc
                movr    0,0 skp
                addzr   2,0
                inc     3,3 szr
                jmp     mloop
```

which we could describe the same operation), but very precise.

In opposition to this, the next text, written in object-oriented notation, is in-terpretable as a description of the application domain of the program—in our example, a bank account:

```
class account
    owner: ref(person)
    amount: integer;
    procedure deposit.
    procedure withdraw.
end account
```

The sign process compresses in this notation a variety of activities that describe a transation. Whereas older programs referred to storage cells and registers, modern pro-grams can be read as assertions about wages, addresses, and positions. This was achieved by creating layers of signs within the systems. The lower levels are still about the machine, but the higher levels concern the application domain. Higher levels are translated into lower levels. Here we notice subsequent transations of signs from the object level to the machine level.

Thus, the layers we can identify from a synchronous perspective have a diachronous explanation. They are like geological sediments, the upper ones being younger than the lower ones (Figure 2).

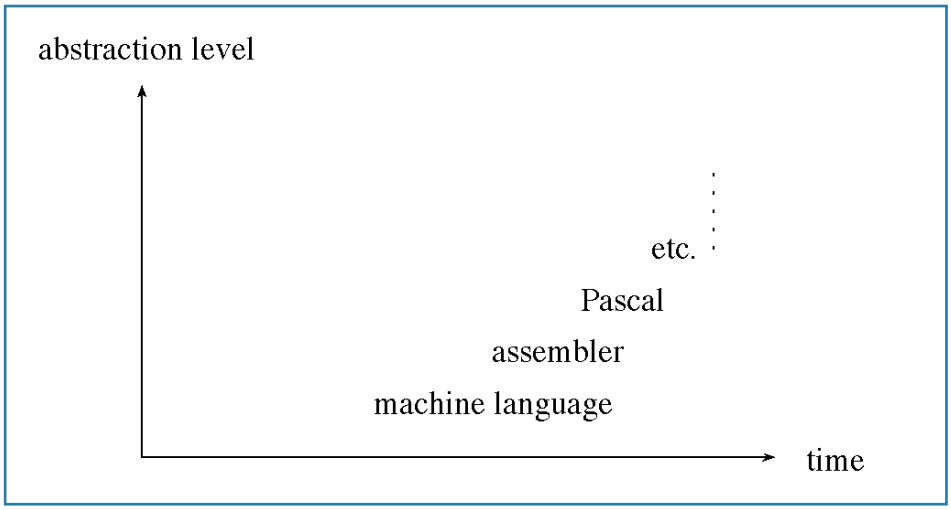


Fig. 2. the evolution from machine to domain reference.

The interpretant: from focus on data to information

By data is meant a formalized representamen of such nature that it can be communicated and transformed by means of a digital process. By information is meant the human interpretation of data, based on a convention. Thus, information in this sense seems to encompass the object, as well as the interpretant of the sign.

At the beginning of the present computer age, much effort was spent on designing

algorithms for producing the representamen, and less energy was devoted to ensuring that users were able to meaningfully interpret the results. Business needs knowledge, but data in itself does not give knowledge. The growing emphasis on object and interpretant means that the systems are required to “be about” relevant issues, and that the computer-based signs computer-based signs must elicit relevant responses in their context of use. However, awareness of this issue first emerged in the mid-80s.

Interactive use replaces batch-processing

The advent of interactive systems was caused by the need for updated information.This represented a step towards semiotic dynamics.

Early batch systems had a clean division between Input, Process, and Output. Input and Output were seen as passive manipulable objects that were transformed by an active, immutable program—a machine that consumed and produced data in the same manner as a factory consumed meat and produced canned beef. Users only saw the batch f cards with the input and the paper output. But with real-time interactive systems, the physical handling of the system became a decisive factor in the interpretative process. Even the input-output distinction is transcended.

In modern object-oriented programming, the basic division of a system is not between passive data opposed to an active algorithm. The basic building blocks are now objects consisting of coherent data and operations that can be handled by other objects or by users.

Only when this level is reached does the computer-based sign acquire its special characteristics that set it apart from all other known kinds of signs—its interactive features. Hypertext is a good example of this transition. Although prophesied by Vannevar Bush as early as 1945, hypertext first appeared for the public in the 80s and as Hypercard, it became a commercial product. Hypertext can be described as a computer-based ver-ion of paper that can live only in an electronic environment, adding interac-tion and action to the passive paper. The interactive features of hypertext, known as navigation, is one of the key issues in hypertext design and the main reason for bothering with hypertext at all. The marriage between hypertext and media led to hypermedia.

Enhancing the expressive power of the medium

In the 70s, the only code available in the computer medium was written language. However, during the 80s, new codes were systematically added to the semiotic repertoire of the medium: first graphics became available, progressing from black/white images via 8-bit pictures with 256 colors to high quality images with millions of colors. Sound was added and is now available in hi-fi quality comparable to records and CDs. The newest code is the filmic code, which, at least in its digital form, still needs some years of development in order to be aesthetically satisfying. The process of enhancing the medium’s expressive power is still in its infancy.

Invention of data-communication and local net

Like all other signs, computer-based signs need to be physically communicated in

order to reach their interpreters. But the digital messages had to stay within their plastic shells until the beginning of the 80s. Only with the advent of digital telephone networks and local area networks did computer-based signs come of age and were given the same traveling opportunities as their older relatives. Before that, computer-based signs were half-caste signs that had to be converted to other media—e.g., paper or tape—before being able to enter a communicative process. Packet switching, which is a frequently downplayed semiotic development, is of extreme importance for this process. The fact that semiotic coherence is preserved while the representamen is switched through packets is fundamental for the success of the endeavor.

In the first half of the 90s, the Internet exploded and its World Wide Web enbodiment introduced multimedia technology into the public networks. Overnight, this turned the Internet into a full-fledged mass-communicative medium that carries out many of the functions formerly performed only by television and newspapers.

Information technology becomes a general communication and management tool

This phase is probably one of the most important and difficult phases in the maturation of the computer-based sign,, a phase that began only 10 years ago and has not yet ended.

Basically, sign systems are socially identified. Computer-based signs first achieved the status of a true sign system when they became embedded in organizations and were extensively used for practical daily communication and coordination. Since communication is an important part of organizations, change of communication means change of the organization. On the other hand, once in use, computer-based signs may give rise to new forms of organizations, such as the virtual office. Multiuser systems connected through networks cause new kinds of semiotic systems to emerge, a kind of collective memory in an organization, a flow of information that seems to exist and change independently of the individual user.

The omni-presence of computer based signs.

In the same way as written language, computer-based signs permeate society. Unlike, for example, films and music, which are used only in the cultural activities of society, computer-based signs are used everywhere: at work, in the family, for entertainment, in political discussions. One may even say that they occur in areas that written language has never reached. Computer-based signs enter as active components in most artifacts we use in daily life. We are aware of them in cars, refrigerators, clocks, radios and TV sets; in manufactoring equipment; in ships and airplanes. In all these artifacts, they are used for controlling physical machines and providing an interface for their users. Semiotics is the meta-language that enables the design of highly effective systems of computer-based sign processing.

Nothing is a a sign unless intepreted as a sign

The core of any semiotic theory is the sign. Within the notion of the sign we shall work with, an epistemology is already embedded: we can know only as much as the magnifying glass of the notion of sign we use affords. Setting a sign perspective allows us to distinguish between various epistemological possibilities: knowing

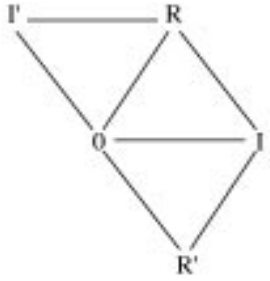
as a process based on language (Saussure's diadic model of a sign); or as a process based on acknowledging structures (the structural approach in its entirety), on logic (Peirce's system), or on the dynamics of cognitive systems. There is no universal key; each context requires a different perspective.

It is obvious that common-sense reasoning embodies language processing. The linguistic sign (as the unity between something signified and a signifier) adequately supports interaction between us and programs based on this type of reasoning. In other situations, such visualization, signs appropriate to the task at hand—i.e. visual representation—will be more adequate.

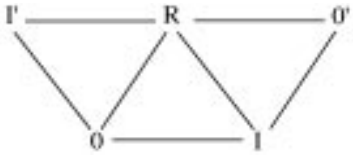
Multimedia, which unite various data types, is, of course, a computational challenge. But it is even more a semiotic experience of a type different from that embodied in the processing of homogeneous single data types. We can use previous semiotic knowledge for word processing and for desktop publishing. In the case of multimedia, we can use semiotic knowledge to structure meaningful interactions. The dynamics of particular sign processes and the dynamics of the composite multimedia hypersign are fundamentally different.

Computation equals sign processes

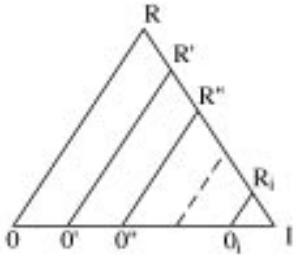
In the examples that make up the core of this book, it will become obvious that the semiotic task starts by defining the nature of the sign we focus our attention on, and thus the nature of the sign processes that need to be defined. In every computation, we do not dealing with isolated signs in a static situation, but with signs in a dynamic context. Indeed, as fundamental as the sign is for any semiotic approach, sign processes are the actual embodiment of the life of such signs (Figure 3).



a.
The object stays the same, the representamen is different, the interpretation process is different (semantic preservation.)



b.
The representamen stays the same (syntax preservation.)



c.
The interpretant stays the same.

Sign processes

Computers, as we know them today, are digital machines. They operate upon representations of data and of instructions making up what are known as programs. Semiotically speaking, they operate in a language of only two signs—0 and 1, or Yes and No—and of operations defined by the Boolean logic embodied in its com-

ponents. The major task we face is to map between the infinite universe of our existence and activity to a world where the only words known are Yes and No, and where the only logic is that embodied in the rules pertaining to these two values. This is what makes the task so incredibly exciting and so formidable.

Moving these zeroes and ones faster and faster and subjecting them to more and more operations embodied in the silicon chip will not make them more meaningful to us. Interactions that map from our intentions and plans to this austere universe are what we are looking for. Computers are omnipresent. Intimidating to those who have not grown up with them, they are nonetheless here to stay. But even those familiar with their presence still have to tame them. Programers are doing it, some very successfully, learning how to think in an intermediate language, also known as a programming language. They are, however, dependent upon the layers between their competence and that of the designers of chips and other components of computers.

In search of coherence

Computer scientists adopted early on the semiotic distinction between syntax (the representamen domain), semantics (the domain of the relation between representamina and objects), and pragmatics. Still, the notion that programmers, or system designers, or programing language authors are involved in semiotics, however unknowingly (like Monsieur Jourdain, in Moliere's comedy, who discovered that he is used prose), begs a short comment. Each particular semiotic contribution is important for the particular activity. But what qualifies the semiotic approach is dedication to the whole image, the coherence of the integrated parts. A good interface will never automatically guarantee the success of a program. A good program with difficult interactions will perform at a percentage of its potential. A coherent integrated semiotic strategy extends to everything that supports and defines the activity. In some ways, such a semiotic strategy is the meta-program that unites program, data flow, I/O performance, connectivity, process and human interface, cultural and social acceptance, learning, and—why not?—satisfaction.

Transparency

Every time we interact with programs, we interact with those who wrote them. The same holds true within the interface of virtual reality, even within the computer games we play. The degree of transparency of any program is indicative of its semiotic adequacy. One of our purposes is to clarify what can be done, and how, in order to achieve a level of transparency that does justice to the program—i.e., uses its possibilities to the maximum—but also to the user—i.e., integrates the user's competence in meaningful interaction.

That much can and should be done in order to achieve some of these goals is relatively accepted inside and outside the computer community. Still, the accent remains on technological progress independent of the preoccupation with coherently integrated systems. And so, a very interesting situation is generated: Instead of improving the means and methods by which we interact with digital machines, we keep reproducing skewed human-machine communication schemes increasing the computational overhead. This is how and why operating systems expand beyond any reasonable limit, and computing resources end up being consumed, not primarily for the function for which systems are built in the first place. In the current war

between operating systems and browser providers, the focus is on Mhz, MB, and Mb.

This book is about how we can usefully/efficiently use computation resources for functions, not for hiding the inappropriateness of our means of interaction with them.

Acceptance

When we began work on this book, each of the authors was involved in practical and theoretical work on the subject. It was encouraging for us to see that the computer science community discovered that semiotics is not some exotic ingredient for their pürogramming cookbooks. The ACM initiatives on computers and semiotics and the Dagstuhl seminar, and the series of colloquia on semiotics and parallel processing added to our sense of being not only useful but also desired. This, of course, does not yet make everything we do automatically stand up to the exigencies of the tasks we are confronted with. If anything, only a tight cooperation among semioticians dedicated to the interaction between humans and computers and computer scientists will eventually bring us closer to achieving our common goals.

It is sad but true that we burn an immense amount of computing cycles only to keep primitive models of human-computer interaction in the zone of acceptability. It is infinitely more saddening that we burn many human cycles of thinking and creativity in interactions so obscure and so inefficient that at times one wonders whether the same task cannot be better performed without machines. Our goal is to provide students dedicated to such problems with a text that will further guide their research. If this book ends up guiding, as well, professionals working in the field, we will be more than satisfied.