Scientific Research Publishing

Journal of Computer and Communications, 2018, 6, 91-101
http://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

A Semiotic Analysis of Programming Languages

Ricardo Maciel Gazoni

TIDD—Technologies of Intelligence and Digital Design, PUC—Pontifical Catholic University, São Paulo, Brazil
Email: gazoni@semiotic.com.br

How to cite this paper: Gazoni, R.M. (2018) A Semiotic Analysis of Programming Languages. *Journal of Computer and Communications*, **6**, 91-101.
https://doi.org/10.4236/jcc.2018.63007

Received: January 26, 2018
Accepted: March 20, 2018
Published: March 23, 2018

Abstract

The paper presents an analysis of computer programming languages based on the semiotics of Charles S. Peirce. The author describes how such languages could evolve in order to achieve some of the expressive characteristics of natural languages. This description evinces otherwise unnoticed differences between natural languages and programming languages, showing that Peircean semiotics is an efficient analysis tool. This suggests further research on the technical features needed to implement an easier way to program computers.

Keywords

Programming Languages, Natural Languages, Semiotics, Charles S. Peirce

1. Introduction

The well-known difficulty to program computers [1] motivated the author to approach the issue from the perspective of semiotics, the science of signs, in order to find new insights. There are several schools and trends of semiotics today, but the present article is based on the work of the American philosopher Charles S. Peirce (1839-1914), whose studies encompass semiotics as a branch of philosophy, more precisely as a sign-based theory of cognition. The idea to apply semiotics to computer science has been long known [2]-[7], but the only work that tries to apply it directly to programming languages is the book of Tanaka-Ishii [8]. There, the author does not fully apply Peircean theory: she instead proposes a new interpretation of semiotic concepts, resulting in a work that departs from a purely Peircean approach and misses its descriptive power for cognitive processes.

Peirce himself frequently used the terms logic and semiotic interchangeably. The publication of his writings began in the first half of the 20th century but is not yet concluded. Its understanding is still in progress, and it is common to see studies of parts of his work—such as the pragmatist philosophy, or even his se-

miotic—that lack a broader perspective. It is neither possible nor necessary to cover it all in this paper. Nevertheless, there are good resources for the study of his work, although part of it is only available in Portuguese. Among these resources are the work of Santaella [9]-[15], Nöth [16] [17] [18] and eventually both [19].

Let us start with the Peircean categories and then give a brief view of the sign and its agency. We will then address the topic of natural languages before we focus on computer languages. Finally we will try to foresee how computer languages may evolve.

2. The Peircean Categories

What the Peircean categories are is better described in Peirce's own words:

> I essay an analysis of what appears in the world. It is not metaphysics that we are dealing with: only logic. Therefore, we do not ask what really is, but only what appears to everyone of us in every minute of our lives. I analyze experience, which is the cognitive resultant of our past lives, and find in it three elements. I call them Categories (CP 2.84, 1902)[1].

Notice that the Peircean categories, at this point of his work, are not metaphysical: they are rather phenomenological. His main concern is not to state what the world is, but what it appears to be. It is essential for the task at hand, which is to delineate semiotic (or logic), the science that aims the understanding of signs and inferences. The categories are three: Firstness, Secondness and Thirdness, terms that are "to be preferred as being entirely new words without any false associations whatever" (CP 4.3, 1898). In a few words,

> The first is that whose being is simply in itself, not referring to anything nor lying behind anything. The second is that which is what it is by force of something to which it is second. The third is that which is what it is owing to things between which it mediates and which it brings into relation to each other (CP 1.356, 1890).

We now detail them further, starting from the easiest to understand.

2.1. Secondness

To describe Secondness, Peirce used words such as "brute force", "binarity" (CP 2.84, 1902), "Obsistence (suggesting obviate, object, obstinate, obstacle, insistence, resistance, etc.)" (CP 2.89, 1902). Secondness describes phenomena based on dyadic relations, where we identify two elements interacting in a necessary way. The easiest way to observe it is to look around: the reader is certainly surrounded by objects that do not change. The material world imposes itself on us, independently of our will, by brute force.

But Secondness is not limited to materiality. Some mental phenomena also

[1]Citations to the Collected Papers [20] are made within parenthesis in the form "CP V.P, Y", where V stands for the volume, P for the paragraph number and Y for the year the text was written.

resist change. I ask the reader to imagine, now, a dragon—or any other nonexistent creature, provided that it has a color. Then, answer yourself the question: which color is the creature just imagined? It does not matter what the answer is but this color will not change. Of course, it would be possible to imagine another creature with other color, or the same dragon with another color. But once imagined, and once the question is answered, nothing will change that dragon's color, even if the memory fails. A case of mental Secondness.

And Peirce goes beyond, classifying deductive inferences as of the nature of Secondness: in such modes of reasoning the conclusions are linked to the premises by necessity. It is not possible to deny the conclusions of a deduction having accepted its premises.

2.2. Thirdness

To see Thirdness in action, it is enough to think about the future. Since it is not possible for a future event to affect us directly—in an action driven by Secondness—we conclude that this action is indirect, an indirect, mediated relation, such as any written word, for instance the word "sea". The marks on the paper, or on the computer's screen, are a manifestation of Secondness. But they are not the most important aspect of the word, because it is not the physical characteristics of this particular set of letters that matters. What matters is that it stands for something that is not there. It creates a relation between three elements—the word, the sea and the reader.

Many phenomena are prominently determined by Thirdness. In Peirce's words,

> Some of the ideas of prominent Thirdness which, owing to their great importance in philosophy and in science, require attentive study are generality, infinity, continuity, diffusion, growth, and intelligence (CP 1.340, 1895).

Also the concept of sign, the subject matter of semiotics:

> The easiest of those which are of philosophical interest is the idea of a sign, or representation. A sign stands for something to the idea which it produces, or modifies (CP 1.339, undated).

Notice also that laws are of the nature of Thirdness, albeit their application is of the nature of Secondness.

2.3. Firstness

Any manifestation of Firstness relates only to itself. Therefore it cannot be perceived directly. However, it is possible to infer some of its characteristics. In Peirce's own words:

> The idea of the absolutely first must be entirely separated from all conception of or reference to anything else; for what involves a second is itself a second to that second. The first must therefore be present and immediate,

> so as not to be second to a representation. It must be fresh and new, for if old it is second to its former state. It must be initiative, original, spontaneous, and free; otherwise it is second to a determining cause. It is also something vivid and conscious; so only it avoids being the object of some sensation. It precedes all synthesis and all differentiation; it has no unity and no parts. It cannot be articulately thought: assert it, and it has already lost its characteristic innocence; for assertion always implies a denial of something else. Stop to think of it, and it has flown! What the world was to Adam on the day he opened his eyes to it, before he had drawn any distinctions, or had become conscious of his own existence—that is first, present, immediate, fresh, new, initiative, original, spontaneous, free, vivid, conscious, and evanescent. Only, remember that every description of it must be false to it (CP 1.357, 1887-1888).

Peirce does not banish the idea of "the first from the second; on the contrary, the second is precisely that which cannot be without the first" (CP 1.358, 1887-1888). Secondness cannot be without Firstness, and Thirdness depends on Secondness to become embodied.

3. Language and Signs

We tend to think natural languages as cultural constructions. Indeed they are, but to say so is not the same as saying that words are created on purpose. In fact, every attempt in history to create an artificial language has failed. In a Peircean framework we can try to find a possible explanation based on a somewhat intriguing affirmation: signs—and words are signs—are determined by their objects (CP 4.531, 1905). Surprisingly, deeper analysis confirms this counterintuitive idea. We begin with the concept of sign.

3.1. Sign

The Peircean definition of sign evolved in time, and several definitions can be found along his writings. Some of them are more complete and complex, some were simplified for didactic reasons. One of these follows:

> A Sign, or Representamen, is a First which stands in such a genuine triadic relation to a Second, called its Object, as to be capable of determining a Third, called its Interpretant, to assume the same triadic relation to its Object in which it stands itself to the same Object. The triadic relation is genuine, that is its three members are bound together by it in a way that does not consist in any complexus of dyadic relations. That is the reason the Interpretant, or Third, cannot stand in a mere dyadic relation to the Object, but must stand in such a relation to it as the Representamen itself does (CP 2.274, 1903).

In this definition, Peirce equates sign and representamen. In other definitions, the sign is the relation between representamen, object and interpretant. What is

important is that a sign has three constituents, the representamen, its object, and an interpretant. The three are related, not two by two, but all three in one and the same relation. Some characteristics of each component are:

1) The representamen is the first, *i.e.*, it has to be present for the action of the sign. There is no rule to define what ought to be the representamen of a sign. Anything can be a representamen, but this does not mean that everything is a representamen. Each word of the present article is a representamen, but also the entire article is another representamen, as well as any section of it.

2) The interpretant is the effect of the sign; the process that creates the interpretant is called semiosis. In the same way as for representamens, there are no rules to define what may be an interpretant. Anything can be an interpretant, provided that it is the effect of a sign. In particular, an interpretant can be another sign that has an interpretant of its own, that may also be another sign, and so on, in a possibly infinite chain of signs.

3) Anything can also be the object of a sign since the object is whatever the sign refers to.

It seems that the nature of the components is not the most important in a sign since almost any class of things can play a role in a sign. What must be taken into account is how the three components relate, how the object determines the sign. Let us start with the use of a sign, taking the use of an existing word as example (a word commonly is a representamen). Suppose someone gives you a call and asks, "Are you seated?" The answer, of course, depends on your position. And if you intend to tell the truth, your answer will depend whether you are seated or not. If the answer is "yes", the word was determined by the fact that you are seated. Hence, the fact determined the word—the object determined the sign.

But then comes the case of the creation of new words. How is the sign—the word—determined by the object in this case? The analysis is also simple. In a nutshell, the word exists because there is something that has to be expressed by it. It is the need to designate whatever is relevant that motivates the creation of words. It is hardly possible to invent a word to represent an irrelevant thing. Although the form of the exact sound of the word is arbitrary, it is also irrelevant to its meaning. The word "tree" in English was created due the same object that led to the creation of the word "árvore" in Portuguese. They sound very differently, but this difference is irrelevant: their objects are the same. The differences in interpretants of both words that may occur in speakers of the two languages are due to what Peirce called the "collateral experience" that each interpreter may have on the subjects (CP 8.179, 1901), and interpretants differ not only between speakers of different languages but also between cultures. Hence, when creating a word, only its form—the representamen—is the creator's choice, and may be arbitrary, but the sign itself is determined by its object.

3.2. Natural Languages

It is clear by now that natural languages are not arbitrarily created from nothing.

They intend to reflect objects that are relevant to the community of speakers of the language, including cultural and social traditions to say the least. The life of a sign in this context is amazingly interesting. Signs tend to increase and gain information as time passes [18]. For instance, the word "electricity" in the eighteenth century could mean a sort of parlor game: people amused each other by rubbing a piece of glass with silk cloth and using it to "magically" move small objects. Today, the word designates much more than this. But the word did not change, neither did the underlying phenomenon.

3.3. Vagueness

This allows us to point an important feature found in natural language: its intrinsic vagueness [19]. Whatever "electricity" may mean, its precise definition is not necessary for people to use the word. Hence, it is possible to increase the amount of information a sign gives about the world. It begins with a vague designation of some relevant phenomenon in vague words, and this vague representation allows us to talk about whatever is not yet very well defined. The not very well defined word enables us to communicate about not very well defined objects which in turn results in the increase of its information about the objects represented by the words. The word "intelligence" is one of these rather vaguely defined words that is currently subject of scientific study. We can conclude that it appears to exist a "weak" artificial intelligence and a "strong" one, only because we were allowed to talk about artificial intelligence by using an only vaguely defined term, "intelligence".

It is possible to identify two main forms of vagueness in natural language. Brifly, we have "Thirdness-based" vagueness, such as in the phrase "all men are mortal". What is vague here is who exactly is subject to death. The answer: whichever one wants. Not a specific man, nevertheless one is able to choose a case for further reasoning. "Firstness-based" vagueness happens in phrases like: "an important man will die next year". Here, we cannot choose any specific man in order to draw further conclusions. Another case of "Firstness-based" vagueness is that behind the name of certain properties. When we say "I am wearing a yellow t-shirt" it is not precisely defined what we mean by "yellow". There is no "Secondness-based" vagueness.

4. Computers and Programming Languages

Computers are electronic devices which are determined by the laws of mechanical causality. We can say that their functioning is "Secondness-based". So are computable functions. They are functions that can be performed by Turing machines, a mathematical construction for which the intuitive concept of computer is appropriate—the word "computer" formerly designated people who made calculations. Since for Peirce all deductive inference is of the nature of Secondness, as stated in Section 2.1, it is clear that we are in a domain where semiosis has mainly a character of Secondness.

Interestingly enough, although we are in such domain, we know from Turing's article [21] that the outcome of computable functions cannot be predicted by a computable function, which makes them unpredictable—not in the sense that random phenomena, such as radioactive decay, are unpredictable, but in the sense that it is impossible to define precisely a procedure that predicts their outcome. This raises the interesting question whether purely mathematical constructs bear some sort of vagueness. The question has to remain open for now.

4.1. Programming languages

Programming languages are ordinarily presented as mechanisms that simplify the use of computers' central processing unit (CPU). These electronic devices behave according to electrical charges that can be understood as bits. These can be seen as an incomprehensible list of numbers that are, at the bottom line, the instructions that the CPU is forced to obey. Programming languages exist in order to make these lists of numbers easier to create and modify. Programmers use programming languages in order to write simple instructions in English-like words that will then be translated into a bunch of numbers somewhere in the memory of the computer. The most distant from a technical specification and more user-friendly the language, the higher is its level. There are many styles of high-level languages, each aiming at a different kind of use: gaming, business, statistics, etc.

However, two moments of programming languages have to be distinguished. They are first invented, when their tokens (the special, or reserved, words of the programming language) are defined and implemented. At a second moment, these languages are used by programmers to write programs. The programmers also create new words, hereafter called ordinary program words, to designate variables, objects, classes, functions etc. When a program is running, it implements, in a computer, the computable functions represented in the program written using the program language.

Programming languages bear some resemblance with natural languages in the sense that the creation and use of the words—either tokens or ordinary words—are defined by relevant objects outside the language. This happens at the two moments cited above with different scopes.

4.2. Creating a Programming Language

When, at a first moment, creating the language, the creator is concerned with the use programmers may be interested in. The creator then defines the reserved words that will be translated into CPU instructions. The final product of this work is a program that either compiles or interprets programs written in the language created. Here it is important to say that even if the future use of the new language may be vague, the CPU routines that it implements are not. Every programming language is constructed in a way that does not allow vague words, either special or ordinary ones. Anything written in a programming language

necessarily translates into CPU instructions that will ultimately stimulate electronic circuits in a well-defined way. This whole process is "Secondness-based".

It is not unusual that the creation of the language takes into account that the programmers will use some sort of software to aid the programming process, such as an integrated development environment (IDE). Such tools ease the construction of programs not only by pointing wrong constructions but also by suggesting corrections and the use of certain words, either special or ordinary ones, according to the programming context. The use of IDEs allows that the programmer may vaguely know what he or she intends to do, and the software will help to define the exact program. But the final text of the program does not contain vague statements, even when the programmer cannot state precisely what each word in the program mean.

4.3. Writing a Program

Then, in a second step of creating a program with a programming language, programmers are concerned with the purpose of the program and its desired features. Now they create new ordinary program words to represent structures, data and actions that will be ultimately translated into CPU instructions by the compiler or the interpreter of the language. As stated above, the final product does not contain any sort of vague expressions, even when the programmer is unaware of it. It is now easy to establish a comparison between natural and programming languages.

5. A Natural Programming Language

The most important semiotic difference between programming languages and natural languages lies in the fact that the former does not bear any vagueness. For this reason, unlike the latter, programming languages cannot be used to express whatever one wants. They express only what can be further translated into CPU instructions and this excludes all vague statements. Particularly, the text of programs may not completely express the features they implement, and this is the reason why program documentation—written in natural language—is so important.

Natural languages, on the other hand, allow not only vague statements but also precise ones. So, they can express anything, including what a program does. It is the role of program specification. And although some vaguely expressed features that are desired in programs can indeed be further detailed and become very precise, it does not always happen. Some features may exist that cannot be precisely described by their own vague nature.

This raises the question of how a programming language that allows vagueness could be? How would it be to program a computer using such language? We are now able to suggest some answers.

The current process of computer programming is a command-obey schema. It happens due to the "Secondness-based" nature of the task. Everything has to be

precisely defined in advance, and this is the greatest difficulty. But if we were able to use vague terms in a computer, then this interaction schema would change. It makes no sense to order a vague action to a computer and expect a consistent answer.

A suggestion that allows dealing with vague orders may be: first, try to gather further information about the statement. And when it is not possible, try to guess what to do, and act accordingly, but try to do things as harmless as possible. If the one who gave the order is present, she can always amend the course of action. Roughly speaking, any schema to deal with vagueness resembles more a kind of dialog instead of a command-obey schema. This means that to write a program using a programming language that allows vagueness is somewhat equivalent to talking to the computer.

An example of such dialog would be something like: "Computer, can you generate a report like the one I am showing with the data I am furnishing you?"—notice, there is no clear instruction here, only the format of a desired outcome to be generated with a certain input. A possible answer: "Well, I don't know how to generate this report, but I do know how to generate this or that one, which in my opinion resembles the one you are asking for." No action was taken; only some kind of analysis and choice, a way to gather further information. A reply would be: "Ok, take this second report, sort it by city and show the totals by state, let's see what comes from it." The overall idea is clear, although the exact phrases used in this example may not be realistic.

Although it seems that some sort of artificial intelligence is required to achieve the dialog shown above, it may not be exactly the case. Proper intelligent action can be identified whenever one tries to better define some vague statement. To do so requires observation, analysis, and inference proper. In other words, intelligent action is needed for signs to evolve, and the work of Peirce can be used to detail this process in a very consistent way. Nevertheless, intelligent action does not seem necessary to deal with vagueness in those cases where inference is not needed. As can be intuitively seen in the dialog above, in order to deal with vagueness it should suffice to avoid disaster and try to gather more information about vaguely defined terms. This is much more than current programs are able to do; however, stated this way, it does not seem an impossible task.

6. Conclusions

The analysis based on Peircean philosophy shows that current programming languages present intrinsic limitations due to their semiotic nature. Surpassing these limits implies an important change in the mode of interaction with computers. It is of course necessary further analysis in order to understand the techniques needed for this specific task. Nevertheless, the work of Peirce seems to be useful on a number of applications within computer science [22], including program specification [23].

It remains an open question whether programming languages could deal with

vagueness in the same way as natural languages do, and how to accomplish it. In spite of that, semiotic analysis seems to be an underestimated tool since it helps us to state good questions and to answer some of them.

Acknowledgements

Sincere thanks to Professor Winfried Nöth for the insightful comments that gave this work its current form. I also thank an anonymous commenter for suggestions incorporated in this final version.

References

[1] Standish Group (2013) CHAOS Manifesto 2013. https://www.scribd.com/document/198550543/Chaos-Manifesto-2013

[2] Zemanek, H. (1966) Semiotics and Programming Languages. *Communications of the ACM*, **9**, 139-143. https://doi.org/10.1145/365230.365249

[3] Andersen, P.B. (1992) Computer Semiotics. *Scandinavian Journal of Information systems*, **4**, 3-30.

[4] Andersen, P.B. (1995) The Force Dynamics of Interactive Systems: Toward a Computer Semiotics. *Semiotica*, **103**, 5-45.

[5] Liu, K. (2000) Semiotics in Information Systems Engineering. Cambridge University Press, Cambridge. https://doi.org/10.1017/CBO9780511543364

[6] de Souza, C.S. (2005) The Semiotic Engineering of Human-Computer Interaction. The MIT Press, Cambridge.

[7] de Souza, C.S. and Leitão, C.F. (2009) Semiotic Engineering Methods for Scientific Research in HCI. *Synthesis Lectures on Human-Centered Informatics*, **2**, 1-122. https://doi.org/10.2200/S00173ED1V01Y200901HCI002

[8] Tanaka-Ishii, K. (2010) Semiotics of Programming. Cambridge University Press, Cambridge.

[9] Santaella, L. (2012) Percepção: fenomenologia, ecologia, semiótica. Cengage Learning, São Paulo.

[10] Santaella, L. (1998) A percepção: uma teoria semiótica. Experimento, São Paulo.

[11] Santaella, L. (1994) Estética de Platão a Peirce. Experimento, São Paulo.

[12] Santaella, L. (2000) A teoria geral dos signos. Pioneira, São Paulo.

[13] Santaella, L. (2002) Semiótica aplicada. Thomson, São Paulo.

[14] Santaella, L. (2004) O método anticartesiano de C. S. Peirce. Editora UNESP, São Paulo. https://doi.org/10.7476/9788539303236

[15] Santaella, L. (2005) O amplo conceito peirciano da mente: sua relevância para a biologia, inteligência artificial e cognição. In: Ferreira, A., Gonzalez, M.E.Q. and Coelho, J.G., Eds., *Encontro com as ciências cognitivas*, Cultura Acadêmica Editora, São Paulo, 167-180.

[16] Nöth, W. (2007) Máquinas Semióticas. In: Queiroz, J., Loula, A. and Gudwin, R., Eds., *Computação*, *Cognição*, *Semiose*, EDUFBA, Salvador, 159-183.

[17] Nöth, W. (2012) Charles S. Peirce's Theory of Information: A Theory of the Growth of Symbols and of Knowledge. *Cybernetics & Human Knowing*, **19**, 99-123.

[18] Nöth, W. (2014) The Growth of Signs. *Sign Systems Studies*, **42**, 172-192. https://doi.org/10.12697/SSS.2014.42.2-3.02

[19] Nöth, W. and Santaella, L. (2011) Meanings and the Vagueness of Their Embodiements. In: Thellefsen, T., Sørensen, B. and Cobley, P., Eds., *From First to Third via Cybersemiotics—A Festschrift Honoring Professor Søren Brier on the Occasion of his 60th Birthday*, SL Forlagene, Copenhagen, 247-282.

[20] Peirce, C.S. (1931-1958) The Collected Papers of Charles Sanders Peirce. Harvard University Press, Cambridge, Vol. 1-8.

[21] Turing, A.M. (1936) On Computable Numbers, with an Application to the Entscheidungs Problem. *Proceedings of the London Mathematical Society*, **2**, 230-265.

[22] Gazoni, R.M. (2015) Peircean Semiotics: Perspectives in Computer Science. *Proceedings of the* 2015 *International Conference on Foundations of Computer Science*, Las Vegas, 27-30 July 2015, 79-85.

[23] Gazoni, R.M. (2016) Creative Thinking in Artificial Intelligence: A Peircean Account. *Proceedings of the* 2016 *International Conference on Computational Science and Computational Intelligence*, Las Vegas, 15-17 December 2016, 537-540.