A Computational Semiotics Approach for Soft Computing

Ricardo Gudwin and Fernando Gomide

DCA-FEEC-UNICAMP

Caixa Postal 6101

13.083-970 – Campinas, SP - Brasil

e-mails: gudwin@dca.fee.unicamp.br, gomide@dca.fee.unicamp.br

ABSTRACT

The aim of this paper is to introduce a unified framework for soft computing based on the computational semiotics approach. Derived from semiotics, a classic discipline in human sciences, computational semiotics provides a mathematical foundation for the concept of knowledge and for knowledge processing that is suitable for intelligent systems design and implementation. As a result, a new perspective to study and to develop intelligent systems emerges. The theory introduces the mathematical definition of objects, object systems and object networks. They are used to model the different types of knowledge used in building intelligent systems. Particularly in this paper, we focus on the use of computational semiotics framework to construct fuzzy systems, neural networks and evolutive systems. The paper also addresses soft computing system in an unified manner.

1. INTRODUCTION

Computational intelligence [9], and soft computing [8], are two terms coined to specify the field within the study of intelligent systems which try to model the intelligent behavior using some ideas from biology and considering the definition and use of uncertainty in computing. The three basic methodologies within soft computing are fuzzy systems, neural networks and evolutive systems. Initially, the methodologies were considered as isolated fields. Progressively, it started to appear hybrid models: neuro-fuzzy, neuro-genetic, fuzzy-genetic, etc, that revealed the complementarity of these concepts and the possibility to put them all in a common paradigm However, hybrid systems still lack an unified framework to characterize its fuzzy, neural or evolutive characteristics in the same formalism. A neuro-fuzzy system is either a fuzzy system with some neural enhancements or a neural network with some fuzzy characteristics. Similarly, a fuzzy genetic system is either a fuzzy system with some genetic components or a genetic algorithm with fuzzy capabilities. The same characteristics can be perceived in combinations of fuzzy-neural-evolutive systems found in the literature.

Semiotics, a traditional discipline in human sciences, was introduced in the nineteen century by Charles Peirce [7]. Essentially, semiotics studies the basic aspects of cognition and communication. Cognition means to deal with and to comprehend phenomena that occur in an environment. Communication means how a comprehended phenomena can be transmitted between intelligent beings. The basic unit of semiotics is called **sign**, defined as anything that in a certain way or aspect, represents something to someone [7]. Semiotics considers how signs are created, how they represent the different aspects of a phenomenon, and how signs can be used to store and to transmit information.

The computational semiotics approach [2,3,4,1] is the computational view of semiotics for modeling, development and implementation of intelligent systems. Computational semiotics is build upon a mathematical description of concepts from classic semiotics. Its formal contents can be regarded as a contribution towards the development of semiotic specific mathematical tools. Thus, it is in the very realm of the formal foundations of intelligent systems.

In this paper, we focus on the use of the tools from computational semiotics as a general framework for soft computing. We show how each methodological cornerstone of soft computing can be instanciated in the computational semiotics formalism, and how hybrid systems can be constructed as well. In this sense, the computational semiotics tools are not envisioned as a new enhancement in any particular area of soft computing, but as a meta-theoretical tool able to formalize all the different behaviors captured by each of the formalisms within soft computing.

2. A MATHEMATICAL THEORY OF OBJECTS

This section introduces some background concepts and definitions used by computational semiotics. The focus here is on the main issues and definitions only. For a more in depth coverage the reader is referred to [1] or [2,3].

2.1 Variable

Let T be a countable set (a "time" set) with a generic element t, and $X \subseteq U$. A variable x of type X is a function $x : T \to X$. Note that a function is also a relation and hence it can be expressed by a set. Thus, $x \subset T \times X$.

2.2 Class

A class C is a set whose elements c_i are tuples of the type:

$(v_1, v_2, \ldots, v_n, f_1, f_2, \ldots, f_m)$, $n \geq 0$, $m \geq 0$

where $v_i \in V_i$, and f_j are functions

$$f_j : \bigtimes_{p \in P_j} V_p \to \bigtimes_{q \in Q_j} V_q .$$

Here $\bigtimes$ means the cartesian product, $P_j \subseteq \{1, \ldots, n\}$ and $Q_j \subseteq \{1, \ldots, n\}$.

2.3 Object

Let C be an non-empty class and c be a variable of type C. Thus c is an object of class C.

Observe that, by definition, an object is a multi-temporal entity, i.e., the definition of an object comprises all its time history. A more intuitive definition concerns an **instance** of an object, viewed as the value of an object at a particular instant of time, i.e., a tuple $(v_1, v_2, \ldots, v_n, f_1, f_2, \ldots, f_m)$.

2.4 Object System

A set of objects c_i is an object system if the c_i's are related to each other in the sense that each instance of such objects, at a given instant, is a function of the instances of all objects at the previous time instant:

$$c_k(t+1) = f(c_1(t), \ldots, c_n(t)).$$

This is only a concise definition of an object system. The complete definition is far more involved. The reader is referred to [1] or [2,3] for details.

2.5 Object Network

An object system is a generic approach for modeling elementary knowledge types. An object network is a special type of object system in which additional restrictions concerning interactions are included. To distinguish object network and object system let us assume places and arcs whose roles are similar to those in Petri nets. Objects in places can only interact with objects in places connected through arcs. Thus, at each instant, the objects defined should be at one place. For each place there is a set of places connected with through input arcs. These places are called the input gates of the place. Analogously, each place has a set of places connected with it by means of output arcs, called output gates. Observe that objects can be of two types: passive and active. Passive objects do not have functions in its tuples and are only used to store information. Active objects do have functions in its tuples, and perform the task of transitions in the object network. Each place can only have objects of the same class. In this sense, we can say that there are passive and active places, if the objects that can be put in a place are passive or active, respectively.

Object networks can be put in a graphical form, in wich places are represented by circles and arcs by lines. Passive places are indicated by circles. Active places are indicated by double circles, and instances of objects by black tokens, as in figure 1.

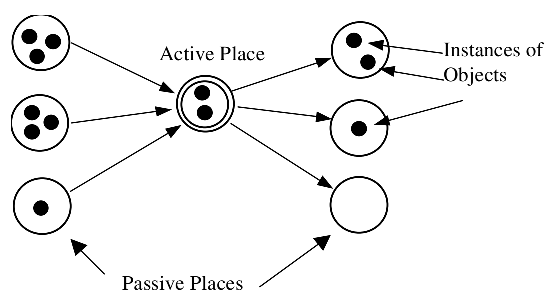

Figure 1 : Example of an Object Network

Observe that differently than in Petri nets [5,6], the tokens are instances of objects that have individuality, i.e., they are not a marking on the place, but are related to objects, with attributes and eventually transforming functions. Again, active objects, that perform the role of transitions, are also mobile and changeable. This gives to an object net great power of representation, allowing the modelling of systems that are not suitable to be modelled by Petri nets, e.g. adaptive systems.

The basic behavior in an object network is the triggering of active objects. Triggering an active object corresponds to the generation of new instances of objects in places directly connected to the place where the active object is through output arcs. To be triggered, an object must first have an enabling scope, that is, a set of object instances put in the input gates, enabling one of the object functions. To select an enabling scope, there is a selection function that select, from the object instances available, those that are to be used for triggering. After triggering, object instances may be put in one ore more output gates of the place where the active object is. This is also determined by the selection function. The object instances used as an enabling scope may (or not) be destroyed for the next time instant.

The complete set of definitions formalizing an object network is omitted here due to space constraints. These definitions can be found in [1] or [2,3] and include the temporal restriction for objects, set variable, generic objects, fuzzy objects, meta-objects, instances of meta-objects, occurrences of meta-objects in objects, generic objects and fuzzy objects, generic meta-objects, occurrences of generic meta-objects in objects, generic objects and fuzzy objects, fuzzy meta-objects, occurrences of fuzzy meta-objects in objects, generic objects and fuzzy objects. All those definitions are used to model the different types of knowledge from the viewpoint of computational semiotics theory.

3. THE FUNDAMENTAL TRANSFORMATIONS

In computational semiotics theory, the transformations that knowledge can suffer are represented by a special kind of knowledge, called argumentative knowledge or, more simply, "arguments". Three basic arguments are indentified: deductive, inductive and abductive arguments. These arguments are modelled by active objects, and the main features characterizing them are their selection and transformation functions, whose purpose are to select the instances of objects that are going to fed the triggering (plus those that are going to be out and the transformation function to trigger), and the transformation function itself, that defines how the input information is used to generate the output, respectivelly.

3.1 Deductive Argument

The deductive argument uses a transformation function which basically collects knowledge from two inputs. The first one constitutes a kind of generic knowledge concerning the knowledge bases of the system. The second input is a specific knowledge to be used as a "key" for the knowledge base. The general scheme for a deductive argument is as in figure 2. In this figure, we find two input places, one of them holding the knowledge base and the other one the specific input that is triggering the deductive argument. One place holds the deductive argument (the active object) and the other one is the output place, where the deducted knowledge is to be put after triggering of the argument.

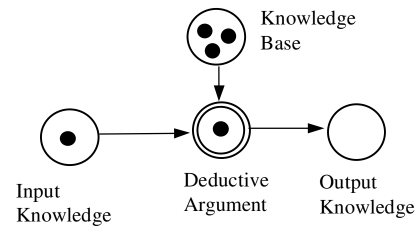

Figure 2 – Deductive Argument

The main issues of the deductive argument, as an active object, are: to select, in the knowledge base, the proper pieces of knowledge and to use them with the input knowledge, activating one of its internal functions (the one more adequate for the type of knowledge involved) to generate the output knowledge

3.2 Inductive Argument

The inductive argument is used to generate a new piece of knowledge that can be compared, in terms of a measure, to other knowledge already in the system. The basic configuration of an inductive argument is depicted in figure 3.

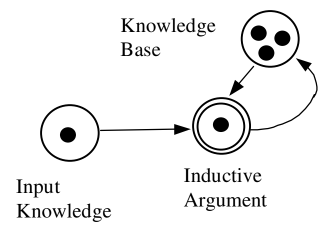

Figure 3 – Inductive Argument

The general use of inductive arguments is to modify a piece of knowledge previously stored in a knowledge base, according to an input. In this sense, the modified knowledge is at the same time input and output of the inductive argument. But it can also be found in a scheme similar to the one in figure 2, changing the deductive argument by an inductive argument. The difference between them is that the knowledge generated by a deductive argument is already contained (in other format) in the knowledge base. In this sense, it only "extracts" the knowledge from the knowledge base. The inductive argument generates a new piece of knowledge, that is not in the original knowledge base. The unique relationship to the knowledge in knowledge base is that it is possible to “measure” its distance to other pieces of knowledge in knowledge base, and this distance is always greater than zero.

3.3 Abductive Argument

The abductive argument is used to validate a piece (or pieces) of knowledge previously generated by a knowledge generator (e.g. an inductive argument or some sort of random process). In this case, the argument does not generate a new piece of knowledge, but chooses, from a set of different candidates, the piece of knowledge that is more plausible (given a plausibility measure). This scheme is shown in figure 4.

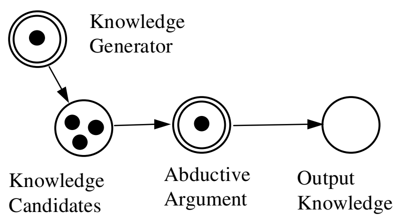

Figure 4 – Abudctive Argument

In terms of a mathematical object, what the abductive argument do is to use its selection function to choose an object among the objects in the input place, and to move it to the output place. In this sense, it does not use its internal function to change knowledge. This will be the case only if it is necessary to change the knowledge format.

3.4 Other Arguments

Other types of arguments can be found, considering combinations of the three basic types, or incorporating knowledge inside the structure of the own argument. Those arguments are not discussed here, but they do exist.

4. IMPLEMENTING SOFT COMPUTING THROUGH OBJECT NETS

In this session, we provide examples to show how we can implement soft computing techniques with the computational semiotics tools.

4.1 Fuzzy Systems

A general fuzzy production system is shown in figure 5.

The main argument in a fuzzy production system is the deductive argument in P6, called the inference engine. The other arguments in P1, P3, P8 and P10 are also deductive arguments, but their only task is to convert knowledge into suitable format.

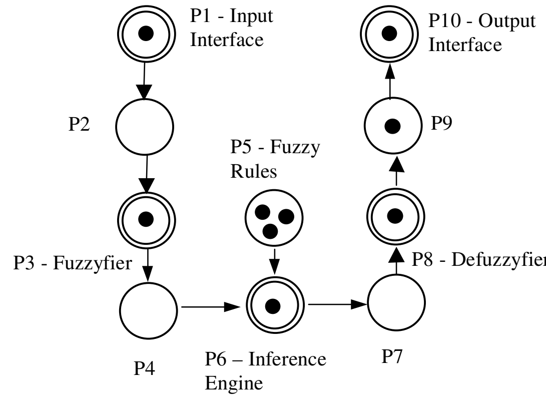

Figure 5 – Fuzzy System

4.2 Neural Networks

Figure 6 shows the representation of a self-organizing neural network by an object network. Note that this neural network has two main arguments. The deductive argument works during the feedforward phase, when generating an output from an input. The learning function is an inductive argument that performs the self-organization of the neural network. It acts considering the input, and modifying the neural structure (i.e., the weights and offsets of the neural net), performing its learning phase.

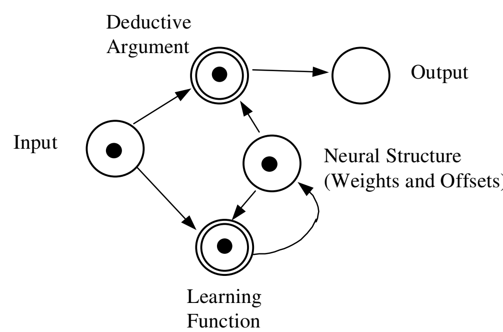

Figure 6 – Self Organizing Neural Network

Other types of neural networks (like supervised neural networks) would have a similar representation, including in this case a place for the desired output feding the learning function argument.

4.3 Evolutionary Systems

An example of an evolutionary system is given in figure 7. In this example, there is an original population, used as input for 4 inductive arguments (performing crossover, mutation, inductive mutation, etc), generating new sub-populations. Those 4 new sub-populations are then used, in conjunction with the original population, to fed an abudctive argument that will choose the new population (and destroy the old one). The best individue of this new population is then extracted by an abductive argument to generate a solution, and the new population is redirected again to the beginning through a feedback deductive argument, that simply moves the new population from the new population place to the original population place.

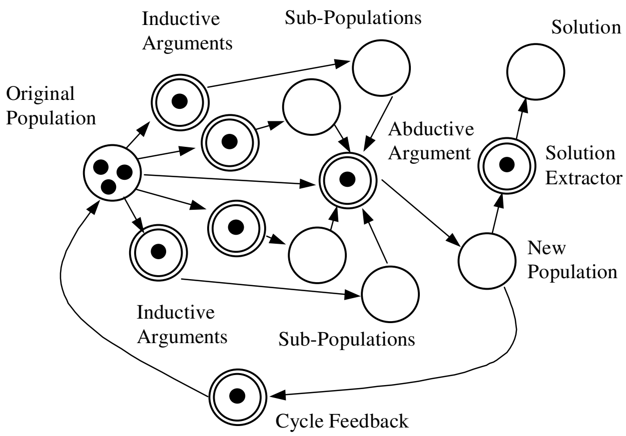

Figura 7 – Evolutionary System

4.4 Hybrid Systems

The soft computing paradigm can be enlarged by using not only its three main constituents (i.e., fuzzy systems, neural networks and evolutive systems), but with the computational semiotics approach, hybrid systems can be constructed as well from the basic tools provided by the basic types of arguments, associated with the three main operations with pieces of knowledge: knowledge extraction, knowledge generation and knowledge selection.

5. CONCLUSIONS

In this paper, we discussed on the use of the computational semiotics tools aiming at an unified representation for soft computing. For this purpose, we introduced the objects networks, a kind of meta-theoretical framework for building intelligent systems. We showed the basic blocks in building knowledge processing systems, discriminated by the deductive, inductive and abductive arguments. These arguments correspond to the functions of knowledge extraction, knowledge generation and knowledge selection respectively. After, we derived examples, using the objects nets, for a fuzzy system, a neural network and an evolutive system. It is worth to mention that despite being used in an informal way through this paper, the object network has a well posed formal definition. To the reader interested in the formal definitions, we suggest the references [1] and [2,3], where a deep coverage can be found.

The main issues addressed by computational semiotics are the following. First, by using an unique formal model, we can specify a whole range of applications within the soft computing framework. This formal model has the advantage of being easily converted into a computational algorithm, leading to a general framework for the design, analysis and implementation of intelligent systems. Second, the formal model is general enough to acomodate specific details of each technique, allowing the use of the most recent developments in each particular area of soft computing. And third, the use of semiotic concepts in the generation of the model can lead to hybrid systems that may contribute for overpass the current developments in soft computing, as they are not tied to any particular model. In this sense, new technologies are suitable to emerge, if we do not define a priori which type of computation is to be adopted (if fuzzy, neural or evolutive), but focus on the knowledge processing which is kept inside them. The development of such hybrid systems, however, is still a matter for future research.

6. BIBLIOGRAPHY

[1] Gudwin, R.R.- *Contribuições ao Estudo Matemático de Sistemas Inteligentes* – Phd Thesis - DCA-FEEC-UNICAMP, Maio 1996 (in portuguese)

[2] Gudwin, R.R.; Gomide, F.A.C. - "Computational Semiotics : An Approach for the Study of Intelligent Systems - Part I : Foundations" – Technical Report RT-DCA 09 – DCA-FEEC-UNICAMP, 1997.

[3] Gudwin, R.R.; Gomide, F.A.C. - "Computational Semiotics : An Approach for the Study of Intelligent Systems - Part II : Theory and Application" – Technical Report RT-DCA 09 – DCA-FEEC-UNICAMP, 1997.

[4] Gudwin, R.R.; Gomide, F.A.C. – "An Approach to Computational Semiotics" – Proceedings of the ISAS'97 – Inteligent Systems and Semiotics : A Learning Perspective – International Conference – Gaithersburg, MD, USA - 22-25 September, 1997.

[5] Jensen, K. - "Coloured Petri Nets : A High Level Language for System Design and Analysis" - *Lecture Notes in Computer Science 483 - Advances in Petri Nets*, pp. 342-416, 1990.

[6] Murata, T. - "Petri Nets : Properties, Analysis and Applications" - *Proceedings of the IEEE*, vol. 77, n. 4, April 1989.

[7] Peirce C.S. - *Collected Papers of Charles Sanders Peirce* - vol I - Principles of Philosophy; vol II - Elements of Logic; vol III - Exact Logic; vol IV - The Simplest Mathematics; vol V - Pragmatism and Pragmaticism; vol. VI - Scientific Metaphysics - edited by Charles Hartshorne and Paul Weiss - Belknap Press of Harvard University Press - Cambridge, Massachussets, 2nd printing, 1960.

[8] Zadeh L., – "Soft Computing and Fuzzy Logic", *IEEE Software*, vol 11, n. 6, pp. 48-56, 1994.

[9] Zurada, J.; Marks II, R.J.; Robinson, C.J. - *Computational Intelligence - Imitating Life* - IEEE Press, USA, 1994.