Fielded Object Networks as a Framework for Computer Intelligence

Rodrigo Gonçalves Fernando Gomide Ricardo Gudwin
Rodrigo@dca.fee.unicamp.br gomide@dca.fee.unicamp.br gudwin@dca.fee.unicamp.br
Department of Computer Engineering and Industrial Automation - DCA - Faculty of Electrical and Computer Engineering - FEEC - State University of Campinas - UNICAMP – Brazil

ABSTRACT

Recently, semiotics has started being the focus of attention of AI researchers due to its interesting capabilities in symbolic processing and knowledge representation. In this paper, we propose the Fielded Object Network (FON), a framework aimed at integrating many scientific fields related to artificial intelligence, e.g. artificial life and distributed artificial intelligence (DAI), in order to get a knowledge representation tool capable of performing semiotic processing. Following this trend, this work presents the basis of FON and shows how it can be used to a hierarchical knowledge processing in intelligent systems. To show that, we implement a Generalized Subsistence Machine (GSM) proposed by Meystel [6] using a FON approach. We also provide simulation results for an AGV application built under the proposed framework.

KEYWORDS: *artificial intelligence, semiotics, artificial life, object-oriented parallel processing*.

1. INTRODUCTION

Recently, the connections between semiotics and intelligent systems has being explored by many researchers in the field of artificial intelligence and intelligent systems, by means of different approaches involving semiotic concepts under the scope of intelligent systems. One of these approaches is the one given by Gudwin [3][4], which introduces a mathematical formalization for the concept of object and proposes how it can be used for knowledge representation and processing based on semiotics concepts. To achieve this objective, a knowledge taxonomy inspired on the work of Peirce [7] was proposed. In this work formal objects are used as both a knowledge representation and processing tool (e.g. how knowledge about the real world is modeled and also how reasoning methods, for example, induction, can be represented using objects). There, the main computational tool is the Object Network (ON) which was successfully tested in an AGV (Automated Guided Vehicle) application. Nevertheless, a lot of things, e.g. the development for formal analysis tools for ON, remain undone.

Based on the ON approach we developed the *Fielded Object Network* (*FON*), an improved representation to handle the same concepts and aiming at the integration of others scientific fields related to artificial intelligence, e.g. artificial life and distributed artificial intelligence (DAI), into an unique framework to develop intelligent systems. This approach follows the current trend of a convergence of symbolic, neuronal and evolutionary approaches in the modern AI [10]. In this paper, we present the main concepts involving the *Fielded Object Network* showing also its reliability in the DAI scope.

In [2] an autonomous intelligent systems engineering methodology is proposed as a complement to the work described in the current paper and some of its aspects are discussed there. This methodology is a work in progress and we are currently evaluating the use of both ON and FON as tools in its development and construction phase.

2. FIELDED OBJECT NETWORK

The main concept of *Object Networks* and *Fielded Object Networks* relies in the *object* definition. An object represents a logical or physical entity in a certain level of abstraction. The *Fielded Object Network* shares the same theoretical basis with Object Network, so due to space limitations, these basic concepts will not be described in detail here. The interested reader is referred to a complementary reading ([4] or [3]).

Considering the *Fielded Object Networks*, an object is an individual and atomic entity composed by five elements: *attributes, functions, field, connections* and *life cycle*.

2.1 Attributes and functions

Attributes are those predicates that describe the objects. For example, a color might be an attribute of an object that represents a space ship. On the other hand, an object that represents a window is not an attribute of the space ship object, but one of its parts. It is important to note that *attributes* and *parts* are different concepts. In the Fielded Object Network scope, if an object must describe one of its parts, this part will be also an object, and this object is stored into the former object's *field* (see section 2.2). The values of attributes in a particular instant of time constitutes an object's state.

Functions (sometimes called *methods*) are another important concept in all object-oriented approaches. In the Fielded Object Network a function is represented by an algorithm that returns some information and changes the state of the object (based in its own state and in the function parameters).

2.2 Field

Fielded Object Network enhances the traditional concept of object, used by the most popular object oriented analysis (OOA) and design (OOD) methods like UML, OMT or Booch [5] by

introducing three new concepts associated to an object: *field*, *connections* and *life cycle*.

A *field* is considered as a place where the object places the objects that it contains, like a box or a container. The *field* might have any kind of spatial structure and dimension. It can be continuos or discrete, ordered or unordered, finite or infinite.

The *field* might impose rules that affect a subset of attributes of the contained objects. We call those attributes *physical parameters* and the rules that change them *physical laws*. The field might impose physical laws *or not*. If it does, any object into the field must have physical properties compatible with the field structure and its physical laws. It is important to note that in the same way that the field space might have any dimension and ordering scheme, the physical laws can be any kind of rule. For example, if a field is a cartesian 3D-continuous space, the physical laws might be similar to gravitational or electromagnetic fields and forces. If a field is a 2D discrete space, the physical laws might be simple rules between neighbor objects. In this last case, the field predisposes its contained objects to work as a cellular automata. Figure 1 depicts an example of this concept.

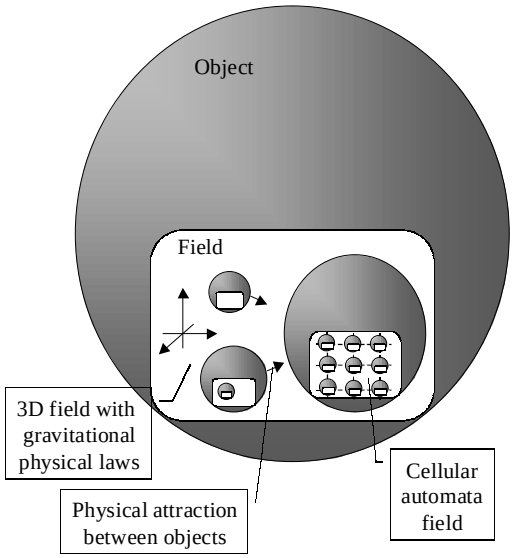

Figure 1 – Field and Embedded Nature

The physical laws are processed into an independent dynamical system called *embedded nature*. We can think the *embedded nature* as an independent process stream (or thread). In fact, in a digital computer, it might be implemented in this way.

If an object has a *field* but does not have an explicit *embedded nature*, we consider this dynamical system to be static. As we will see, an object has many independent asynchronous processing streams. When the *embedded nature* detects a physical (happening within the field) interaction between two or more objects (like a collision), it might warn all objects using a special asynchronous function called *hit function*. In this case a *communication connection* (see section 2.3) may be opened. Later in this paper, we will explain all possible interactions between objects and the synchronization scheme used to make them possible in a digital computer. As we can see, this approach introduces a non-deterministic factor that might be very important on intelligent systems engineering.

In a Fielded Object Network, there is only one object that is not into any field. This object is called *universe object*, being the source of containership of all other objects. In the other hand, objects that do not contain any objects in their field are called atomic objects. The object that contains another one is said to be its *container object*.

2.3 Connections

An object has direct access to its attributes and functions, as well as its contained objects attributes and functions. In both cases the communication between those objects are simple and reliable. As we have already seen in the last section, the Fielded Object Network is topologically ordered. This fact imposes some topological restrictions in all communication cases where this access is not due to a containment relationship. Thus, it is not possible for an object to communicate with another one out of its field without a *connection*. A *connection* is a communication line between two or more objects, wherever they are.

It is important to note that objects can interact in a *logical* or *physical* fashion. The embedded nature and the *hit function* (as explained in the last section) perform the *physical* interaction between objects. A process called *assimilation* is responsible by the *logical interaction* between objects. As we will see, there are three different assimilation types. In all of them, an object might assimilate another one if and only if there is a *connection* between theirs container's field. In this sense, we consider that if an object is within a field, there is a natural connection from it and its container. Figure 2 shows some examples. The connections are graphically represented as links between objects. The connection might be directed or undirected.

As we will see in the next section, all objects, in the worst case, have at least two independent processing streams (*embedded nature* and *life cycle*). When implementing a FON into a digital computer, this fact imposes that some care should be taken to ensure its data and structural consistency. All communication mechanisms are defined as atomic operations and are protected by binary semaphores to properly deal with race conditions and to avoid deadlocks. Due to practical reasons, to allow this safe communication mechanism in the FON, an additional restriction is applied: an object must explicitly allow its assimilation before being assimilated and it should specify the allowed assimilation type.

There are three different assimilation types. The first one is a *destructive assimilation*. In this case, the object is moved from the source field to the destination one. The second assimilation type is the *non-destructive* where the object is cloned (copied) in the destination field. The latter and more complex assimilation mechanism is the *message passing*. In the message passing mechanism the assimilation is performed with a help of a third object called *messenger* that goes to the source field, physically

attach itself to the assimilated object, modify its state based on assimilated object state and then return to the original field. This is necessary when we do not want to get a whole object, but only part of its structure.

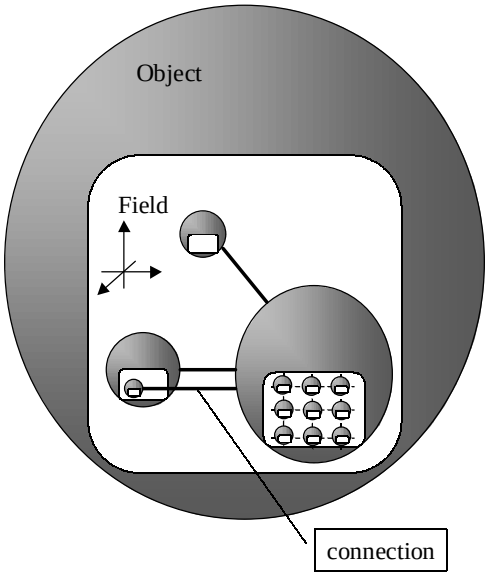

Figure 2 – Connection

2.4 Life cycle

The last and maybe the most important FON object characteristic is the concept of *life cycle*. A *life cycle* is an independent and parallel dynamic system that is ruled by some kind of algorithm while the object exists. If the *life cycle* does nothing, or does not exist, the object is called *passive*, otherwise it is called *active*. A *life cycle* can be assigned dynamically to an object. In this sense, an object can initially be passive and become active, or be active and suddenly become passive. We say, when we assign a *life cycle* to an object, that we are giving life to it.

3. WHY AND WHEN FIELDED OBJECT NETWORK?

Distributed Artificial Intelligence (DAI) corresponds to the intersection of Distributed Computing and Artificial Intelligence. In Distributed Computing, several processors share data, but not control. It focuses on low-level parallelization and synchronization issues [9]. In DAI the intelligent control as well as data is distributed and it focuses on problem solving, communication and coordination. Recently, people tend to break DAI into two classes: Distributed Problem Solving (DPS) and Multiagent Systems (MAS) [11][9], despite that the boundaries between them is still not clear. In DPS a complex task is decomposed and distributed. A solution is synthesized by collecting all subtasks partial results. DPS focuses on information domain management. Usually in DPS domain, strong assumptions are performed to ensure compatibility of different problem-solving entities. Those restrictions normally do not apply on MAS systems.

As we already said, an object has, in the worst case, two independent processing streams: the *embedded nature* and the *life cycle*. In a digital computer implementation, those processing streams might be independent and the connection communication mechanism can be implemented aiming a distributed communication (e.g. using TCP/IP, CORBA or PVM). It makes the FON a Distributed Computing tool oriented for Distributed Artificial Intelligence engineering, more specifically, for Multiagent Systems. This paper will focus on how the Generalized Subsistence Machine (GSM) [6] can be mapped using the FON architecture aiming a DAI architecture grounding. Another interesting research theme associated to FON is Artificial Life. This is to be addressed in a future paper

4. A FON APPROACH TO GSM

"Generalized Subsistence Machine (GSM) is defined as a system which is unified by a goal to *exist as an entity*" [6] and has been proposed by Meystel as a basic unit of an intelligent system. One of the GSM merits relies on its multiresolutional system of entities. A GSM structure can be depicted in Figure 3 where "P" means Perception; "WM" means World Model and "BG" means behaviour generator; "S" means sensors; "W" means world and "A" means actuators.

Due to space limits imposed to this paper, we will not discuss the reliability of a GSM. For a completed description of it and its entities please refer to [6] and [1]. A review of the importance of the multiresolutional approach to artificial intelligent systems can be found at [6] and [8].

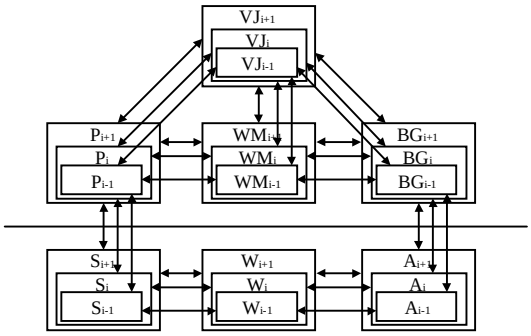

Figure 3 – Multiresolutional GSM

Taking the advantage of FON parallelism and scalability a GSM can be implemented as shown in Figure 4.

Sometimes the Value Judgement (VJ) element (object) can be incorporated into Behavior Generation and World Model elements (objects). It is important to note that using the FON approach the multiresolutional characteristic of the GSM structure is improved by another one: the parallelism. One of the GSM claims is that the world is hierarchical, thus, a hierarchical modelling would achieve better results. Following this thought, we claim that the world is massively parallel, thus, a parallel approach is important too (this

is one of the DAI claims) but it would be naive to think that a multiresolutional and parallel approach is adequate for all cases. This fact turns evident one of the most important FON advantages: the objected-oriented encapsulation concept. Using this concept an MAS can be conceived with GSM and non-GSM agents together. More than that, an agent can be constructed with GSM and non-GSM modules that work together in a cooperative or competitive fashion and each one can use different knowledge representation and processing approaches.

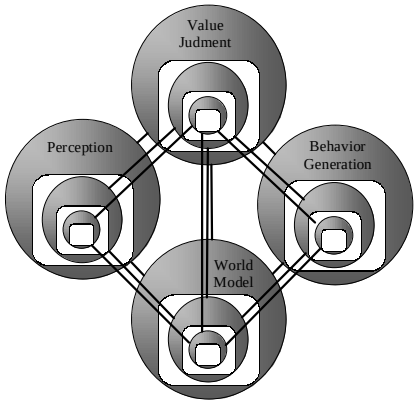

Figure 4 - GSM as a FON

5. AN EXAMPLE: AUTONOMOUS GUIDED VEHICLE

As an example of the FON and GSM reliability we implemented an example considering an Autonomous Guided Vehicle (AGV – See Figure 5). The AGV is an autonomous vehicle without a prior knowledge of the environment. It runs in order to achieve some goals like avoiding obstacles and reaching desired points in the environment. This is not a trivial example but, also not complex enough to cover all FON capabilities. Nevertheless, the idea is to focus on the hierarchical and parallel characteristics and how they can be used to solve the AGV problem.

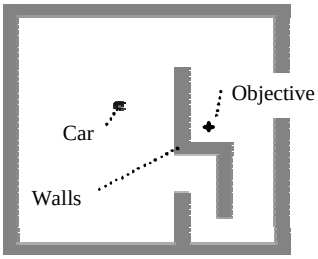

Figure 5 - AGV

Figure 6 shows an overview of the FON used in this example.

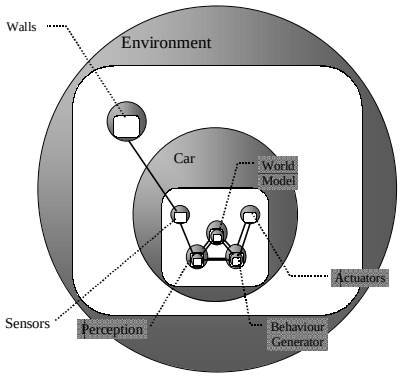

Figure 6 - AGV structure

The *field* of the "environment" object is a 2D-continuous space and its *embedded nature* corresponds to the physical laws derived from classical mechanics. All other objects do not have embedded nature (this example does not cover all FON potentialities). The AGV GMS can be viewed in details in Figure 7 where the BG, P and WM indexes "LL", "ML" and "HL" means, respectively, low level, medium level and high level.

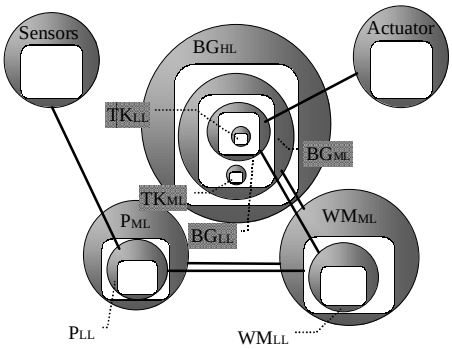

Figure 7 – GSM

The low-level perception object (P_{LL}) uses an ultrasonic sensor to test the presence of obstacles, inside of its range, in a certain direction. To difficult things to the AGV, the ultrasonic sensor's range is limited to four times its diameter. The medium-level perception object (P_{ML}) uses the P_{LL} to find out where the obstacle starts and where it ends getting data in a lower resolution (higher scale).

The low-level world model object (WM_{LL}) uses the P_{ML} to build a model of the environment that surrounds the AGV and warn the low-level behavior generator (BG_{LL}) if a collision is imminent. The medium-level world model object (WM_{ML}) uses the WM_{LL} and the P_{ML} to continuously build and verify a world model.

The high-level behavior generator object, in this example, does only provide a graphical interface to allow users to create medium-level tasks (TK_{ML}). In future works, this element will take care of higher order objectives like energy consumption and objective point definition. A medium-level task is something like "go to a specific place without any collision". Once TK_{ML} object is

created into the medium-level behavior generator object (BG_{ML}), it uses the WM_{ML} to create a plan (Figure 8). This plan is an attribute of the BG_{ML} object and is formed by a list of subtasks. Each subtask is a low-level task (TK_{LL}) and is created by the BG_{ML} into the BG_{LL} field.

A TK_{LL} is something like "go to a specific point" or "stop". The BG_{LL} object sends commands to the actuator object in order to increase or decrease the AGV speed, to modify the steering angle or to rotate round itself. If any BG object receives a collision warning from the WM object, they send a stop command to the actuator object. If the WM_{ML} detects any important change in the world model, in other words, if it detects any change that conflicts with the viability of the BG_{ML} path planning it immediately warns the BG_{ML} to recalculate the plan. The Figure 9 depicts the final result.

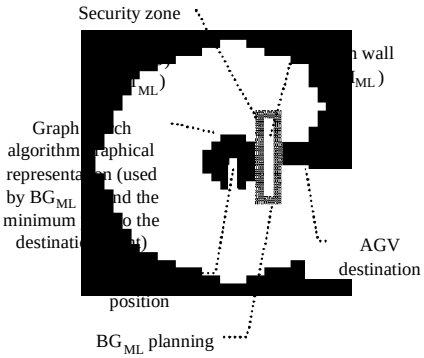

Figure 8 – BG_{ML} planning

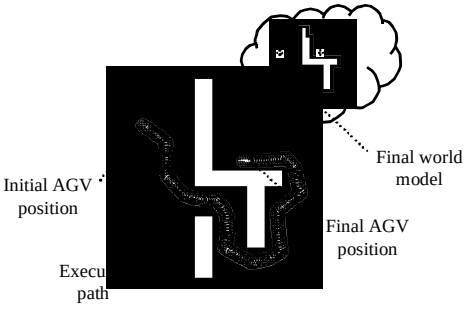

Figure 9 - Final result

6. CONCLUSION

In this paper we presented the main concepts of the Fielded Object Networks. The FON intends to be a tool in an autonomous intelligent systems engineering methodology. This methodology is still a work in progress but many of its ideas and concepts were introduced in [2]. This approach is akin to the most recent developments in the literature of intelligent systems, and shows a promising methodology to modeling, analysis and design of such class of systems. At the same time, the FON object-oriented flavor also suggests a direct methodology for the computational implementation of these systems and the development of fourth generation software tools for autonomous intelligent systems.

In a future work, we plan to formalize the presented concepts, and explain how the FON and ON can be used into the development and construction phases of an autonomous intelligent systems engineering methodology.

7. REFERENCES

[1] Albus, J.S., *Outline for a Theory of Intelligence*, IEEE Transactions on System Man and Cybernetics, Vol. 21, No. 3, May/June 1991.

[2] Gonçalves, R., Gudwin, R., Gomide, F., *Semiotic Oriented Autonomous Intelligent Systems Engineering*. Proceedings of the ISAS'98 – Inteligent Systems and Semiotics – Gaithersburg, MD, USA – Semptember, 1998.

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

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

[5] Larman, C., *Applying UML and Patterns – An Introduction to Object-Oriented Analysis and Design*, Prentice-Hall Inc, New Jersey, 1998.

[6] Meystel, A. M., *Intelligent Systems: A Semiotic Perspective*, International Journal of Intelligent Control and Systems, Vol. 1, No. 1, pp. 31-57, 1996.

[7] Peirce, C. S., *Collected Papers of Chales Sanders Peirce*. Edited by Charles Hartshorne and Paul Weiss – Belknap Press of Harvard University Press – Cambridge, Massachussets, 2nd printing, 1960.

[8] Starks, S.A., Kreinovich, V., Meystel A., *Multi-Resolution Data Processing: It is Necessary, It is Possible, It is Fundamental*. Proceedings of the ISAS'97 – Inteligent Systems and Semiotics: A Learning Perspective - Gaithersburg, MD, USA – 22-25 September, 1997.

[9] Stone, P., Veloso, M., *Multiagent Systems: A Survey from a Machine Learning Perspective*. Carnegie Mellon University CS technical report number CMU-CS-97-193.

[10] Tarasov, V., *An Artificial Life View on Intelligent Enterprises Fuzzy Evolutionary Multi-Agent System.* Proceedings of the ISAS'97 – Inteligent Systems and Semiotics: A Learning Perspective – Gaithersburg, MD, USA – 22-25 Semptember, 1997.

[11] Weiβ, G., Sen, S., Adaptation and Learning in Multiagent Systems. Springer Verlag Inc., Berlin, 1996.