IGI PUBLISHING **ITJ4003**

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.igi-global.com

This paper appears in the publication, International Journal of Cognitive Informatics and Natural Intelligence, Volume 2, Issue 1
edited by **Yingxu Wang © 2008, IGI Global**

Testing the Behaviour of Entities in a Cognitive Language[1]

Alberto de la Encina, Universidad Complutense de Madrid, Spain

Mercedes Hidalgo-Herrero, Universidad Complutense de Madrid, Spain

Pablo Rabanal, Universidad Complutense de Madrid, Spain

Ismael Rodríguez, Universidad Complutense de Madrid, Spain

Fernando Rubio, Universidad Complutense de Madrid, Spain

ABSTRACT

We present a programming environment to help study the behavior of cognitive models. The core of the environment is a programming language based on high-level constructions that allow the easy development of general schemes that can be used (and even modified) without requiring knowledge about the underlying language. In addition, we have introduced observation facilities to the language to help analyze how each process takes its own decisions. Thus, we can analyze not only the external behavior of each entity, but also how it constructs its knowledge. Moreover, our framework also provides visualizing facilities.

Keywords: *cognitive informatics; functional languages; learning; simulators*

TESTING THE BEHAVIOUR OF ENTITIES IN A COGNITIVE LANGUAGE

The relation between the computer science and other older well established sciences is known to be twofold. On the one hand, the computer science provides tools, automatic analysis techniques, and even new concepts to other sciences, which enables the construction of new methodologies and theories in these areas. On the other hand, many methods used in computer science are inspired in the nature. Examples of this relation are ants-based algorithms and neural networks. This twofold relation is particularly fruitful in the scope of cognitive science, which exploits the similarities between natural and artificial reasoning processes. In particular, the roots of several artificial intelligence algorithms are based on the cognitive processes of the brain, while simulating a cognitive process in a computational environment may help to validate/refute a theory in the field of Neuroscience.

There is a topic that has been typically concerned by informatics scientists that, actually, fits perfectly in the context of cognitive informatics (Zhang et al., 2007; Wang, 2002;

Wang & Kinsner, 2006). This is the relation between *artificial* cognitive processes and *human* cognitive processes. This issue has been addressed for years by computer scientists, and it is one of the main topics of interest in artificial intelligence. One of the first approaches proposed to relate both concepts was made by Alan Turing and it consists in a nowadays classical test. In brief, the *Turing test* (Saygin, Cicekli, & Akman, 2000; Turing, 1950) says that a system should be considered *intelligent* if its *behavior* cannot be distinguished from that of a human being.

Following this idea, in the long term we are specially interested in being able to compare how both automatic systems and humans learn the rules governing their environment. Let us note that comparing the learning processes of both humans and machines is hard because it requires to compare the cognitive theories each of them builds and refines along time. This enables the comparison of the levels of accurateness of each of them along time.

In this article, we will concentrate on developing an environment to help analyzing the behavior of an automatic system. We will be able to observe its external behavior by recording the relations between its input stimuli and its output responses. However, the most interesting part is that we will also be able to observe its internal behavior. That is, we will observe how the entity *constructs* its responses to the input stimuli.

In order to obtain our objectives, we will consider an specific programming environment. The main part of our programming environment is its core language. Eden (Klusik, Loogen, Priebe, & Rubio, 2001; Loogen, Ortega-Mallen, Pena, Priebe, & Rubio, 2002; Rubio & Rodríguez, 1998) is a parallel extension of the functional language Haskell (Jones & Hughes, 1999) which provides "controlled" parallelism, as opposed both to more implicit approaches such as GpH (Glasgow Parallel Haskell) (Trinder et al., 1996, 1998) and to more explicit ones such as *Concurrent ML* (Reppy, 1991). In this sense, in Eden the programmer decides which expressions are processes and how processes are connected, having explicit control over the communication topology. The remaining aspects such as sending and receiving messages, process placement, data distribution, and so forth, are implicitly controlled by the runtime system. The parallel constructions of Eden allow to easily specify any topology. This is specially important when simulating the complex topologies that can appear in cognitive models. Moreover, Eden programming has the advantages of functional languages. That is, its high-level constructions simplifies the task of developing programs, and it facilitates the creation of executable templates that reduce the time needed to develop applications.

The structure of the rest of the article is the following. In the next section we review some basic learning theories. In the second section we outline the main characteristic of our language. Then, in the third section we show how to implement cognitive models. Afterwards, the fourth section describes the basic details of our observing facilities, while the fifth section contains a detailed description about how to perform observations in a systematic way. Finally, in the sixth section we present our conclusions and some lines for future work.

LEARNING THEORIES

Before describing what kind of learning analysis we will be able to perform with our programming environment, we must remark that we are specially interested in being able to analyze how an automatic system can deduce the rules governing a situation.

We will base our observations and the players tasks on two main theories of learning: behaviorism and constructivism. Before explaining how we use these models let us explain them:

Behaviorism: the aim of this theory is to use experimental methods to observe the behavior of the subject. With respect to learning, this premise leads to consider that learning happens just when a correct response is given after the presentation of a stimulus, and the trainer can detect whether the subject has learned or not by observing his or her behavior over a period