Robocontrol 2010
4º WORKSHOP DE ROBÓTICA APLICADA E AUTOMAÇÃO

unesp

AN INTRODUCTORY EXPERIMENT WITH A CONSCIOUS-BASED AUTONOMOUS VEHICLE

Ricardo Capitanio Martins da Silva[1], *Ricardo Ribeiro Gudwin*[1]

[1]DCA-FEEC-UNICAMP
Av. Albert Einstein 400
13.083-852 Campinas, SP
{martins,gudwin}@dca.fee.unicamp.br

Abstract: This work describes an introductory experiment where the Baars-Franklin Architecture (BFA), an artificial consciousness approach, was applied to synthesize a control system for an autonomous vehicle. The BFA was reported in the literature as a successful control system to different kinds of agents: CMattie, IDA and CTS. In this paper, BFA is for the first time applied for controlling an autonomous vehicle. Firstly we introduce the theoretical foundations of this approach for the development of a conscious agent. Then we explain the architecture of our agent and at the end we discuss the results and first impressions of this approach.

Keywords: artificial consciousness, intelligent systems, autonomous vehicle, multi-agent systems

1. INTRODUCTION

The claim to be building a "conscious" software agent is a very hard one, from both a scientific and an engineering point of view. Nevertheless, in the last ten years there has been an intensive growth in the scientific study of consciousness [1, 2]. A technological offspring of these studies is the field of artificial consciousness [3–7]. Among the many computational approaches to consciousness, the framework developed by Stan Franklin, at the University of Memphis [4, 8–10], based on the model of consciousness given by Bernard Baars, called Global Workspace Theory [11] is of a special interest, due to the fact it is firmly rooted onto a sound background given by a respected scientific theory of consciousness. Together, Baars (from a scientific perspective) and Franklin (from an engineering perspective) grew up an interesting framework which could be viewed, at the same time, as a proof of concept of a scientific theory, and a new approach for the development of computational systems. In this work, our aim was to understand and test what we are calling here the Baars-Franklin architecture (BFA), a computational architecture being jointly developed by Franklin and Baars during the last 10 years.

The BFA has already been applied to many different kinds of software agents. The first application of BFA was CMattie [4, 8], an agent developed by the Cognitive Computing Research Group (CCRG) at University of Memphis, whose main activities were to gather seminar information via email from humans, compose an announcement of the next week's seminars, and mail it to members of a mailing list. Through the interaction with human seminar organizers, CMattie could realize missing information and ask it via email.

The overall BFA received major improvements with subsequent developments. One remarkable implementation of it was IDA (Intelligent Distribution Agent) [12], an application developed for the US Navy to automate an entire set of tasks of human personnel agent who assigns sailors to new tours of duty. IDA is supposed to communicate with sailors via email and, in natural language, understand the content and produce life-like messages.

The BFA was also used outside of Franklin's group. Daniel Dubois from University of Quebec developed CTS (Conscious Tutoring System) [13], a BFA-based tutoring system applied in the training of astronauts in manipulating Canadarm2, the robotic control system at the International Space Station.

The main motivation for our study was to understand, after all, what exactly it is for an agent to be "conscious", and what are the advantages of consciousness, from an engineering point of view (at least regardig the BFA approach). In order to grasp this understanding, and create a value judgment on the technology, we applied BFA in the control of a simple autonomous vehicle, a kind of application which was never reported since so far as being held by the BFA. The control of an autonomous vehicle pose some interesting research problems when compared to other kinds of software agents where BFA has already been tested. In the original applications where BFA was tested, the perception system is based on the exchange of e-mail messages (the case of CMattie and IDA), and interactions in a HCI (human-computer interface), in the case of CTS. In the control of an autonomous vehicle, perception must rely on remote (e.g. visual, sonar, etc) and/or local (e.g. contact) sensors, capturing world properties and interpreting them in order to create a world model. The behavior generation module is also different, as the agent must act on the environment by means of actuators, and so a whole motor system must be built. In order to focuse on the understanding of the main capabilities of the BFA and avoid unnecessary second-order effects caused by real autonomous vehicles, we decided to apply it in a simulated autonomous vehicle, and not to a real one. This decision is though just a first step. We realize that this is just a first contact with a new kind of technology, and we envision further experiments, in the future, with real autonomous vehicles. It is important to say, here, that the application is not the main focus of our experiment, but much more how the BFA can be adapted in

order to be used in this context.

In next section, we briefly introduce Baars theory of consciousness, Global Workspace Theory, and in the sequence we describe how we customized BFA in order to deal with an autonomous vehicle. After that, we introduce CAV (Conscious Autonomous Vehicle), the simulated vehicle we used in our study and its environment, and a brief analysis of the results of our simulations using CAV.

2. GLOBAL WORKSPACE THEORY AND BFA

Bernard Baars has developed the Global Workspace Theory (GWT) [11, 14] inspired by psychology and based on empirical tests from cognitive and neural sciences. GWT is an unifying theory that puts together many previous hypothesis about the human mind and human consciousness.

Baars postulates that processes such as attention, action selection, automation, learning, metacognition, emotion, and most cognitive operations are carried out by a multitude of globally distributed unconscious specialized processors. Each processor is autonomous, efficient, and works in parallel and high speed. Nevertheless, in order to do its processing, each processor may need a set of resources (mostly information of a specific kind), and at the same time, will generate another set of resources after its processing. Specialized processors can cooperate to each other forming coalitions. This cooperation is by means of supplying to each other, the kinds of resources necessary for their processing. They exchange resources by writing in and reading from specific places in working memory. Coalitions may form huge complex networks, where processors are able to exchange information to each other. But processors within a coalition do have only local information. There may be situations, where the required information is not available within the coalition. To deal with these situations, and allow global communication among all the processors, there is a global workspace, where processors are able to broadcast their requirements to all other processors. Likewise, there may be situations where some processor would like to advertise the resource it generates, as there may be other processors interested in them. They will also be interested in accessing the global workspace and broadcasting to all other processors. In the broadcast dynamics, only one coalition is allowed to be within the global workspace in a given instance of time. In order to decide which coalition will go to the global workspace in a given instant of time, a whole competition process is triggered. Each processor has an activation level, which expresses its urgency in getting some information or the importance of the information it generates. A coalition will also have an activation level which is the average of activation levels of its participants. At each time instant, the coalition with the highest activation level will win the access to the global workspace. Once a coalition is within the global workspace, all its processors will broadcast their requests and the information they generate. The broadcast mechanism do allow the formation of new coalitions, and also some change in working coalitions.

For Baars, consciousness is related to the working of this global workspace. Processors are usually unconscious, having access only to local information, but in some cases they may require or provide global information, in which case they request access to consciousness, where they will be able to broadcast to all other processors. This is the case when they have unusual, urgent, or particularly relevant information or demands. This mechanism supports integration among many independent functions of the brain and unconscious collections of knowledge. In this way, consciousness plays an integrative and mobilizing role. Moreover, consciousness can be useful too when automatized (unconscious) tasks are not being able to deal with some particular situation (e.g. they are not working as expected), and so a special problem solving is required. Executive coalitions, specialized in problem solving will be recruited then in order to deal with these special situations, delegating trivial problems to other unconscious coalitions. In this way, consciousness works like a filter, receiving only emergencial or specially relevant information.

Inspired on Baars description of his theory of consciousness, and also on previous work in the computer science literature, Franklin proposed a framework for a software agent which realized Baars theory of consciousness, in terms of a computational architecture, constituting so what we are calling here the Baars-Franklin architecture. In specifying BFA, Franklin used the following theories as background, among others not detailed here: Selfridge's Pandemonium [15] and Jackson's extension to it [16], Hofstadter and Mitchell Copycat [17] and Maes' Behavior Network [18].

From Hofstadter's Copycat, Franklin borrowed the notion of a "Codelet" (and also the Slipnet, for perception). He noticed that these *codelets* were more or less the same thing as Selfridge's "demons" in Pandemonium theory and also a good computational version for Baars *processors*. Jackson's description of an arena of *demons* competing for selection will fit as well Baars description of processors competing for access to consciousness. Using these similarity, Franklin set up the basis of BFA: cognitive functions are performed by coalitions of codelets working together unconsciously, reading and writing tagged information to a Working Memory. Each codelet has an activity level and a tagged information. A special mechanism, the *Coalition Manager* will manage coalitions and calculate the activity level of each coalition. Another special mechanism, the *Spotlight Controller*, will be evaluating each coalition activity level, and defining the winning coalition. Also, the *Spotlight Controller* will be responsible for performing the broadcast of the tagged information of each codelet in the winning coalition, to all codelets in the system. The agent behavior is decided using a Behavior Network, whose propositions are related to the tagged information in the Working Memory.

Unfortunately, a full description of BFA is beyond the space available in this text. We refer the interested reader to [4, 10, 13, 19], where a more detailed description of BFA

is available. Some background in the auxiliary theories we mentioned above is provided in the sequence.

2.1. Pandemonium Theory

Selfridge's Pandemonium Theory is a connectionist architecture originally used for pattern recognition. Selfridge [15], influenced by the parallelism of human data processing, suggested a parallel architecture composed of multiple independent processes called *demons*. Each demon works simultaneously recognizing specific conditions (or a set of them). Demons have links that allows them to "call" other demons.

A pandemonium can be used to recognize letters. For example, an "R" letter can be decomposed into features like a "belly" on the upper right, one vertical line and a "leg" on the lower right. When *"feature demons"* whose names are "vertical", "belly" and "leg" listen to their names being called, they start to shout out to the *"cognitive demons"*. Since some features can individually be present in several patterns, more than one cognitive demons may be sensitized, "B-demon" and "D-demon" responsible respectively for B and D. It happens because B-demon and D demon also listen to belly and vertical demons. However, the R-demon will shout proportionally to the degree of "R-ness" it have heard from the feature demons. It will be the loudest demon once it can hear all feature demons even when B-demon and D-demon are shouting out to the *"decision demon"*. So, the decision demon recognizes the pattern as "R".

John Jackson extended the original Pandemonium theory of perception by creating the stadium metaphor, organizing demons in two different locations, the equivalent of stands and arena of a stadium. Jackson [16] proposed a system consisted of a crowd of usually dormant demons located at the stands, from where a few demons could go down to the arena and start exciting the crowd. Some demons in the crowd gets more excited and starts to yell louder. If the activity of demons in the arena drops below a threshold they may return to the stands and the loudest demons in the crowd replace them. Besides the crowd get excited watching the demons in the arena, the last ones can spread activation to the formers through links. These connections between demons are created or strenghened according to the time they are together on the arena, following a Hebbian learning scheme.

2.2. Copycat Architecture

Copycat is a hybrid symbolic-connnectionist architecture that is intended to model analogy making along with recognition and categorization. It was developed by Hofstadter and Mitchell [17] with the premise that analogy making is a process of high-level perception. Copycat makes and interprets analogies between situations in a predefined and fixed domain like letter-string analogy problems.

Those analogies emerge from the activity of many independent processes, called *codelets*, running in parallel, sometimes cooperating, sometimes competing with each other. Copycat starts with a fixed number of codelets in a codehack, predetermined by the designer.

Codelets count with an associative network (the *Slipnet*) that contains interrelated concept types (*nodes*) and links between them. Codelets look for specific words or parts of words and if they find them they activate some nodes of the Slipnet. Nodes can vary in their level of activation which is a measure of relevance to the current situation. They spread some activation to neighbors and lose activation by decay. The Slipnet is a long-term memory and represents what Copycat knows. It does not learn anything during execution.

At last, Copycat has a working memory where perceptual structures are built and modified. At each moment the content of the working memory represents Copycat's current perception of the situation it is facing.

2.3. Behavior Network

Pattie Maes [18] developed a behavior-based action selection mechanism, built as a society of behaviors or competence modules in a distributed, recurrent, non-hierarchical network. This network is formed by four kinds of nodes. The first kind of node (and the most important) represents a low level behavior (e. g. approach food, drink water, walk around). The second kind of node represents propositions (or predicates e.g. glass-on-hand, glass-with-water-inside, glass-empty), which can be true or false. The third kind of node represents goals (or motivations). The fourth kind of node represents sensors from the environment.

Sensor nodes are linked to proposition nodes. Behavior nodes are input linked from *preconditions* propositions which must be true for the behavior to be executable. In its output, they are linked to two possible kinds of propositions: *add* propositions, which are expected to become true after the behavior is executed, and *delete* propositions, which should be set to false after the behavior is executed. For example, a behavior *"drink water"* could have the preconditions *glass-on-hand* and *glass-with-water-inside*. Its add list could contain *glass-empty* and the delete list would contain *glass-with-water-inside*. Goal nodes are linked to proposition nodes, which are backward linked to behavior nodes. See figures 3 and 4, further, for an example of the connection among links. In these figures, triangles are proposition nodes, ovals are behavior nodes, round squares are sensor nodes and pentagons are goal nodes.

The network executes as follows. Each behavior do have an activation level, which is changed by two waves of spreading activation: one from sensor nodes forward and the other from goal nodes backwards. The first one spreads activation forward from sensor nodes to propositions which are evaluated (true or false) according to the environment situation and from them forwards to behavior nodes which need these predicates to be true to be fired. The second spreads activation backwards from goal nodes to predicate nodes and then

to behaviors which can satisfy these goals. More details on the spreading mechanism can be found in [10, 18]. At the end, after all the energy is spread-up, the behavior which remains with the highest activation level is chosen to be executed. Only one behavior is chosen to be executed at each operational cycle.

The energy is spread internally through the network as follow:

- each executable node spreads activation energy to its successors
- each non-executable node spreads activation energy to those predecessors that will make a false precondition become true
- each node (executable or not) decreases the activation level of its conflictors.

The global algorithm performs as follow at each time step in the loop:

1. The environment and global goals inject activation to the behaviors and protected global goals take away activation from behaviors.
2. Activation levels are normalized at the behaviors so that the average activation level in the network remains constant.
3. Behavior is active if it fullfils the conditions: (i) it is executable; (ii) its activation level is over threshold; (iii) it has the highest activation level among others that satisfy conditions (i) and (ii).
4. A selected behavior, after it finished its action will reset its activation level to zero and threshold level is reset to default.
5. If no behavior is selected, reduce the threshold by 10%.

3. Our implementation of BFA

In our experiment, we developed a conscious-based control system, which we call CAV - *Conscious Autonomous Vehicle*, to control a simulated autonomous vehicle in a virtual environment (see figure 1). The simulated vehicle and its environment were originally presented in [20] (where more details in its characteristics can be obtained) and were adapted for our current studies. In this environment, the autonomous vehicle is equiped with sensors and actuators, which enable it to navigate through an enviroment full of objects with different characteristics. An object can vary in its "color" and each color is linked to: a measure of "hardness" which is used in the dynamic model as a friction coefficient that can slow down the vehicle movement (or completelly block it), a "taste" which can be bad or good, and a feature related with "energy" which

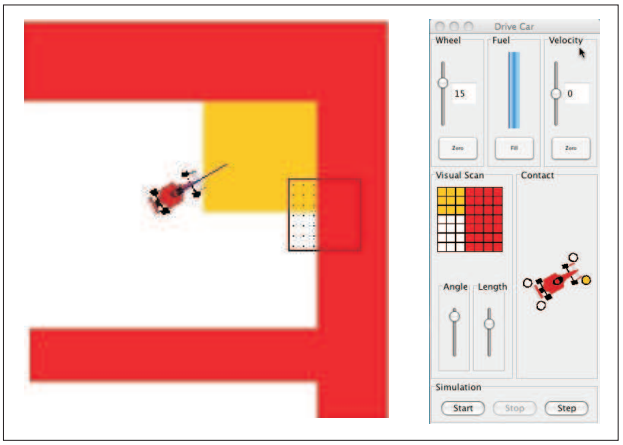


Figure 1: Sensory-motor structure of the autonomous vehicle

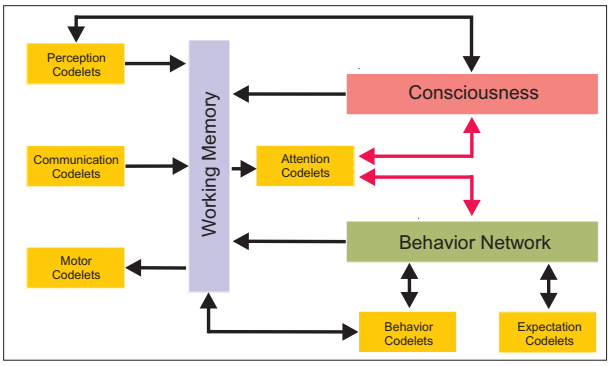


Figure 2: *CAV*'s Architecture

indicates that the object drains/supplies energy from/to the vehicle's internal rechargeable battery.

The vehicle connects to its control system through sockets. In this sense, the control system is a completelly separate process, which can be run even in a different machine. So, different control systems can be attached to the vehicle and tested for the exact same situation.

When the simulation is started, the vehicle builds an incremental map of the environment based on the sensorial information. Our agent adds landmarks to this map and uses them to generate movement plans. It has two main motivations: it should navigate from an initial point up to a target point, avoiding collisions with objects; and it should keep its energetic balance, taking care of the energy level in the internal batteries.

Our agent architecture (see figure 2) is essencially rooted in the BFA implementation as in [12], more specifically the behavior network [10] and consciousness [4] modules. *CAV* brings some modifications in the implementation related with the application domain, and the interaction among consciousness and behavior network. The following sections contain a brief description of CAV's modules.

3.1. *Codelets*

CAV is heavily dependent on small pieces of code running as a separate threads called codelets (BFA borrows this name from Hofstadter's Copycat). Those codelets correspond pretty well to the specialized processors of global workspace theory or demons of Jackson and Selfridge.

BFA prescribes different kinds of codelets such as attention codelets, information codelets, perceptual codelets and behavior codelets. In addition to that, it is possible to create new types of codelets depending on the problem domain. CAV's domain does not require string processing as most other BFA applications. Instead of that, the vehicle state is well divided in registers at the working memory. It is possible to have access to all variables anytime. Because of this, CAV does not use information codelets which in BFA are used to represent and transfer information. We have two kinds of behavioral codelets: the behavior codelets, linked with the nodes of the Behavior Network and responsible for "what to do", and motor codelets, which know "how to act" on the environment. With this in mind CAV has the taxonomy of codelets presented at Table 1.

3.2. *Working Memory*

The working memory consists of a set of registers that are responsible for keeping temporary information. The major part of the working memory is related to the vehicles' status. The communication codelet constantly overwrites the registers like speed, wheel degree, sensorial information and vehicle position. CAV's working memory works also as an interface among modules, for example, between consciousness and the behavior network. Some codelets, including attention codelets watch what is written in the working memory in order to find relevant, insistent or urgent situations. When they find something, they react in order to compete for consciousness. Whenever one of then reaches consciousness, its information will influence the agent's actions.

3.3. *Consciousness mechanism*

The consciousness mechanism consists of a *Coalition Manager*, a *Spotlight Controller*, a *Broadcast Manager* and attention codelets which are responsible for bringing appropriate contents to "consciousness" [4]. In most of the cases, codelets are observing the working memory, looking for some relevant external situation (e.g. a low level of energy). But some codelets keep a watchfull eye on the state of the behavior network for some particular occurrence, like having no plan to reach a target. More than one attention codelet can be excited due to a certain situation, causing a competition for the spotlight of consciousness. If a codelet is the winner of this competition, its content is then broadcasted to the registered codelets in the broadcast manager. We have three main differences between standard BFA and CAV, related to this module. The first one is that we don't use information codelets. The second is that not all of the codelets are notified like in BFA, just the registered ones. At last, some codelets can be active outside of the playing field. In this case their contents will never reach consciousness.

3.4. *Behavior Network*

CAV's behavior network is based on a modified version of [18] by [10]. Negatu adapted Maes' behavior network so each behavior is performed by a collection of codelets. Negatu's implementation also divided the behavior network in *streams* of behavior nodes.

The behavior network works like a long-term procedural memory, a decision structure and a planning mechanism. It coordinates the behavior actions through an "unconscious" decision-making process. Even so it relies on conscious broadcasts to keep up-to-date about the current situation. This is called "consciously mediated action selection" [10].

CAV uses two main behavioral streams, the *Target* stream and the *Energy* stream, as in figures 3 and 4.

3.5. *Cognitive Cycle*

In GWT, all codelets and the consciousness mechanism are asynchronous and parallel processes. On the first implementations of BFA, these were all implemented by completely asynchronous threads. Nevertheless, due to many synchronism problems among codelets, further implementations of

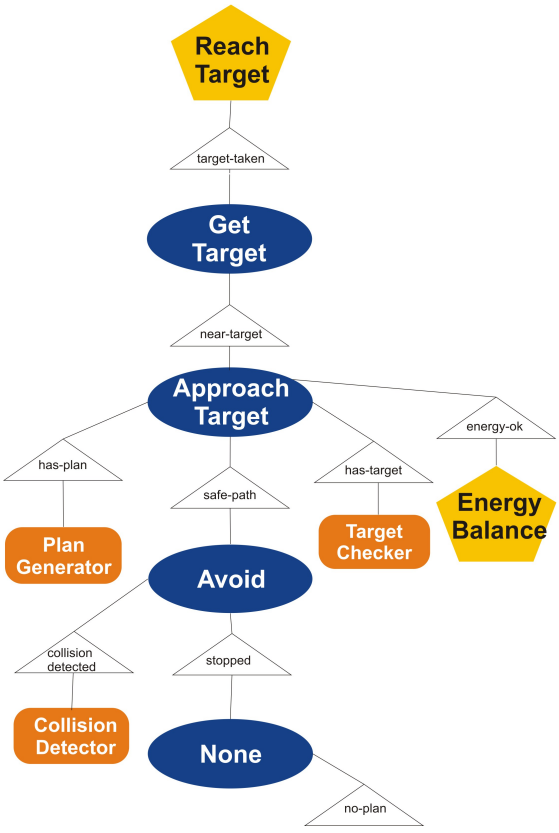


Figure 3: Behavior Network - Target Stream

Table 1: ***CAV's Codelets Taxonomy***

Type	Role
Communication	Perform the communication with the simulator, bringing novel simulation information
Perception	Give an interpretation to what the agent senses from its environment
Attention	Monitor the working memory for relevant situations and bias information selection
Expectation	Check that expected results do happen
Behavior	Alter the parameter of the motor codelet
Motor	Act on the environment

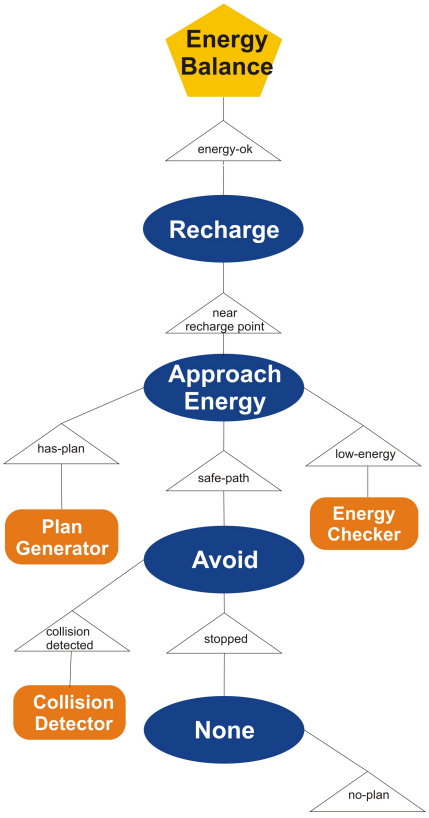


Figure 4: Behavior Network - Energy Stream

BFA prescribed the creation of a *Cognitive Cycle*. This cycle imposes some synchronism points on codelets threads, and organizes the interaction among BFA's components in the form of an operational cycle. This solved synchronism issues of the multi-thread environment and made less dificult the computational implementation without detriment of the main ideas in GWT.

CAV's cognitive cycle (CCC) brings significant differences when compared to standard BFA's one[1]. We removed the first three original steps: perception (interpretation of sensory stimuli), percept to preconscious buffer (the percept is stored in working memory), local associations (retrive local associations from transient episodic memory (TEM) and long term associative memory (LTM)). This last one is quite obvious once CAV does not have an implementation of TEM or LTM. In the other cases, the removal of the two first steps is related to the problem domain. CAV does not process streams of characteres like IDA. So CAV does not need a Slipnet. Moreover the input data of CAV is well structured as working memory's registers can be updated anytime. It guarantees that all codelets will handle the most possible up-to-date input data. The "recruitment of resources" step has also been removed, because the "answer" of all listening codelets happens in parallel with the cycle, not inside of it.

[1]For standard BFA's cognitive cycle see [21].

The remaining CCC five steps are summarized below (adapted from [21]. We will indicate major accordances with standard BFA with sentences written in italics):

Competition for consciousness

Attention codelets, whose job is to bring relevant, urgent, or insistent events to consciousness, access working memory and the behavior network state. Some of them gather information and actively compete for access to consciousness. The competition may also include attention codelets from recent previous cycle.

Conscious broadcast

A coalition of codelets (possibly with just a single codelet) gains access to the global workspace and has its contents broadcasted. This broadcast is hypothesized to correspond to phenomenal consciousness. Not all CAV's codelets are registered at the Broadcast Manager (e.g. the behavior codelets). So the information between Behavior Network and consciousness pass through attention codelets when those codelets gain consciousness access (see figure 2). In doing so, the propositions added to the behavior network state by behavior codelets can be known by all registered codelets.

Setting goal context hierarchy

At this stage CAV updates all the new propositions which were added since the last cycle and incoporates new and more

accurate information to the behavior network. The goals are checked and updated. It is also possible to add or remove a goal following the current situation.

Action chosen

The behavior net chooses a single behavior. This choice is heavily affected by the update of the past stage. It is also affected by the current situation, external and internal conditions, by the relationship among behaviors and by the residual activation values of various behaviors.

Action taken

The execution of a behavior results in the behavior codelets performing their specialized tasks, wich may have external or internal consequences. The acting codelets also include an expectation codelet whose task is to monitor the action and bring to consciousness any failure in the expected results. CCC does not wait for the running end of a behavior codelet. CAV keeps a list of active behavior codelets and, if some particular codelet is already running, it does not start another instance of it. But it can abort a running behavior codelet, if it is necessary. For example, if a new perception makes a plan unfeasible, during the execution of a behavior codelet (let's say the vehicle is going from a point A to a point B and a new obstacle is detected), then the behavior codelet is aborted, as a new plan must be generated.

4. A Brief Analysis of CAV's Implementation

A running simulation of CAV's performance is ilustrated in figure 5. The main experiment worked as expected. The vehicle was able to pursue its main objectives: to avoid collision with obstacles while exploring the environment, and at the same time maintaining an energy balance. While exploring the environment, if the energy level decreased to a critic limit, CAV correctly postponed its exploratory behavior, looked for the closest source of energy and traced a route to it to feed itself. After refreshing its batteries, it returned to its exploratory behavior. As we said before, though, our main goal was not simply related to the achievement of these tasks (something which could be achieved by more traditional methods, as e.g. in [20]), but understanding how "consciousness" could be used in such an application.

By applying BFA to this application, we would like to evaluate the value of "consciousness" (as in BFA) to the construction of a new generation of controllers to autonomous agents. Pragmatically, we would like to understand what exactly it is this "consciousness" technology, and what the benefits to expect while applying it to control autonomous agents. This goal was also achieved while we had the experience of studying BFA and applying it to the current application. Our findings are summarized in the next subsections.

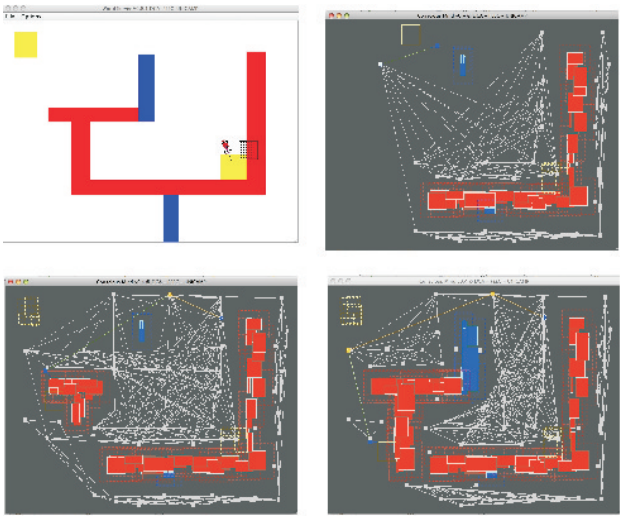


Figure 5: Example of Simulation

4.1. *Executive Summary of Perception*

Humans have the capacity of perceiving external stimuli like sounds, colors, smell and internal stimuli such as pain and feelings at the same time, and keep the focus on what is more important. Consciousness mechanism has the same role: it produces what Koch called an *executive summary of perception* [22]. During agent execution, a large number of (internal/external) sensory stimuli can be percepted by codelets. Consciousness mechanism prioritize the most relevant set of such perceptions.

4.2. *Decision-Making on Up-to-date Input Data*

It is very important in autonomous navigation problems to take decisions upon the most recent data. CAV implementation does not lock the input date updates through the entire cognitive cycle. It allows each codelet (specially attention codelets) to figure out any issue as soon as possible and then take actions toward a solution.

4.3. *A Both Parallel AND Serial Architecture*

CAV embraces both the parallel processing of codelets and the serial functioning of consciousness. Due to the quasi-independent codelets structure it is easy to evolve or substitute a certain codelet. It also easily supports the adition of new features. The serial mechanism of consciousness serializes the parallel results of codelet processing. This serialization shows the most relevant events first in order to handle the current situation.

This symbiosis, among consciousness and parallel codelets, acts according to the consciousness hypothesis defended by Dennet: "Human consciousness (...) can best be understood as the operation of a "*von Neumannesque*" virtual machine *implemented* in the parallel architecture of a brain." We suggest that the BFA is an instance of such hypothesis considering the serial outputs of the consciousness

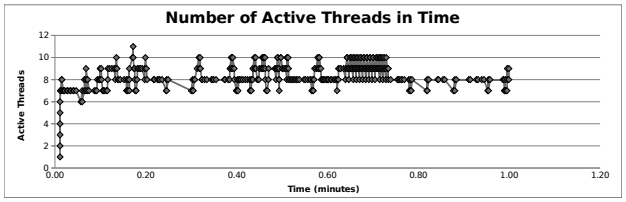


Figure 6: Number of Active Threads in Time

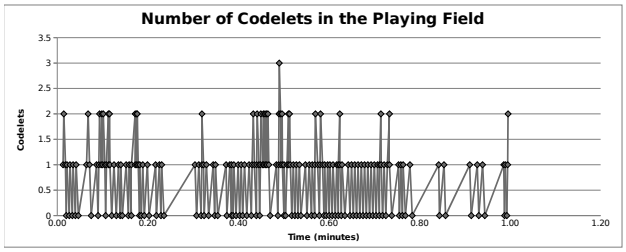
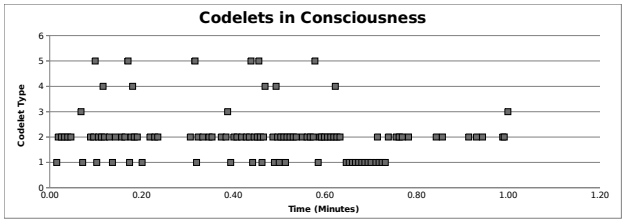


Figure 7: Number of Codelets in the Playing Field



1 - PlanGenerator 2 - ObstacleRecorder 3 - TargetCarrier
4 - CollisionDetector 5 - PathChecker

Figure 8: Types of Codelets in Consciousness

mechanisms as the output of a "von Neumannesque machine" and the codelets infrastructure the "parallel architecture of a brain".

4.4. A Quantitative Analysis

Some data related to the experiment can be viewed in figures 6, 7 and 8.

Figure 6 shows the number of active threads at each instant of time. We can see that an average of 8 threads used to be working at the same time. Figure 7 shows the number of codelets running at the same time at the playing field. An average of 1 or 2 codelets were at the playing field at the same time. The maximum of codelets at the playing field at the same time was 3. Finally, figure 8 shows the different types of codelets accessing the consciousness at each time. We can see that most of the time the codelet *ObstacleRecorder* was at consciousness. The second more frequent was *PlanGenerator*. The other three, *TargetCarrier*, *CollisionDetector* and *PathChecker* were less frequently at the consciousness.

These data refers to 1 minute of simulation. The subsequent instants of time show a more or less the same behavior. Other codelets, like e.g. *LowEnergy*, also appear from time to time, but they didn't appeared in the timeframe showed in the figure.

5. CONCLUSION

BFA showed to be a very flexible and scalable architecture, due to its consciousness and behavior network mechanisms implemented through independent codelets. Newer features can be easily included by means of newer codelets performing new roles. Consciousness mechanism makes possible a deliberation process that enable the perception of most relevant information for the current situation, building what Koch called an executive summary of perception. Much work remains to be done, specially related to a better model formalization and a better understanding of the overall role of coalitions. However, seen as an embryo of a conscious autonomous vehicle, the first results of this study show the feasibility of such technique, motivating our group to continue on this line of investigation.

Acknowledgments.

Ricardo Capitanio acknowledges a fellowship from CNPq and Ricardo Gudwin acknowledges CNPq for grant #479514/2007-0. Special thanks to Prof. Franklin and to the University of Memphis, who graciously granted us access to some IDA technology components.

REFERENCES

[1] Anthony P. Atkinson, Michael S. C. Thomas, and Axel Cleeremans. Consciousness: mapping the theoretical landscape. *Trends in Cognitive Sciences*, 4(10):372–382, October 2000.

[2] Susan Blackmore. *Consciousness - A very short introduction*. Oxford University Press, 2005.

[3] Igor Aleksander. Modeling consciousness in virtual computational machines. *Synthesis Philosophica*, 44(2):447–454, 2007.

[4] M. B. Bogner. *Realizing "Consciousness" in Software Agents*. PhD thesis, The University of Memphis, December 1999.

[5] Alain Cardon. Artificial consciousness, artificial emotions, and autonomous robots. *Cognition Process*, 7:245–267, 2006.

[6] Antonio Chella and Riccardo Manzotti. *Artificial Consciousness*. Imprint Academic, 2007.

[7] David Gamez. Progress in machine consciousness. *Consciousness and Cognition*, 17:887–910, 2008.

[8] Stan Franklin and Art Graesser. A software agent model of consciousness. *Consciousness and Cognition*, 8:285–301, September 1999.

[9] Aregahegn Seifu Negatu and S. Franklin. An action selection mechanism for "conscious" software agents. *Cognitive Science Quarterly*, 2:363–386, 2002.

[10] A. S. Negatu. *Cognitively Inspired Decision Making for Software Agents: Integrated Mechanisms for Action Selection, Expectation, Automatization and Non-Routine Problem Solving*. PhD thesis, The University of Memphis, August 2006.

[11] B. J. Baars. *A cognitive theory of consciousness*. Cambridge University Press, 1988.

[12] S. Franklin. A "consciousness" based architecture for a functioning mind. In Darryl N. Davis, editor, *Visions of Mind: Architecture for Cognition and Affect*, chapter 8, pages 149–175. Idea Group Inc (IGI), 2005.

[13] D. Dubois. *Constructing an agent equipped with an artificial consciousness: application to an intelligent tutoring system*. PhD thesis, Université du Québec à Montréal, August 2007.

[14] B. J. Baars. *In the Theater of Consciousness: The Workspace of the Mind*. Oxford University Press, 1997.

[15] O. G. Selfridge. Pandemonium: a paradigm for learning. In *Mechanism of Thought Processes: Proceedings of a Symposium Held at the National Physical Laboratory*, pages 513–526, London: HMSO, November 1958.

[16] J. V. Jackson. Idea for a mind. *ACM SIGART Bulletin*, xx(101):23–26, July 1987.

[17] D. R. Hofstadter and M. Mitchell. The copycat project: A model of mental fluidity and analogy-making. *In Holyoak, K.J & Barnden, J.A. (Eds.). Advances in connectionist and neural computation theory*, 2:31–112, 1994.

[18] P. Maes. How to do the right thing. *Connection Science Journal*, 1:3, 1989.

[19] Ricardo Capitanio Martins da Silva. Análise da arquitetura baars-franklin de consciência artificial aplicada a uma criatura virtual. Master's thesis, DCA-FEEC-UNICAMP, July 2009.

[20] R. R. Gudwin. *Contribuições ao Estudo Matemático de Sistemas Inteligentes*. PhD thesis, Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, 1996.

[21] B. J. Baars and S. Franklin. How conscious experience and working memory interact. *Trends in Cognitive Sciences*, 7(4):166–172, 2003.

[22] C. Koch. *The Quest for Consciousness - A Neurobiological Approach*. Roberts & Company Publishers, 2004.