First steps toward a cognitive architecture based on adaptive automata

Joao Eduardo Kogler Junior[1] and Reginaldo Inojosa Filho[1]

University of Sao Paulo, Escola Politecnica, Av. Prof. Luciano Gualberto, 158, Tr. 3
Sao Paulo SP 05586-090, Brazil

Abstract. We introduce the use of adaptive automata as a basis for information representation to build a cognitive architecture. We discuss some fundamental issues of the cognitive processes from the conceptual and functional points of view, with the purpose of setting an operational framework to develop methodologies for simulation and project of cognitive characters and robots.

1 Introduction

Our research effort has been targetted to the development of intelligent characters and robots that use cognition as a means to achieve their goals. We have done some prior investigation on building artificial life like characters in the past (Miranda, 2001a , 2001b), implemented as agents controled by finite-state machines. The characters developed on those works were very simple virtual creatures, each with one sensor and one effector. The sensor gathered visual input from the environment and the effector allowed the virtual creature to move on a flat surface. The finite-state machine received the results of the visual perception module and following this, it produced an output to activate the effector to move in some direction. The finite-state machine of the creature was not a pre-specified one, built following some project: it was created by an automatic procedure, which chose randomly the available states, the topology and the transitions. Depending on the resulting machine, the creatures could behave consistently with their surronding environment rules, or not, displaying sometimes an unappropriate behavior. The environment had two types of objects, with distinctive visual properties: good objects, that feed the creatures with some 'energy', and bad objects, that take some energy from them. If the energy was completely consumpted, the creature died. So, the goal of the creatures was to live as long as they could, interacting properly with these two kinds of objects. By starting so randomly, death was an expected event in this scenario. However, the individuals that reached the 'maturity', were able to mate in pairs, producing new creatures. The maturity was some pre-specified elapsed time counted from the starting of the simulation. The state-machines could also suffer mutations that eventually would improve the behavior (or not), favoring the creature. After some generations, a well-adapted character usually resulted from this evolutionary process. So, our conclusion was that the species could

Published in Angelo Loula & João Queiroz (Eds), Advances in Modeling Adaptive and Cognitive Systems. UEFS, 2010.

learn the environment rules in a long-term process, although at the cost of the lives of many individuals. The next fundamental step for this research project was to improve the creature concept in order to provide a better individual performance from the very beginning of the species. The answer, seemed to us, was to use and manage knowledge. Although we could interpret the species evolution as a cognitive process, each individual was not exactly a cognitive creature. Through this work, we have faced many challenges concerning with how to provide to the characters a method for handling specific useful information that they could have gathered during their interaction with the surrounding medium. Now we have a new proposal for tackling this problem in an appropriate way.

Our original method consisted in making the character as an information gathering agent controled by the finite-state machine implemented by an automaton. The automata were represented as strings that encoded the structure of the graphs corresponding to the respective state-transition machines. They could be improved by means of genetic optimization procedure, that takes the strings that represent the automata as the genetic codes of each one, and change them by applying mutations and crossovers during a mating between each two automata. On the long term, this resulted in machines that presented behavior very consistent with the goal achievement by the agents, which consisted in survinving as long as possible in an environment that contained both benefitial and threatening elements. Although the qualitative results had shown promising, the performance was poor concerning the long time required to produce interesting solutions. From the standpoint of the species, it would be expected an improvement after each generation, but from the individual standpoint, the possibility of improper behavior was very evident. This was mainly because the character didn't posses a mechanism for knowledge management. This means, to preserve the gathered information, process it in order to find invariants, memorize them, categorize the invariants and select among them those that could form a basis for representation of the acquired knowledge. This representation would rathaer be in a form that encodes the useful information on a manner that could be promptly used on making inferences for taking actions.

Our present proposal consists in doing the above described knowledge management process, by means of a method based on adaptive automata. The overall idea consists in taking automata as the representatives of the information gathered from the environment. Then, by using a set of appropriate operations and transformations,a new automaton it will be produced from them. These last ones will encompass the more recent information brought by the perceptual modules, with the prior knowledge encoded on other already existing automata. The resulting system will thus be a bigger automaton that acquired more states through this process. It is thus referred as an adaptive automaton.

This paper is organized as follows: on section two, we will describe the agent that represents the cognitive characters or robots. Then in section three we will present more formally the concept of adaptive automata. On section four, we will use this concept to build an agent architecture that copes with the cognitive

processes. And finally, we will propose a set of experiments that are going to be conducted, in order to test the idea and discuss some future perspectives.

2 Cognitive agents

An agent will be taken here as an useful abstraction of the character or robot. It considers only the details that are relevant for a given analysis or simulation. The minimum items that are required for the simplest agent is to have an input set (the stimuli from the environment), an output set (its actions over the environment) and a minimal set of rules to produce the output from the input. The transformation that associates the outputs to the corresponding inputs is called here the agent behavior. This transformation depends on the system internal states , which generally are not observable. The transformation involves a state-transition function and an output observer.

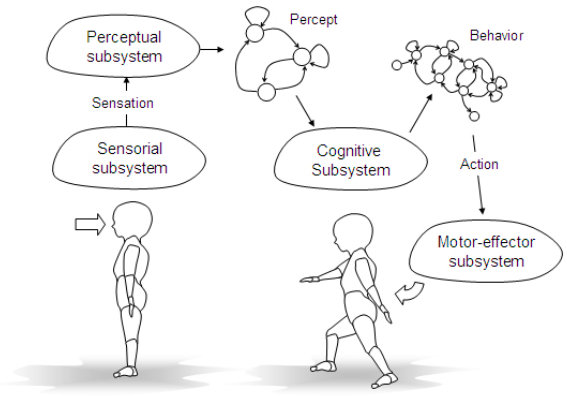

Fig. 1. The cognitive agent architecture - The agent comprises all subsystems shown here. Its interface with the environment is mediated through an avatar, indicated here by the humanoid character.

The state-transition function depends on the inputs and on past states, to generate the present state. The output observer function depends on the present state. When dealing with discrete states, one can implement the state-transition function by a finite-state machine, represented by an automaton. The overall agent behavior depends on which kind of agent is being considered (for account of agent types, see Russell and Norvig 1995). In the simplest cases of agents with internal states, the automaton implementing the finite-state machine is ultimately the responsible for generating the outputs, as a response to the inputs

(and depending on the internal states). In more complex cases, this will happen through the mediation of other process that can select or change the state machine. These are the cases of adaptive agents. More specifically, we are interested here on a particular type of adaptation strategy, that employs adaptive automata.

Our agent architecture will be organized in the following way, as depicted on figure 1. The inputs from the environment are gathered by the sensorial subsystem. This module role basically is to encode the inputs in a proper way to conform to the subsequent module, the perceptual subsystem. For instance, in the case of the visual sensing, the sensorial subsystem would provide the mapping of the available visual field into a numeric matricial representation, after the spatial sampling and amplitude quantization of the light intensity field. Then, the perceptual subsystem receives the sensorial information, called sensations, and analyse them, generating a set of several representations with different perceptual properties. These representations are then fused into a single one, called here the percept, that addresses now all kinds of uses that the cognitive system could perform with the information contained in it. The cognitive subsystem is responsible for selecting the useful information, correlate it with past information and with the acquired knowledge and performing the further reasoning that leads to the generation of the output. The main aspect that is characteristic of our architecture is that it does it by means of adaptive automata.

3 Adaptive automata

According to Neto 2001, a formal device is said to be adaptive whenever its behavior changes dynamically, in a direct response to its input stimuli, without interference of external agents, even its users. In order to achieve this feature, adaptive devices have to be self-modifiable. In other words, any possible changes in the devices behavior must be known at their full extent at any step of its operation in which the changes have to take place. Therefore, adaptive devices must be able to detect all situations causing possible modifications and to adequately react by imposing corresponding changes to the devices behavior.

3.1 Non-adaptive Rule-driven devices

This section is based in Neto 2001. A (non-adaptive) rule-driven device is any formal machine whose behavior depends exclusively on a finite set of rules which map each possible configuration of the device into a corresponding next configuration. The device is said to be deterministic if and only if, for any given initial or intermediate configuration and any input stimulus,its defining set of rules determines one and only one next configuration. The device is nondeterministic otherwise.

Formally, the non-adaptive rule-driven device is represented by:
$ND = (\mathcal{E}, \mathcal{O}, C, c_0, A, NR)$, where:

- $\mathcal{E}$ as the finite set of all possible events that are valid input stimuli for ND. The symbol ε denotes "the empty" element. In our definition, $\varepsilon \in \mathcal{E}$.
- $\mathcal{O}$ is a finite set of all possible symbols to be output by ND as result of the application of the rules in NR.
- C is the set of all its possible configurations.
- c_0 is the initial configuration of ND, with $c_0 \in C$.
- $A \subseteq C$ is the subset of its accepting configurations.
- $NR \subseteq C \times \mathcal{E} \times C \times \mathcal{O}$ is the set of non-adaptive rules. These rules $r \in NR$ have the form $r = (c_i, e, c_j, z)$, where $c_i, c_j \in C$ and $z \in \mathcal{O}$. Operationally, we have that the rule r, in response to any input stimulus $e \in \mathcal{E}$, changes the current configuration c_i to c_j, consumes e and outputs z.

The string $w = w_1 w_2 ... w_n$ is a stream of input stimuli, where $w_k \in \mathcal{E} - \{\varepsilon\}$ being $k = 1, 2...n \geq 0$. A rule $r \in NR$ is said to be compatible with the current configuration c if and only if $c_i = c$ and $e \in \mathcal{E}$ is either empty or equal to the device's current input stimulus. A valid change in configuration caused by a determined rule r is represented by notation $c_i \Rightarrow^e c_j$.

Note that e, z or both may be empty. Let $c_i \Rightarrow^{\sim} c_m$, $m \geq 0$ denote the sequence $c_1 \Rightarrow^{\varepsilon} c_2 \Rightarrow^{\varepsilon} ... \Rightarrow c_m$ an optional sequence of empty moves. Now, if $c_i \Rightarrow^{\sim w_k} c_m$ means an optional sequence of empty moves followed by a non-empty one consuming the symbol w_k, an input stream $w = w_1 w_2 ... w_n$ is said to be accepted by ND when $c_1 \Rightarrow^{\sim w_1} c_2 \Rightarrow^{\sim w_2} ... \Rightarrow^{\sim w_n} c_n \Rightarrow^{\sim} c$. The language described by ND is the set $L(ND) = \{w \in \mathcal{E}^* \mid c_0 \Rightarrow^w c, c \in A\}$ of all streams $w \in \mathcal{E}^*$ that are accepted by ND.

3.2 Adaptive Rule-driven devices

Define T as a nonempty sequence of natural numbers that starts at 0. Each value k assumed by T may index the names of time-varying sets.

The adaptive device is represented by: $AD = (ND_0, AM)$, where:

- ND_k is AD's subjacent non-adaptive device at some operation step $k \in T$. ND_0 is the AD's initial subjacent device.
- $AM \subseteq BA \times NR \times AA$ defined for a particular adaptive device AD, is an adaptive mechanism to be applied at any operation step $k \in T$ to each rule in $NR_k \subseteq NR$. AM must be such that it operates as a function when applied to any sub-domain $NR_k \subseteq NR$. This will determine a single pair of adaptive actions to be attached to each non-adaptive rule. BA (before-action) and AA (after-action) are sets of adaptive actions, both containing the null action. Adaptive actions map the current ND_k of AD into a new ND_{k+1} by applying adaptive rules.

The relationship between the AD and the ND is depicted on figure.

3.3 Adaptive Automata

State machines are very important mathematical models used for describing dynamic systems and processes. In any time,these devices have a configuration which fully determines its further behavior. A formal machine operates by successively changing the device from one configuration to another, in response to stimuli consumed from their input stream, we may state that such devices start their operation at some initial configuration that follows well-known fixed restrictions. After having processed the full input sequence of stimuli, the device reaches some configuration which may indicate that its whole input stream has been either accepted or rejected (Neto , 2001).

In the Adaptive Automata, the non-adaptive device element ND_k is a finite-state automaton (Neto and Pariente 2002, Rocha and Neto 2005) , composed of a set S of states, a finite non-empty alphabet Σ, a transition map δ, an initial state $s_0 \in S$ and a set $F \subset S$ of final states. Transitions map ordered pairs specifying the current state and the current input symbol into a new state. At each execution step of an automaton, the devices current state and the current input symbol determine a set of feasible transitions to be applied. In deterministic cases, the set is either empty (no transition is allowed) or it contains a single transition (in this case, that transition is immediately applied). In non-deterministic cases, more than one transition are allowed to be executed in parallel. In sequential implementations, a backtracking scheme chooses to apply one among the set of allowed transitions. There are two types of transitions:

- from state A to state B: transitions (A, α, B), which consume an input symbol α
- and the empty transitions (A, ϵ, B), which do not modify the input.

Adaptive actions change the behavior of an adaptive automaton by modifying the set of rules defining it. The adaptive mechanism AM of adaptive automata is defined by attaching a pair of (optional) adaptive actions to the subjacent non-adaptive rules defining their transitions, one for execution before the transition takes place and another for being performed after executing the transition. Thus, adaptive transitions make reference up to a pair of adaptive actions, $\mathcal{B}$ (before-action) and $\mathcal{A}$ (after-action). Their notation is summarized below:

$(s, \alpha) : \mathcal{A} \rightarrow (s', \alpha') : \mathcal{B}$

where $s \in S; \alpha \in \Sigma$ and $\mathcal{B}$ and $\mathcal{A}$ are the adaptive actions.

In the general case, adaptive actions $\mathcal{A}$ and $\mathcal{B}$ are representations of parametric calls to adaptive parametric functions, which have the general form $F(\Omega)$ with $\Omega = \{\varphi_1 ... \varphi_n\}$ where Ω are the n-parameters set . This one describe the modifications to apply to the adaptive automaton whenever they are called. These changes are described and executed in three sequential steps: (a) An adaptive action may be specified for execution prior to applying the specific changes to the automaton. (b) A set of elementary adaptive actions specifies the modifications performed by the adaptive action being described. (c) Another optional adaptive action may performed after the specific modifications are applied to the automaton. Elementary adaptive actions specify the actual modifications to

be imposed to the automaton. Changes are performed through three classes of adaptive actions, which specify a transition pattern against which the transitions in use are to be tested:

1. ?[*pattern*]: Inspection-type actions (introduced by a question mark in usual notation), which search the current set of transitions in the automaton for transitions whose shape match the given pattern.
2. −[*pattern*]: Elimination-type adaptive actions (introduced by a minus sign in usual notation), which eliminate from the current set of transitions in the automaton all transitions matching the given shape.
3. +[*pattern*]: Insertion-type adaptive actions (introduced by a plus sign in usual notation), which add to the set of current transitions a new one, according to the specified shape. The adaptive mechanism turn a usual automaton into an adaptive one by allowing its set of rules to change dynamically.

4 Cognitive agent architecture with adaptive automata

The main rationale behind the present idea is that the use of adaptive automata can lead to a cognitive architecture with interesting properties, concerning the easiness of conception and improvement. Usually, this kind of subject could be considered a very complex one, challenging and demanding a great effort for project and implementation. The point that shows a very promising direction is the use of a method based on a novel kind of knowledge representation and management in this context.

The agent behavior, in a more general case, is targetted to certain goals, which should be reach within a satisfactory performance. The fulfillment of these would arise from the actions executed by the agent in response to the received stimuli. So, the focus is in the performance on acting. The responsible for this performance is the internal state machine, that should be designed with optimal characteristics, relative the criteria that lead to this performance optimization. To reach the optimum there are two aspects that should be considered: (i) the functional characteristics of the system under optimization and, (ii)the existence of satisfactory solutions within the optimization space. This second issue can be properly satisfied by means of a stepwise procedure, provided that the environment would not be dynamic or if it doesn't change too fast. If this is the case, the agent can find a proper solution decomposing the problem in small temporal steps and following the input progress in this pace. What is more problematic is the first issue: the agent's state machine should have sufficient functionality to solve the problem. And this will dictate the ability of the agent to be well-adapted to a variety of environments. There are two remedies for this situation: or the agent would be restricted to live in a limited variety of environments types and configurations, or it should have adaptability enough to add new functionalities to its behavior. We are addressing here the second case, which is more general. The solution we are providing is the use of the adaptive automata to construct an evolving finite-state machine that controls the agent's behavior.

The acquisition of more functionality, as we argued, should happen in a way that the agent overall behavior could satisfy the performance optimization criteria. The difficulty encountered in this point is that the requirements for stating these criteria are not determined by the agent alone, but rather by the environment. The execution of its actions in the surrounding medium provides the challenges that could jeopardize the goal reaching by the agent. Then, the solely way of getting more functionallity is extracting it from the environment. This is the reason that lead us to formulate the model in the presented manner. The agent's perceptual module will provide the candidate information that contains the new functionality to be aided to the agent's state machine. It will produce percepts encoding this possibly useful information. The cognitive module will be the responsible by the detection, in the percepts, of the presence of new funtionalities ant to add them to the state machine. If no new funcionalities are present in the percepts, in this case they are only encoding episodic and situational information, that will be treated as a regular input by the currently existing state machine, which thus can generate the outputs from the recent inputs and based on the current state of the machine. If there are some new functionalities present in the percepts, then they will be incorporated to the state machine before the output evaluation. The mechanism that enables this scenario is provided by the adaptive automata framework.

The cognitive property of the system is resulting from this adaptation capability immanent in such framework. The purely perceptual reactive agent is capable of analysing the environment and acting based on this analysis results, employing algorithms that can vary from the simple search to the very sophiscated variational methods. However, in these cases, the best that it can do is to perform optimizations based on pre-existing functional characteristics and at most control their parameters adaptively. To take the problem-solving process in a cognitive fashion means to add more structure (in this case topological means analytic)to the agent state-transition function, gathered from the new information. This new information is considered more valuable depending on the spectrum of new situations it will enable the agent to deal with succes. Thus, invariants to a great number of transformations worth more than others. These consist on structural knowledge, the kind of information used to build representations. This is thus a clear portrait of cognition.

From the representational point of view, the perceptual subsystem implements two functions:

1. Filters the sensorial information to a data representation. Such filtering preselects only the relevant elements contained in the sensorial information. The result of this filtering is the generation of a symbol string I that encodes the sensorial information that will feed the perceptual subsystem.
2. Then the string I is converted to an automaton that has the capability of recognizing the string, Such automaton, denoted by Pi, is the percept, that will be directed as input to the cognitive subsystem.

Inside the cognitive subsystem, the percept will be incorported to the automaton denoted by M, that produces the agent's behavior. M is an adaptive

automaton and the incorporation of the percept Pi is done by means of an adaptive function that is part of M. Given that the cognitive agent has a set A of possible actions, that can be executed to produce effected actions, each new percept Pi that is incorporated by the automaton M, is also linked to one or more actions of the set A, by the insertion of a new transition that conects the two automata by two states, one of each automaton.

This step closes the process of knowledge acquisition from the system, by the cognitive agent. The following step consists in a process of self-evaluation of the automaton M. Such process is managed by the cognitive subsystem also. The structure of M is evaluated under three conditions:

1. If the formation and incorporation of the percepts was well done, so there are no useless percepts recently incorporated.
2. If the associations with the executable actions of set A are useful for the agent
3. If the structure is organized efficiently, or if there is no, or little, redundancy, duplicated percepts or parasite percepts (these last ones cause unecessary loops in the agent output generation).

In its normal functioning, the cognitive agent sensorial module will always be generating strings I, taking them and comparing with the current inputs stored in the agent's memory. In case there is no equivalent percept, the inclusion into the machine M above described will be performed. In case the percept is already existent, the corresponding associated action is executed, by generating a proper output. One of the cognitive subsystem functions is to analyse the history of the executed actions and use this analysis result in the self-evaluation process.

5 An experiment for testing the cognitive agent architecture and final remarks

In this paper we have discussed the problem of projecting an agent with cognitive capabilities in order to provide a wide spectrum of adaptability, applicable to virtual characters and robots. Our research line points to both targets, first to build characters for the computer graphics animations and computer games industries and, second, to build cognitive robots. We did several experiments with both kinds of applications in the past twenty years, however, this is the first time that we have a good proposal for a strong cognitive framework. We are considering a set of experiments to be done in order to test this framework. The first ones, which are going to appear in a forthcoming paper, consist in testing the framework with an open architecture that we have built in the past and has been available for public download in the internet (Miranda et al - 2003). The experiment consists in extracting visual informations from the environment and converting them into binary patterns that are examples of boolean functions inputs. These functions can be approximately estimated by means of an unsupervised bayesian procedure. The functions generalize the inputs gathered in certain

situations. Since we will be considering very simple ones (just the presence of two types of objects - a good one and a bad one), it is quite simple to categorize the functions into simple behaviors. Thus, the perceptual module will build two classes of pieces of adaptive automata, that can be used by the cognitive module to build the state-transition function. The results seem to be promising, but as we need more simulations to provide a final conclusive presentation, they will be left for the forthcoming publication (Kogler et al - 2008).

References

Miranda,F.R. , Kogler Jr,J.E. , Hernandez,E.M. , Netto, M.L. : An artificial life approach for the animation of cognitive characters. Computers and Graphics, Oxford, v.25, n.6, 2001 - pp. 955-964 (2001,a)

Miranda,F.R. , Kogler Jr,J.E. , Netto, M.L. , Hernandez,E.M. : ARENA and WoxBot: first steps towards virtual world simulations. In: XIV Brazilian Symposium on Computer Graphics and Image Processing, 2001, Florianopolis. Proceedings of SIBGRAPI 2001, IEEE Computer Society, 2001 (2001,b)

Miranda,F.R , Santos,C.S. , Kogler Jr., J.E. : The virtual camera project - http://camera3d.sourceforge.net - Sourceforge.net (2003)

Kogler Jr, J.E. et al : Evaluation of the performace of a cognitive open architechture applied to artificial life and robotics - In preparation (2008)

Russell,S. , Norvig,P. : Artificial Intelligence, A modern approach, Prentice-Hall, Upper Saddle River (1995)

Neto, J.J. : Adaptive rule-driven devices general formulation and case study Proc. Od the 6th conference in implementation and applications of automata, Pretoria, South Africa, 23-25 July, 2001, pp. 158-176.

Neto, J.J. , Pariente, C.A.B.: Adaptive Automata - a Revisited Proposal. Lecture Notes in Computer Science. J.M. Champarnaud, D. Maurel (Eds.): Implementation and Application of Automata 7th International Conference, CIAA 2002, 2002, Vol.2608, Tours, France, July 3-5, Springer-Verlag, 2002, pp. 158-168

Rocha, R. L. A. , Neto J. J.: An adaptive finite-state automata application to the problem of reducing the number of states in approximate string matching. CACIC 2005 - XI Congreso Argentino de Ciencias de la Computacin, Concordia, Entre Ros, Argentina, Outubro 17-21, 2005.