Construction of models based on adaptive automata through grammar descriptions

Ricardo Luis de Azevedo da Rocha; João José Neto

Escola Politécnica da Universidade de São Paulo
Av. Luciano Gualberto, trav. 3, n.158 Cidade Universitária
ZIP CODE 05508 – São Paulo – Brazil
e-mail: rlarocha@usp.br; jjneto@pcs.usp.br

SUMMARY

This research seeks to propose a formulation to an automated method of selection of solutions to problems, in order to use it as a computational device. The choice of solutions is carried out through the comparison of the automata models generated starting from the entrance of a problem in grammar form.
The generated models are adaptive, that is, the formal substratum used as basis in the method and the device proposition is the adaptive automata.

Key word: Method, Software Engineering, Model Construction, and Automata

1. INTRODUCTION

To solve problems is a task that troubles the human being since when we became conscious of our existence and of the existence of the physical world; we were faced upon the necessity to survive in a hostile and competitive atmosphere. According to Darwin, only the more adapted to the environment survive [8]. However, it can be presumed that the adaptation task is not static and absolutely inherent, that is, once an individual of certain species is born, there is no biological determinism on its destiny. There is, therefore, survival or death possibilities, in agreement with the events associated to its life and with its capacity to deal with such events.

1.1. Vision of the problem and evolution

In philosophical terms, a lot has been studied regarding conscience, and the largest difficulty is conceptual, ontological, since it is possible to study the effects of the conscience, although it is not possible to study its origin, because it is not an experience lived by a third person [6]. The critics of the idea of building intelligent machines use the difficulty of defining conscience, by its own nature, to deny the possibility to construct of such machines. John Searle affirms that the conscience is a capacity originated from brains constituted by biological neurons. It would be, therefore, prohibited to silicon brains to have conscience, unless if they could reproduce the same phenomenon's found in the biological brains [6].

In the theory of the computation, two models now in use work with the adaptation possibility: the genetic algorithms [8] (that reproduce, in computational terms, the operation of the mechanism of natural selection of the fittest found in nature, whose reproduction cells suffers genetic recombination and mutation) and the computational agents [9] [5] (a computer system, that operates in an autonomous, independent way, being capable to make decisions). In the case of the genetic algorithms, the existence of a measuring function is foreseen (it is built previously to the execution of the model) that allows establishing a comparison criterion among several solution models generated by the genetic algorithm and, so, providing means to choose the best. Despite the construction of the models imitate some of the precepts found in nature; such as the recombination and the genetic mutation to build new models, the elaboration of the measuring function is still a difficult problem.

On the other hand, the computational agents have for existence prerogative the knowledge, the belief. This way, an agent can establish its actions to autonomously face the problems that are presented.

When observing the computational models, it was noticed that for the proposal of this research, none of them would be the ideal model, although some characteristics found in each one were shown useful and, in some cases, necessary (as the self-modifying capacity). The first reasonable suggestion was to constitute a new model and a new construction method, starting from the observed models and their construction methods.

Starting from this suggestion, a comparative study of the observed models was accomplished, looking for the establishment of the ideal group of parameters for the generation of the hybrid final model. However, the comparative study didn't show how to solve the problem of generating solutions autonomously, since in all the models the solution should already be embedded or, at least inferred (through a measuring function). In [5], there is the statement saying that a device is capable to learn everything that can represent. The solution of problems is not a completely different case from learning it is an inverse problem. Thus, at this point, the followed road was to pass through a complexity study, a study about the value of information in itself.

Thereby, a theory that includes algorithms, complexity, probability and information measure, the so-called *algorithmic probability* of Ray Solomonoff [7], was retrieved.

1.2. Objectives

The concrete proposal of this research is the development of a model construction and resolution of complex problems method using an adaptive formalism [1] as basis. The main goal is to find solutions for computational problems, and being more specific, for those computational problems that can be transformed into language problems (This choice is due to the need of limiting the range of the received information that are treated by the generated device). For that, we studied some of the currently adopted methods and construction and resolution techniques of the models comparatively, and, we used characteristics of these in the intent of proposing an alternative method, preserving the studied characteristics that were considered important.

Another objective is to propose a mechanism to search for solutions, which is suitable to the proposed methodology. This mechanism should be simulated in a computer, that is, it should be a software mechanism. A prototype for the mechanism is built and tested later on, in way to validate the hypotheses of the proposal. Thus, the device BSMA (Search of Solutions for Adaptive Machine) was proposed, formalized, and an experimental prototype was built and exercised, so that this research was completed. More details can be found in [4].

2. PROPOSED METHOD

The method of investigation of solutions is proposed starting from the characteristics found in the model of the defined computational device BSMA, that, as mentioned, uses as formal basis the adaptive automaton, increased with characteristics from another lifted up models, seeking in that way to guarantee the integrity, the integration and the uniformity of the method.

2.1. Definition of the Method and Construction of the Device BSMA

The proposed method is based on induction, in the following way: For a given problem or question Q to be solved, the largest possible number of information should be collected, as for example, theories, models, examples of results. Starting from this information, we organize the picked material according to their nature, and work with an ordering of this material operating theories together with models, as if they were hypotheses in an inductive process, and with the examples, as if they were observed data. Starting from there, we have an appropriate outline to the use of Solomonoff's theory of prediction, and we can apply Bayes' rule, substituting the a priori probability by an universal measure, such as the Kolmogorov complexity for prefixes. Then, the answer to the question is provided as a prediction, if there is reachable answer for the used device [3].

When is no information about the problem, the question about how to generate an answer is solved through the use of the common sense. Thus, an answer that is accepted by most of the individuals is searched for and, then, verified if such answer is correct. The form of producing the answer through the use of common sense flees to the scope of this work.

The method to be used here will be limited in its aspect of producing solution for new problems, that is to say, the method is only used when is some information on the phenomenon or the problem in study. The use of the device BSMA, by a researcher, should be accomplished through the dispatch of models or theories, and of example data that can be suitable to train the device in the process of searching for a solution for the proposed problem. The training process can be made repeated times for the same volume of data, since the device has limited time of execution, doesn't possess common sense and nor conscience, therefore, it could not have answer even during the training process.

The operation of the device is done by means of cycles. Thus, a cycle corresponds to a complete execution of the models; that is to say, the computation steps executed by the used models compose it. On the other hand, it is necessary to impose limits on the execution time inside of a cycle, as well as a limit for the number of cycles. That is guaranteed by the existence of a time measurement device (clock), that controls every execution time.

2.1.1. Specification of the BSMA Device

The adaptive automaton, just as defined, is composed by transitions that can or cannot activate adaptive functions. In this research, when building a model based on this kind of automaton, the transitions are enumerated, that is, they are ordered and counted. This way, it is always possible to identify structural parts of the automaton model, through the ordering number of each part. In the same way, the adaptive automata models to be executed, or in execution, inside the device BSMA, are also enumerated.

There is a single controller of time (universal clock). In fact the clock counts the computation steps, and also the computation cycles (a computation cycle is composed by a group of steps, in which at least one generated model finished its processing, or the processing time limit was exhausted [4]). The following rule is established: a random change cannot be executed in two subsequent computation cycles by the same adaptive function, only by another function. This means that should have a gap between the execution of a random change and the next random change execution.

The concept of combination of automaton models is also defined starting from the original adaptive automaton model. The combination between two models of adaptive automaton is only possible among distinct pairs from the present models in the device.

A combination, or a random modification, doesn't eliminate the original models; it just increases new models to the already existent. The new introduced models should be arranged in construction order, without disturbing the previous order of models. It is specified that at the end of a computation cycle the models that didn't conclude its computation are discarded and a new order it is generated.

It is fundamental that there exists a controller element, external to the automaton models, that can manage and compute the steps and computation cycles, and that induces the combinations. Therefore, each automaton model has autonomous operation while the controller allows it. This controller can be modeled in a similar way to the Turing machine, with an infinite tape, filled with random numeric elements, of which it withdraws the values to be passed in a combination or random change. This tape can, with practical advantages, be substituted by a function that generates random numbers.

However, the controller's role isn't only that. It should evaluate the models through the results

obtained and, in that way, recognize the most suitable, discarding the others. The selected models serve as basis for the generation of the models in the next computation cycle.

Concluding the definition of the method, it is necessary, to establish the requirements of the BSMA device:

1. The device should possess a central controller;
2. The central controller should coordinate the actions to be taken in the generated models;
3. The models should be structured as adaptive automata;
4. The models should be executed concurrently or in lexicographical order;
5. The central controller should be capable to combine the models;
6. The controller can introduce random changes in the structure of the models, which are based on adaptive automata;
7. All and any change in the structure of the models can only be accomplished before a computation cycle;
8. The computation steps should compose the computation cycles, and the controller should indicate when the steps and the cycles should begin and when a cycle should finish;
9. The controller should be capable to evaluate the models, in way to choose the most capable to find a solution;
10. The execution of the models is foreseen to last a certain amount previously established of computation steps;
11. The controller should verify, at the end of the total processing limit-steps, which computations didn't still finish and conclude them, eliminating the corresponding models soon after.

Starting from those characteristics, we define the structure of the computational device that is used by the proposed method. It must be pointed out that, if any computational system is proposed with these characteristics, it can also be used as a learning system that, through an appropriate training, be capable to learn to accomplish a certain task and later on to answer to it.

2.2. BSMA Device Proposed

It is fundamental, for the purpose of finding solutions, that the device possesses a register that allows to identify if the adopted computational trajectory is adequate, that is, if it leads or not to a solution. For the specification of the register, there are three viable alternatives: Training, Learning through examples and Specification of a verification function.

The chosen alternative was the first one. By that, at each pair of values of supplied input/output, the automaton models are generated and exercised; their behaviors are compared with the expected behavior for each pair. Thus, models that present different behaviors from the expected are discarded. Each one of the models that reaches the expected answer is used to compose new solution processes.

So, the input element of the device should specify an initial construction, associated to a set of triples specified by the user of the device:

$$\{(e_i, s_i, v_i) \mid e_i - \text{input}, s_i - \text{output}, v_i - \text{value}, i \geq 1\}^n, n \geq 1.$$

The register for the measuring function operates with a set of data for which we want to find a consistent hypothesis, and with the set of automaton models, that represents the set of the lifted hypotheses. This measuring function determines, in fact, if the solution models are or not thrashing a path that takes to the goal to be reached, carrying out a role of aim maintenance.

In the Figure 1, the controller element appears as the most important item, since it centralizes all the actions of the device, triggering the exercise of the solution models through the input and identification of the problems, accomplished by the input element. Thus, the controller induces alterations in the constructions, in way to try to generate new models. In case that it doesn't find to

a solution, a cyclic processing may be needed and, even so, a solution could not be reached. In this case, the answer taken on is “No Feasible Solution”, by inexistence or impossibility of practical application.

The constructions, that represent the solution models generated by the device, are represented in the figure through the generated models (M1.. Mn), and together they compose an attaché, that is, a vector of possible solutions, even so, not necessarily, completely filled. The attachés, composed by the constructions (M1.. Mn), are independent to each other, being able to generate different solutions for the problems. The amount of existent attachés in the device depends basically on its parameter of limit of the amount of allocated space, and also of the amount of different generated solution models. The combination of both indicates the total amount of attachés, although the maximum limit is established by the parameter of limit of the amount of space.

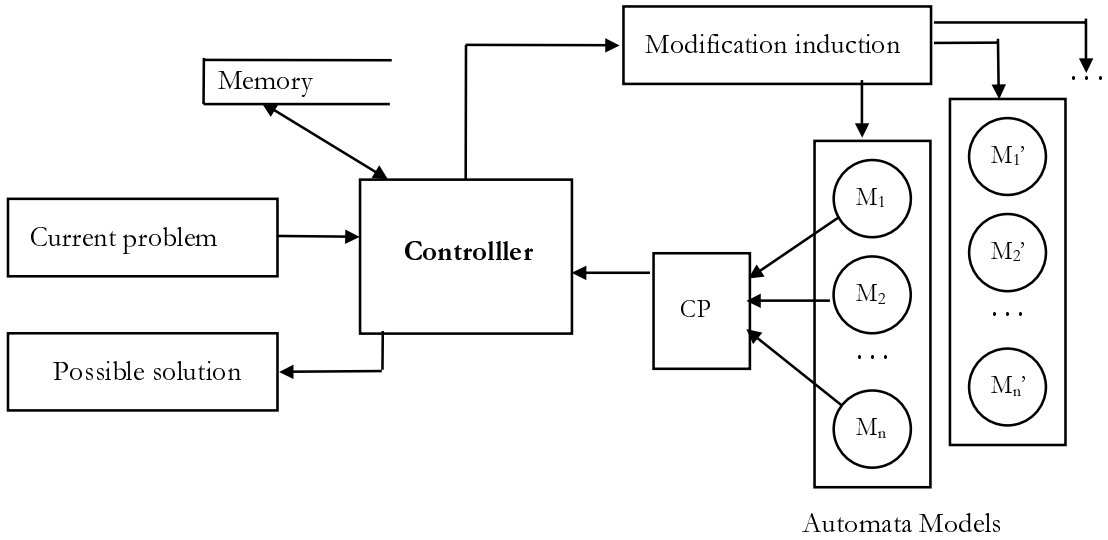


Figure 1 - Architecture of the BSMA device proposed

It can be observed that each construction inside of an attaché generates output, which doesn't pass directly to the controller element. The Figure 1 exhibits the outputs being treated for an element “CP” (comparison element). This element symbolizes, in fact, that the controller accomplishes a comparison among the exercised constructions that remained, and this allows to generate the values of the complexity measures used but not that an specific element denominated “comparison element” exists. More details can be found in [4].

2.2.1. Meaning of the Solution for the BSMA Device Proposed

The solution is, therefore, taken shape inside of an attaché, for the more adapted constructions, making with that the device always has to choose, inside of the attachés, the constructions that should remain as possible solution models, and to discard the others. As specified previously, this choice is also accomplished through the use of the measuring function, which plays the part of aim maintenance.

Again, it should be observed that, in this device, the solutions found should always be present in some construction exercised inside of an attaché, that is, it is supposed that the construction was already capable, structured to find the solution. The main idea used here is the possibility to introduce solution models into the attachés generated by the device, through alterations introduced into the constructions inside of the attachés using combination or probabilistic changes to find a capable configuration to deal with the problem in subject. The random changes have, therefore, extremely important role in the search for solutions, since, they can endow the models with different characteristics from the previously found and to enlarge the search space for solutions.

It is considered that the answer of the device can either represent a solution or barely a negative

answer, being able to, therefore, not to be the end of the task of search for solutions. This happens because, due to the used parameters or even due to the complexity of the problem, the solution cannot be feasible. However, being assumed that the problem in subject presents feasible computational solution, the output of the device will be a solution, performed through an adaptive automaton model. More details can be found in [4].

3. IMPLEMENTATION ASPECTS

The choice taken place to implement the device BSMA was to use a programming language quite scattered in the scientific community of Artificial Intelligence, that is the language LISP. The use of this language allows a faster development of the programs, since its abstraction level is higher, and it operates naturally with symbolic processing. However, the use of a compiler for the pure LISP language is not so advantageous to this work, since we would not have the facilities of debugging, of generation of graphic elements, that we have in newer compilers, which introduced functions for object orientation and the use of a graphical interface.

The proposed BSMA device was, therefore, implemented initially as a prototype, in a platform of microcomputers of the family IBM-PC and with its code being written in language LISP.

3.1. Construction of the BSMA device

The device was built in a way that its components behave in the following way:

a) There is a function for automaton interpretation previously coded, that handles the constructions in a way similar to the universal Turing machine, which executes programs and their data that describe the behavior of a particular Turing machine;
b) The automaton models are being built and modified as the computational device is unwinding its processing. Thus, starting from a specific input, the models are generated and interpreted later on. This generation, modification and interpretation procedure is accomplished until obtaining an answer.

So that the device can execute it’s processing, it is necessary that it receive the information that will allow it to generate of the automaton models. This way, the first structured element is the input, after that the controller with its functions, and at last the output element [4].

3.1.1. Analysis of the BSMA Device

When analyzing the proposed device, it is possible, by simple inspection, to define five fundamental functional classes: the class of input; the controller's class; the class of the automaton models interpreter; the class of the automaton models; and the class of the outputs.

As established previously, the proposal is to use specific grammar ontology, language ontology. This doesn't remove the generality of the principle, although in practice it reduces the implementation difficulty drastically. The generality is preserved due to the well-known equivalence among languages, grammars, automata, recursive functions as alternative forms of expression of computations. In this work, that is especially concentrated on showing the feasibility of such device, it becomes very convenient to use this simplification in this phase of the development of this research. Thus, the input element can capture the user's specifications and build a generic and possibly non-deterministic initial automaton model [4]. For the specification of the problem, it should be sent a set of three lists of arguments and a symbol (the initial symbol of the language). The lists are the following ones: List of Terminal Symbols; List of Non-terminal Symbols; List of Productions.

A specification example is the regular language composed by even amounts of elements “a”, L = (a^2)*, or using the adopted notation L = (a a)*, whose specification is: Initial symbol: s; List of Terminal Symbols: (a); List of Non-terminal Symbols: (s aa a*); List of Productions: ((s -> a*) (a* -> () (aa aa a*)) (aa -> a)).

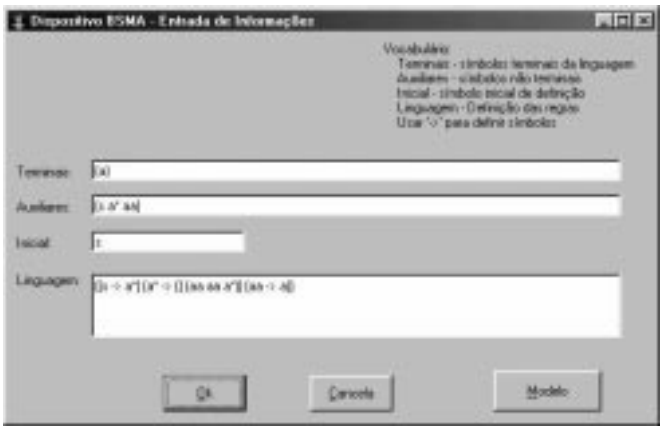


Figure 2 - Input Screen (regular language)

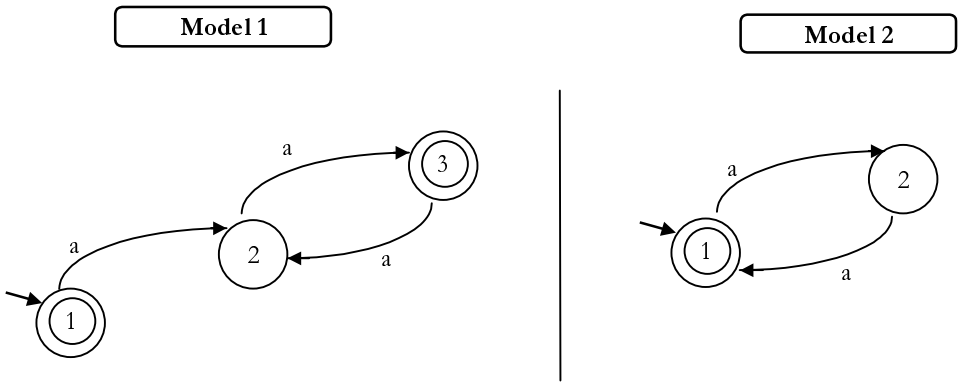
In the example in subject, the second rule of the list of production rules: (a* -> () (aa aa a*)) is interpreted as an "or" between the symbol of the empty rule "()"and the list (aa aa a*).

The test (or training) cases, which will allow exercising the suitable models and, starting from that exercise, to determine which are the more suitable, the best of them. Using the same previous example, the lists are of the type: Valid cases: ((a a) (a a a a)); Non-valid cases: ((a) (a a a)).

For the valid cases two strings were specified: "aa" and "aaaa", while for the non-valid cases were specified: "a" and "aaa".

Figure 3 - Input of training cases (regular language)

Through these specifications, the input element captures the information and passes them again to the control element, which continues with the generation stage and exercises them. In this case:



Using the criterion adopted for the measuring function, we obtain a complexity value for the first model larger than for the second model. Thereby, the second model is the best. Towards that, the

answer sent to the user is the second model, in the following way:

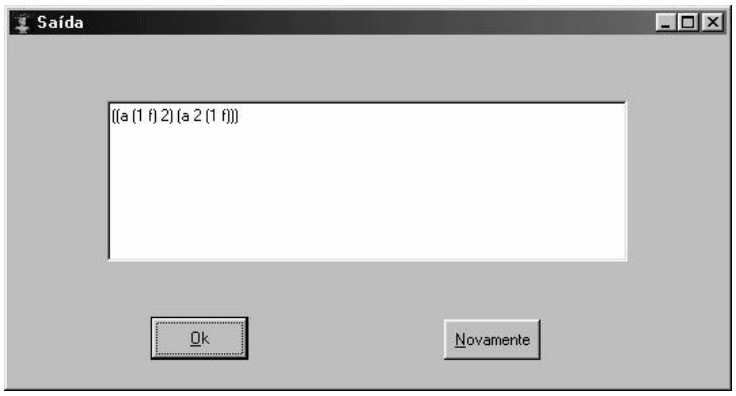
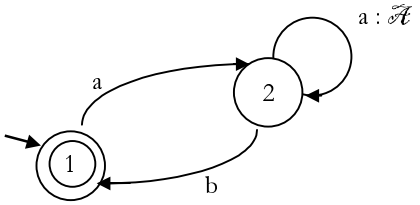


Figure 3 - Output generated by the device

This way, the generated result is the simplest model of automaton to solve the problem. In this screen, the user can close this processing up coming back to the initial screen, or to request new attempt.

In an identical way the context free languages can be considered. But in the context dependents' case adaptive functions are used. Thus, when representing a language of the type a^nb^n, it should be specified in the following way: ((s -> a*) (a* -> () (aa a* bb)) (aa -> a) (bb -> b)). The device BSMA treats a production of this type through the creation of adaptive transitions. In the exemplified case, when entering in the production a*, through the rule aa, the device generates an adaptive function that, when executed, generates a new transition that connects the current state to himself, consuming the symbol indicated by the rule aa. When leaving of the production a*, through the rule bb, the device generates an adaptive function that, when executed, eliminates a transition that connects the previous state to the current state, and consumes the symbol indicated by the rule bb. So we have the following model:



The adaptive function $\mathscr{A}$ executes four adaptive actions basically; the first of inspection, identifying the transition that consumes the symbol b and it connects a state to be determined to the final state (1). After the identification, the adaptive function eliminates this transition, and in its place it restores two other, the first connecting the state determined in the inspection action to a new state (generated), and the second connecting the state generated to the final state (1), both consume the symbol b. Thus, at each execution of the adaptive function $\mathscr{A}$ one new intermediary (not final) state is generated, and the model in execution becomes to have one transition more.

3.1.2. Execution of Models in the BSMA Device

The execution of the models is performed through a simulation. Each model is composed by its transitions (adaptive or not), and in the simulation we have a table in whose columns we have a model, a pointer to the current state of the model and an execution stack (in fact it is represented by a list), and in the lines we have the several models that should be she simulated. At each simulation step the table is traversed, and if some model cannot make a transition, then it is marked with a fault indicator (in its current state). At the end of the simulation only the models that could reach some final state, and have their stack empty, are considered accepted.

3.2. Comments on the Practical Part

For the device proposed BSMA, the way used to limit the total number of models is the choice based on the complexity measure. In the studied cases, when limiting the search space, nevertheless, the best models were present in the remaining space. This suggests that this complexity measurement can be in fact an element router, and that its use for the proposed BSMA device can aid the best choice, even if one needs to eliminate some models because of space lack, or to limit a combinatorial explosion.

An interesting and classical consideration, regarding regular languages and finite automata, can be found in [2]: a study regarding the worst case in the generation of models of deterministic automaton reveals that, for the case of being a total of R production rules in a regular grammar, to generate the best deterministic model, the algorithm should spend, in terms of time, values of the order $O(2^K)$, where K represents the number of states. When making the same calculation for the worst case of the proposed BSMA device and, considering that won't be restrictions with relationship to the number of transitions and generated models, the following is gotten: the total of generated models is of $2^{(k-1)}$, therefore, of the order $O[2^{(k-1)}]$. When comparing the two values, we have: $K = 2R$; $k = R + 2 \Rightarrow k \cong K / 2$, for $R >> 2$ (big R) [4].

This is a better result than the previous, although it is also of exponential order. However, it can be considered that the algorithm implemented by the proposed BSMA device can limit the search space enough in a feasible computation, working in a polynomial space, at the cost of decreasing the chance of finding a solution, in case the reduction is drastic in relation to the total search space [4] [10].

4. FINAL CONSIDERATIONS

In this research, a method is proposed to structure computation models, using the adaptive automata as substratum. Other important computational models, with different characteristics from the adaptive automata, were also studied to compose the method and to allow the construction of a device that could learn and to achieve solutions to complex problems.

Among the different characteristics found in another computational models we have: the possibility of introduction of random modifications, similar to a genetic mutation (found in genetic algorithms); the combination of structural parts, transitions, in the models (also found in genetic algorithms); the training possibility and the exercise of elements in parallel (found in neural nets), etc.

For other problems that involve optimization, or search for the best alternative, the BSMA device proposed is also propitious, since it always accomplishes the search for the best solution model, based on complexity reduction. It suffices to formulate the problem with this directions, that is to say, as a problem of complexity minimization expressed in language form, and to use the device.

Finally, the method and the device proposed BSMA allows the use of the adaptive automata to solve learning problems and resolution of problems, because the method foresees training. Thus a model, like the example of the decision tree, can “learn” to solve a problem and, starting from then, infer answers to subjects or situations to which it was not trained. With that, we have an alternative study in the artificial intelligence area.

Comments

At once, it is observed that a possible and important amplification is the use of natural language to formulate the user's solicitations, what will demand the elaboration of a sophisticated man-machine interface that is appropriate to that purpose.

In relation to the prototype of the BSMA device, some critics can be done. The main of them says respect to the constructions of automata: such constructions can be enlarged to best represent and

explore the adaptive automata, this way the input of problems would be enlarged through some form of context dependent specification.

5. BIBLIOGRAPHICAL REFERENCES

[1] JOSÉ NETO, J. Adaptive automata for context-dependent languages. *ACM SIGPLAN Notices*, v. 29, n. 9, p. 115-124, Sep. 1994.

[2] LEWIS, H. R.; PAPADIMITRIOU, C. H. *Elements of the theory of computation*. New Jersey, Prentice-Hall, Inc, 1998.

[3] LI, M.; VITÁNYI, P. *An introduction to Kolmogorov complexity and its applications*. 2nd. ed., New York, Springer-Verlag, 1997.

[4] ROCHA, R. L. A. *Um método de escolha automática de soluções usando tecnologia adaptativa*. São Paulo, 2000. 211p. Doctoral Dissertation – Escola Politécnica, Universidade de São Paulo.

[5] RUSSELL, S. J.; NORVIG, P. *Artificial intelligence a modern approach*. New Jersey, Prentice-Hall, Inc., 1995.

[6] SEARLE, J. *O Mistério da Consciência*. São Paulo, Martins Fontes, 1998.

[7] SOLOMONOFF, R. J. A formal theory of inductive inference - Part I and Part II. *Information Control*, v.7, p.1-22; p.224-254, 1964.

[8] SPEARS, W. M.; DE JONG, K. A.; BÄCK, T.; FOGEL, D. B.; DE GARIS, H. An overview of evolutionary computation. In: European Conference on Machine Learning, 1993. *Proceedings.* p. 442-459.

[9] WOOLDRIDGE, M.; JENNINGS, N. Formalizing the cooperative problem solving process. In: Thirteenth International Workshop on Distributed Artificial Intelligence (IWDAI-94), Lake Quinalt, WA, 1994. *Proceedings.* p. 403-417.

[10] ROCHA, R. L. A.; NETO, J. J. Uma proposta de método adaptativo para a seleção automática de soluções. Tandil - Argentine, ICIEY2K, *Proceedings*, Accepted.