Learning Regularities with a Constructivist Agent

Filipo Studzinski Perotto
Instituto de Informática
Universidade Federal do Rio Grande do Sul
Av Bento Gonçalves 9500 – Porto Alegre, Brasil
+(55) 51 3217 3301
fsperotto@inf.ufrgs.br

Luís Otávio Álvares
Instituto de Informática
Universidade Federal do Rio Grande do Sul
Av Bento Gonçalves 9500 – Porto Alegre, Brasil
+(55) 51 3316 6843
alvares@inf.ufrgs.br

ABSTRACT
This paper presents an agent learning architecture based on the constructivist approach. After a brief introduction, we show details about the proposed model. Firstly, we explain how the agent represents the knowledge. We introduce an innovative learning method, inspired by the constructivist description of the human learning process. The kind of environment regularities that can be discovered by the method is analyzed. Finally, we discuss how an extension of this agent architecture can lead to the construction of abstract concepts.

Categories and Subject Descriptors
I.2.6 [**Learning**]

General Terms
Algorithms, Theory

Keywords
Constructivist AI, Inductive Learning, Agent Architecture

1. INTRODUCTION
The Constructivist Artificial Intelligence comprises all works on this science that refer to the Constructivist Psychological Theory [6]. The constructivist conception of intelligence was brought to the field of Artificial Intelligence in the early 90's, by the Drescher's pioneer work [2]. Relevant related works have been presented in [1], [3] and [7]. This paper proposes an agent learning architecture based on the constructivist paradigm. Our model differs from the previous works because it is guided by direct induction, instead of statistical correlation. This approach has two advantages: reduces computational costs and makes it easier to find high-level deterministic regularities. A previous description of our research has been presented in [5].

2. KNOWLEDGE REPRESENTATION
Our constructivist agent tries to construct knowledge to represent perceived regularities occurring in the environment. This representation is structured in a set of schemas. Each schema represents some regularity checked by the agent during its interaction with the world. A schema is composed of three vectors: context, action and expectation. The context and expectation vectors have the same length, and each of their elements is linked with one sensor. The action vector has their elements linked with the effectors.

AAMAS'06, May 8-12, 2006, Hakodate, Hokkaido, Japan.

In a schema, the context vector represents the set of equivalent situations where the schema is applicable. The action vector represents a set of similar actions that the agent can carry out in the environment. The expectation vector represents the expected result after executing the given action in the given context. Each element vector can assume any value in a discrete interval defined by the respective sensor or effector. In addition, some elements can undertake an undefined value. For example, an element linked with a binary sensor must have one of three values: true, false or undefined (represented, respectively, by '1', '0' and '#'). The undefined value generalizes the schema because it allows ignoring some properties to represent a set of situations. For example, a schema which has the context vector = (100#) is able to assimilate the situations (1000) and (1001).

There is compatibility between a schema and a certain situation when the schema's context vector has all defined elements equal to those of the agent's perception. Compatibility does not compare the undefined elements. This strategy gives more flexibility when the agent needs to represent a set of world states as a same situation. So, there is no need of considering each possible perception combination, what reduces the problem complexity. The use of undefined values, and the consequent capability to create generalized representations to similar situations, enables the construction of a tree.

Each node in that tree is a schema, and relations of generalization and specialization guide its topology. The root node represents the most generalized situation, which has the context and action vectors completely undefined. Adding one level in the tree is to specialize one generalized element, creating a branch where the undefined value is replaced by different defined values. This specialization occurs either in the context vector or in the action vector. The structure of the schemas and their organization as a tree are presented in Figure 1.

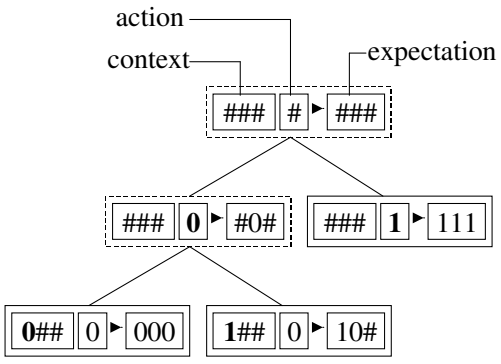

Figure 1. Schemas Tree. Each node is a schema composed of three vectors: context, action and expectation. The leaf nodes are decider schemas.

The context in which the agent is at a given moment (perceived through its sensors) is applied in the tree, exciting all the schemas that have a compatible context vector. This process defines a set of excited schemas, each one suggesting a different action to do in the given situation. The cognitive system chooses one to activate (based on an emotional value attributed to its expectations). Then the schema's action is performed (through the agent's effectors). This algorithm always chooses the compatible schema that has the most specific context, called "decider schema". Appended on each decider, there is a representation of the specific really experimented situations that happened in the past.

3. LEARNING METHODS

The learning process happens through the refinement of the set of schemas. The agent becomes more adapted to its environment as a consequence of that. After any situation, the agent's cognitive system checks if the result (context perceived at the instant following the action) is according to the expectation of the activated schema. If it fails in its prediction, the error between the result and the expectation serves as parameter either to correct the tree or to adjust the schema. If we consider context, action and expectation as a triple search space, then the learning problem is to induce a set of schemas in this search space that represents the environment regularities.

Our learning method combines top-down and bottom-up strategies. The context and action vectors are put together and this concatenated vector identifies the node in the schemas tree. The schemas tree grows up using the top-down strategy. The agent has just one initial schema. This root schema has the context vector completely general (without any differentiation) and expectation vector totally specific (without any generalization), created at the first situation experienced, according the result directly observed after the action. The context vector will be gradually specialized and differentiated. In more complex environments, the number of features the agent senses is huge, and, in general, only a few of them are relevant to identify the situation. The top-down strategy is better in this kind of environment, because it is a shorter way to begin with an empty vector and searching for these few relevant features, than to begin with a full vector and having to eliminate a lot of not useful elements. The expectation vector is equivalent to labels on the leaf nodes of the tree (called decider schemas).

The evolution of expectations uses the bottom-up strategy. Initially all different expectations are considered as different classes, and they are gradually generalized and integrated with others. The agent has two alternatives when the expectation fails. In a way to make the knowledge compatible with the experience, the first alternative is to try to divide the scope of the schema, creating new schemas with more specialized contexts. Sometimes it is not possible, then the only way is to reduce the schema expectation.

Three basic methods compose the learning function, namely: *differentiation*, *adjust* and *integration*. Differentiation is a necessary mechanism because a schema responsible for a too general context cannot have precise expectations. If a general schema does not work well, the mechanism divides it into new schemas, differentiating them by some perceptive element of context vector. In fact, the differentiation method takes an unstable decider schema and changes it into a two level sub-tree. The parent schema in this sub-tree preserves the context of the original schema. Its branches sustain the children, which are the new decider schemas. These deciders have their context vectors more specialized than their parent. They attribute a value to some undefined element dividing the scope of the original schema, each one engaging itself in a part of the domain. In this way, the previous correct knowledge remains preserved, distributed in the new deciders, and the discordant situation is isolated and treated only in its specific context. Differentiation is the method responsible to construct the schemas tree. Each level of the tree represents the introduction of some constraint into the context vector to describe compatible situations. Figure 2 illustrates the differentiation process.

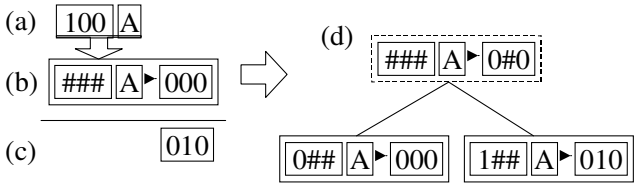

Figure 2. Differentiation method; (a) experimented situation and action; (b) activated schema; (c) real observed result; (d) sub-tree generated by differentiation.

The algorithm needs to choose what will be the differentiator element, and it could be from either the context vector or the action vector.

When some schema fails and it is not possible to differentiate it, then the cognitive system executes the adjust method. This method reduces the expectations of an unstable decider schema in order to make it more reliable. The algorithm simply compares the activated schema's expectation and the real result perceived by the agent after the application of the schema and changes the elements that are not compatible to each other for the undefined value '#'. As the mechanism always creates schemas with expectations totally determined and equal to the result of its first application, the walk performed by the schema is a reduction of expectations up to the point it reaches a state where only those elements that really represent the regular results of the action carried out in that context remain. Figure 3 illustrates that.

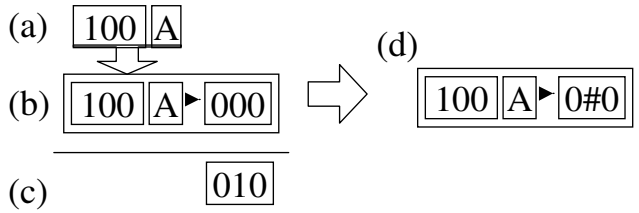

Figure 3. Adjust method; (a) experimented situation and action; (b) activated schema; (c) real observed result; (d) schema expectation reduction after adjust.

In this way, the schema expectation can change (and consequently the class of the situation represented by the schema), and the tree maintenance mechanism needs to be able to reorganize the tree when this change occurs.

Finally, successive adjustments in the expectations can reveal some unnecessary differentiations. When the cognitive system finds two schemas with similar expectations to approach different contexts, the integration method comes into action, joining these

schemas into only one, and eliminating the differentiation. The method operates as shown in figure 4.

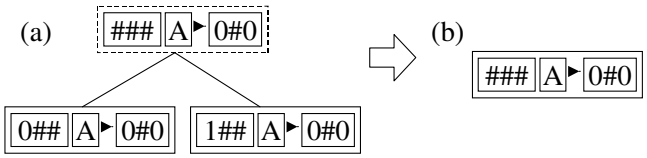

Figure 4. Integration method; (a) sub-tree after some adjust; (b) an integrated schema substitutes the sub-tree.

We have made some experiments in simple scenarios and exhaustive tests have shown that the agent ever converges to the expected behavior, constructing correct knowledge to represent the environment regularities, as well as the regularities of its body sensations, and also the regular influence of its actions over both. We may consider our results were successful, once the agent has learned about the consequences of its actions in different situations, avoiding emotionally negative situations, and pursuing those emotionally positive.

4. CONCLUSION AND FUTURE WORK

We have shown an agent architecture based on constructivist AI principles, which provides adaptive capability because the agent constructs its knowledge incrementally, by interaction with its environment. The agent has cognitive construction autonomy. We have presented a learning method composed by three functions: differentiation, adjust and integration. These functions operate over a schemas tree, which is transformed during the agent life in a way to describe the observed regularities of the world. Some experimental results have corroborated the mechanism ability to discover regularities and use this knowledge to adapt the agent behavior.

However, the learning method presented can be extended to be able to detect high-level regularities. Currently it can induce only stationary, deterministic, instantaneous and sensorimotor regularities. Therefore, we intend to introduce in our agent the following capabilities: temporal analysis, abstract concept creation, and probabilistic features. Furthermore, we want to improve the cognitive mechanism to chain the schemas, forming action sequences, and permitting the agent to create plans instead just reacting.

In the basic method, when some schema fails the first alternative is to differentiate it. If it is not possible, the mechanism repairs the schema by reducing its expectation. This alternative supposes that there is no regularity and the analyzed result is unpredictable. It is a possible explication to the fail, called "unpredictability hypothesis", but it is not necessarily true every time. In fact, there is no deterministic, instantaneous, sensorimotor regularity, but the agent must considering some alternative hypotheses, which represent other kind of regularities, and which can explain the fail.

First, the agent can postulate a “noise hypothesis”. It supposes that the error is caused by some perceptive noise, and it is a good explication to the schema’s fail if the error occurs with a very low frequency, and if the error is a random change in the element value. In this case, the best answer is to preserve the schema without modifications.

The second possible explication is the “stochastic regularity hypothesis” which says that a few different results are possible, each one with specific probabilities. The solution, if the hypothesis is right, is to create a special schema with stochastic expectations.

The third alternative is the “temporal context regularity hypothesis”. This resource permits the agent to perceive contexts distributed in the time, in addition to the capability of identifying regularities in the instantaneous perception. This hypothesis claims that is necessary to observe event sequences to explain the result of an action. It supposes that the result is a consequence of chained actions. The solution, in this case, is to utilize a special temporal schema, which improves the temporal window, analyzing more instants in the past to make its predictions.

The last option is the “abstract context regularity hypothesis”, which supposes that the error can be explained by considering the existence of some abstract property that is capable of differentiating the situation, but it cannot be directly perceived by the sensors. This abstraction requires the agent postulating the existence of a non-sensorial element. An abstract property in the agent's mind could represent two kinds of environment features: hidden properties (in partially observed worlds) and sub-environment identifiers (in non-stationary worlds).

With these new capabilities, the agent becomes able to overpass the sensorial perception constructing abstract concepts. By forming these new knowledge elements, the agent acquires new forms to structure the comprehension of its own reality in more complex levels.

After postulating these hypotheses, the agent needs to discover what of them is the correct one. It is not possible to do that a priori. The solution is to use a selectionist idea. During some time, the agent preserves all the hypotheses, and evaluates them. At the end, the best solution survives.

5. REFERENCES

[1] Chaput, H. 2004. The Constructivist Learning Architecture. PhD Thesis. University of Texas.

[2] Drescher, Gary. 1991. Mide-Up Minds: A Construtivist Approach to Artificial Intelligence. MIT Press.

[3] Holmes & Isbell. 2005. Schema Learning: Experience-based Construction of Predictive Action Models. In Advances in Neural Information Processing Systems, volume 17.

[4] Morrison, C. Oates, T. & King, G. 2001. Grounding the Unobservable in the Observable: The Role and Representation of Hidden State in Concept Formation and Refinement. In Working Notes of AAAI Spring Symposium

[5] Perotto, F. Vicari, R. Alvares, L. O. 2004. An Autonomous Intelligent Agent Architecture Based on Constructivist AI. AIAI. p.103-116

[6] Piaget, Jean. 1957. Construction of Reality in the Child. London: Routledge & Kegan Paul.

[7] Wazlawick, R. Costa, A. C. R. 1995. Non-Supervised Sensory-Motor Agents Learning. In: Artificial Neural Nets and Genetic Algorithms. New York: Springer-Verlag: 49-52.